Senet

A Programming Language for Playing Board Games

22th December 2015

Lilia Nikolova - lpn2112
Maxim Sigalov - ms4772
Dhruvkumar Motwani - dgm2138
Srihari Sridhar - ss4964
Richard Muñoz - rtm2129

Table of Contents

1. Overview

Past projects for Programming Languages and Translators have included languages for expressing the setup and flow of playing card games. Inspired by such languages, we propose to extend the domain to general, two-dimensional board games. Examples of games that might be expressed in our proposed language are tic-tac-toe, checkers, and chess. A similar idea has been investigated by Romein, Bal and Grune (1995), who described a language called *Multigame* that compiled to a parallel game playing program.[1] The authors focused their research on parallelized artificial intelligence to find optimal moving while playing games created in *Multigame*. In part due to this research focus, the authors restricted the group of games that could be described in *Multigame* to those with fixed-sized boards (thereby excluding card games) and to those where all players have perfect information.

We propose a similar language focused on simple expression of board games; however, its compiler will create games that may be played interactively on the command line. The players will execute the game program of their choice after which they will be presented with prompts that navigate them through the game. We have named our new language *Senet* after one of the oldest-known board games, which traces its origins back to ancient Egypt.

1.1 Goals

With our proposed language, we aim to provide:

1. Intuitive, relatively high-level expression of the setup and flow of board games;
2. Simple description of boards and pieces; and
3. Static, strong typing, and a mix of C and Python syntax to minimize the learning curve.

2. Language Tutorial

The following shows a “hello world” program in *Senet*.

```
@setup { }

@turns {

func void begin() {
```

[1] J. Romein, H. Bal and D. Grune. (1995). Multigame - A Very High Level Language for Describing Board Games. ASCI 95, pp. 278-287.

```
      print("Hello World\n");
      end;
}

}
```

Every senet program consists of two major sections: @setup and @turns. The former section is used to declare global functions, groups (*Senet* classes), and variables. In the latter section, a set of functions can be declared, which automatically act though they might be turns in a board game. Every *Senet* program starts with the `begin()` turn function and the end keyword in a turns function is used to terminate the program.

The other way to end a turn function is to use a pass statement, which takes a turn function and an integer (which identifies a player number). The pass statement will then execute the turn with that player number. The first player number is always 0. In *Senet*, this turn passing can be used to create interactive games between one or more players, and also change the turn type. Below is a short example of turn passing:

```
@setup { }
@turns {

func void begin() {
    if (PLAYER_ON_MOVE = 1) {
        end;
    } else {
        pass(hello, 1);
    }
}

func void ending() {
     print("Hello!"); print("\n");
     pass(begin, 1);
}

}
```

In addition to the turns function, *Senet* groups allow object-oriented construction of board games inheriting from built in groups (see Section 4.3).

3. Language Reference Manual

3.1 Lexical Conventions

There are 6 kinds of tokens: identifiers, keywords, integer literals, string literals, expression operators, and other separators. Blank, tab, and newline characters are only used to separate tokens, and are discarded by the scanner.

3.1.1 Comments

Lines (or the remainder of a line) can be commented one at a time with #. Senet does not recognize multiline comments.

3.1.2 Identifiers

Identifiers consist of a sequence of letters, digits, and underscores. They must begin with either a letter or underscore. *Senet* considers two identifiers differing only by case to be different.

3.1.3 Keywords

The following are reserved keywords in *Senet* and may not be used for any other meaning:

- `if`
- `else`
- `elif`
- `for`
- `while`
- `break`
- `continue`
- `end`
- `return`
- `True`
- `False`
- `None`
- `in`
- `int`
- `str`
- `bool`
- `void`
- `list`
- `group`

- `and`
- `or`
- `not`
- `assert`
- `func`
- `pass`
- `@setup`
- `@turns`
- `this`

3.1.4 Literals

Senet includes three kinds of literals that have fixed values: `integer`, `string`, and `list` literals. In addition, *Senet* includes a literal representing `None`. Notably, *Senet* does not support floating point literals.

Integer

An `IntLiteral` is a sequence of digits. All integer literals in *Senet* are base 10.

String

A `StrLiteral` is a sequence of characters enclosed by double quotation marks. A double quotation mark inside a string must be written as `"\""`. A newline character inside a string must be written as `"\n"`. A backslash character inside a string must be written as `"\\"`.

Void

A `VoidLiteral` represents `None`, the absence of a value.

Bool

A `BoolLiteral` takes a `bool_lit` as an argument, which in turn is either `True` or `False`.

List

A `list_lit` consists of elements of a common type separated by commas and enclosed in brackets. In addition, a `list_lit` may be the empty list of length zero (`[]`). In that case, it will have type `void`. The expression list must consist of literals or field expressions.

```
list_lit -> []
          | [ expr_list ]
```

3.2 Meaning of Identifiers

Since *Senet* is a strongly-typed language, identifiers are associated with types. There are 5 kinds of identifiers in the language: basic types, derived types, group definitions, function definitions, and group instances.

3.2.1 Basic Types

Senet includes a number of basic types inspired from C and Python as shown in the table below:

Basic Types	**Meaning**
`int`	32-bit Integer
`str`	String
`bool`	Boolean (`True` or `False`)
`void`	type of `None`, a value used to represent the absence of a value

3.2.2 Derived Types

The following table lists *Senet* types are derivatives (collections) of a basic type. The list type takes a basic type, `T`, which indicates the type of its elements.

Derived Types	**Meaning**
`list[T]`	Linked lists; e.g. `list[int] a; a = [1, 2, 3];`
`group T`	Object instance of type `group T`.

3.2.3 Group Definitions

The language is object-oriented, with inheritance (but no multiple inheritance). New groups can be defined with the `group` keyword as described in Section 3.3. *Senet* includes the standard groups shown in the table below built-in and meant to be extended by the programmer. *Senet* does not currently support virtual functions. For boards,

Standard groups	Meaning	Built-in Methods & Attributes
`Object`	Base object group; all groups extend this	`Object()`: default constructor, which contains a single statement: `return this;`

`Board`	Defines board geometries, win conditions, cleanup methods. It's constructor initializes the cells and occupied fields.	`place(Piece p, int x)`: places the piece on the board at index `x` of `b.cells`
		
		`remove(int x)`: removes the piece at index `x`, and sets `occupied[x]` to `False;`
		
		`owns(int x)`: returns the number of the player at index `x` of `cells`
		
		`full()`: returns `True` if all spots are occupied, else `False`
		
		`cells`: list of board cells, each element is either a dummy Piece instance with `s = " "` or the Piece at that spot.
		
		`occupied`: list of `bool` stating whether each cell is occupied.
		
		`toi(list[int] l)`: takes human-readable coordinate list, maps it to an internal cell index
		
		`tol(int x)`: takes an internal cell index, and maps to a human-readable coordinate list

`Piece`	Defines possible moves, keeps track of position, owning player, and other needed variables. Contains a special `__repr__()` method that returns `s`.	`s`: string representation `owner`: the owner of the piece `fixed`: whether or not the piece can be overwritten

3.2.4 Group Instances

Identifiers with types corresponding to group definitions can be created as:

```
group Id Id ( expr list ) ;

expr list -> expr
```

```
            | expr list , expr
```

If the group's constructor requires parameters,the number of expressions matching the number of parameters must be included inside the parentheses.

3.2.5 Functions

An identifier is bound to a function using syntax described in Section 3.4. The identifier can then be used to call the function using a `Call`, which is described further in Section 3.3.

3.2.6 Boards Library

Senet comes with a standard library of Board subclasses:

Board Declaration	**Meaning**
`Boards.Rect(int x, int y)`	`x` by `y` size rectangular board
`Boards.Loop(int x)`	Loop-shaped board with `x` cells
`Boards.Line(int x)`	Linear board with `x` cells
`Boards.Hex(int x)`	Hexagonal lattice of radius `x`

3.3 Expressions

The precedence of expression operators in *Senet* follows the ordering of the following subsections 4.1 to 4.10, with the highest precedence first. All operators are left associative unless otherwise specified in the subsections below.

An expression can be any of the following:

```
expr -> IntLiteral
     |  StrLiteral
     |  list_lit
     |  bool_lit
     |  VoidLiteral
     |  field_expr
     |  Binop
     |  Assign
     |  Call
     |  Element
     |  Uminus
     |  Not
```

```
|  Noexpr
|  ( expression )
|  Place
|  Remove
```

There are a number of binary operation (`binop-expr`) expressions. Because they have different precedence, the following binary operation productions will be discussed in separate subsections, along with `field_expr, Element, Call`, and `assign-expr`.

```
binop-expr -> multiplicative-expr
            | additive-expr
            | relational-expr
            | equality-expr
            | log-and-expr
            | log-or-expr
```

The most basic, and highest precedence expression are literals (`integer`, `string`, and `list`), identifiers, and parenthesized expressions. In addition, `Noexpr` represents an empty expression.

Access to an object's fields (methods or attributes) is accomplished using a period (`'.'`) between the object's identifier and the field as shown below. The type of the `field-expr` is the type of the accessed field.

```
field_expr -> Id
            | This
            | field_expr . Id
```

Functions are called as follows, with return type equal to the called function's return type. The identifier must be a function.

```
Call -> Id ( expr list )
```

3.3.1 List Element Access

A single element of a list can be accessed using an element-expression. The left expression must have type `list`. The right expression (in brackets) must have type `int`. The return type is the type of the list's elements.

```
Element -> expr [ expr ]
```

3.3.2 Unary Minus

The language includes the unary minus (`'-'`) operator that negates the value to its right. It's nonassociative. The expression operand must have type `int`. The operator is described by the following production rule:

```
Uminus -> expr
```

3.3.3 Multiplicative Expressions

Senet also includes operators for multiplication (`'*'`), integer division (`'/'`), and modulo (`'%'`). The language does not support floating point arithmetic. The expression operands must be of type `int`. The value has type `int`.

```
Mult expr -> expr * expr
           | expr / expr
           | expr % expr
```

3.3.4 Additive Expressions

Math operations in *Senet* are purely integral. The language includes operators for addition (`'+'`) and subtraction (`'-'`). The types of the expression operands must be `int`. The value has type `int`.

```
Add expr -> expr + expr
          | expr - expr
```

3.3.5 Relational Operators

Senet includes the following comparison operators: `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to). The operands must of of type `int`. The value has type `bool`.

```
Rel-op -> >
        | <
        | >=
        | <=

relational-expr -> expr Rel-op expr
```

3.3.6 Equality Operators

Senet includes the following equality operators `==` (equal to), `!=` (not equal to). The types of the operands must be the same. The value has type `bool`.

```
equality-expr -> expr == expr
               | expr != expr
```

3.3.7 Logical NOT

The logical not operator negates its operand, which must be of type `bool`. The return type is `bool`. It's nonassociative.

```
log-not-expr -> not expr
```

3.3.8 Logical AND

The logical and operator takes two operands of type `bool`, and has a value of type `bool`.

```
log-and-expr -> expr and expr
```

3.3.9 Logical OR

The logical or operator takes two operands of type `bool`, and returns a `bool`.

```
log-or-expr -> expr or expr
```

3.3.10 Assignment Expressions

The assignment operator in the language (`'='`) is right associative. The value of an assignment expression is the value of the rightmost of the expressions it is composed of.

```
assign-expr -> Id = expr
             | field_expr = expr
```

3.3.11 Board State Modification

The place (>>) and remove(<<) operators affect board state. The >> operator takes a `Piece` and places it on the board at a coordinate (described by a `list[int]`). Therefore, it requires its left operand to be derived from `Piece` and its right operand to be a `Board` and a `list[int]`, which describe the coordinate(s) on the board upon which to place the piece.

Similarly, the << operator takes a piece off the board. The left operand must be of type `Board` and the right operand must be of type `list[int]`.

These operators automatically check to see if the move is legal before performing. If it is not legal, the statement has value `False`. Otherwise, it has value `True`.

```
board-mod-expr ->
     field_expr >> field_expr >> [ list<int> ]
     field_expr << [ list<int> ]
```

3.4 Declarations

Any given declaration may be one of the following:

```
decl -> /* nothing */
      | decl vdecl
      | decl fdecl
      | decl gdecl
```

Basic-typed variables are declared as follows:[2]

```
vdecl -> type_id Id
       | type_id Id = expr ;

type_id -> int | bool | str | void
         | list<type_id>
         | group Id
```

Group definitions follow, where the identifier in parenthesis must be a group from which the new groups is extended:

```
gdecl -> group Id ( Id ){ vdecl_list fdecl_list };

vdecl_list -> /* nothing */
            | vdecl_list vdecl

fdecl_list -> /* nothing */
            | fdecl_list fdecl
```

Functions can be declared as follows:

---

[2] If `type-id` is an `identifier` or `list<identifier>` the identifier has to be a defined group.

```
fdecl -> func type-id Id ( formals_opt ){ vdecl_list stmt_list }
       | assert Id ( formals_opt ){ vdecl_list stmt_list }

formals_opt -> /* nothing */
             | formals-list

formal_list -> type-id Id
              | formal_list , type_id Id
```

There is an additional special class of `assert` functions. This special class of functions evaluate a list of conditionals sequentially and returns `True` if all of them evaluate to true. If any of the statements evaluates to false, it returns a `False` and breaks immediately.

Note that compound types (groups and lists) cannot be initialized in the setup block.

3.5 Statements

A statement is singular:

```
stmt -> expr-stmt
      | selection-stmt
      | iteration-stmt
      | jump-stmt
      | board-mod-stmt
```

A statement sequence allows many statements to be executed in order (left to right):

```
stmt_list -> /* nothing */
           | stmt_list stmt

stmt_list_req -> stmt
               | stmt_list_req stmt
```

3.5.1 Expression Statement

An expression statement executes a single expression. The expression statement has a value equal to the expression's value:

```
expr-stmt -> expr ;
```

3.5.2 Selection Statements

Branching on `if`, `elif`, and `else` are described by the following:

```
selection-stmt ->
  if ( expr ) { stmt_list }
| if ( expr ) { stmt_list } elif  (expr) { stmt_list }
| if ( expr ) { stmt_list } else { stmt_list }
```

3.5.3 Iteration Statements

An iteration statement may be:

```
iteration-stmt -> while-stmt
                | for-stmt
```

A while loop statement executes the statement in brackets repeatedly as long as the expression in parentheses evaluates to `True`.

```
while-stmt -> while ( expr ) { stmt_list }
```

A for loop statement executes the statement in brackets repeatedly similarly to the while loop statement. However, the for loop statement requires a basic-typed variable to be declared, and it executes (incrementing the declared variable each time) until the variable is greater than the upper end of the range.

```
for-stmt -> for ( type_id Id { expr_list } ) { stmt_list }

expr_list -> expr
           | expr_list , expr
```

3.5.4 Jump Statements

A jump statement may be one of the following:

```
jump-stmt -> break ;
           | continue;
           | return expr ;
```

A `break` statement must be inside of a `for` or `while` loop. The effect of this statement is to terminate the loop and execute the next statement after the loop. A `continue` statement must be inside of a `for` or `while` loop. The effect of this statement is to jump to the begin of the next loop iteration. A `return` statement must be inside of a function block.

3.6 Program Structure and Scope

3.6.1 Structure

A *Senet* program consists of the following simple structure:

```
program -> setup-block turns-block EOF
```

Thus, all games must have two program sections: the `setup-block`, which contains global functions, groups, and parameters used to set up the game; and the `turns-block`, which contains a `function-list` which describe turn "phases" functions each of which operate as a "`while True`" loop but can call other functions in the `@turns` section. One of the phase functions must be named `begin`, this will be called when the game begins.

```
setup-block -> @setup { declaration-list statement-list }
turns-block -> @turns { function-list }
declaration-list -> /* nothing */
                  | declaration-list declaration
```

3.6.2 Scope

Identifiers declared in the `setup-block` are visible to the remainder of the program, in the order they are declared. First, the variable declarations are output, followed by groups, and then functions. Identifiers declared within functions are visible only to each function. Identifiers declared as part of groups can be accessed using the field access operator whenever the group instance can be accessed. Within a function that is within a group definition, other identifiers that are part of the group can be called via a field-expression using the `this` keyword. Instances of group definitions can only be created if the group definition can be accessed.

3.6.3 Other Built-Ins

A number of library functions are built into the language:

- `clear_input()`
- `exit()`
- `read(int)`
- `stoi(str)`
- `print(expr list)`
- `rand()`

To begin the game over, use `restart`. To quit the program, use exit. To read `x` number of integers from standard input, use `read(x)`. To convert a string `s` to an integer, use `stoi(s)` -- note that `s` must be a string of one character only. To pass a string s to standard output, use `print()`, which takes an expression list. `clear_input()` is useful when interacting with players so discard the remainder of standard in. `rand()` is the same as its C counterpart.

In addition, one variable is built into the language to control the flow of games:

- `PLAYER_ON_MOVE`

PLAYER_ON_MOVE begins at 0 and is used to know which player is currently taking his or her turn. The programmer can use a `pass` statement to change this variable.

3.7 Example Program

Tic-tac-toe is a two-player game that is played on a three row, three column board. The players take turns placing either an “X” or an “O” in each cell. A player wins if three of their pieces fall in a line (vertical, horizontal, or diagonal). The game ends in a draw if all cells are full and no player has won. Below, we describe how our language could be used to create an interactive tic-tac-toe game.

```
@setup
{

list[group line] victory_conds;

group line(Object) {
    list[int] loci;

    func group line __init__(list[int] l) {
        this.loci = l;
        return this;
    }
};

group ttb(Rect(3, 3)) {

    assert owner_of_line(group line l, int player) {
        this.owns(l.loci[0]) == player;
        this.owns(l.loci[1]) == player;
        this.owns(l.loci[2]) == player;
    }

    func bool three_in_a_row(int player) {
        int i = 0;
        group line l;
```

```
        while (i < 8) {
            l = victory_conds[i];
            # print("DEBUG: three_in_a_row, i = "); print(i); print("\n");

            if (this.owner_of_line(l, player)) {
                return True;
            }
            i = i + 1;
        }
        return False;
        }

    func bool won(int player) {
        if (this.three_in_a_row(player)) {
            return True;
        }
        return False;
    }

    assert draw() {
        this.full();
    }

    func str __str_of_row(int row) {
        group Piece l; group Piece m; group Piece r;
        str ret;
           # print("DEBUG: start of __str_of_row with row = "); print(row);
print("\n");
        # print("DEBUG: print [0, row] = "); print([0, row]); print("\n");
        l = this.cells[this.toi([0, row])];
        # print("DEBUG: l = this.cells call ok\n");
        m = this.cells[this.toi([1, row])];
        r = this.cells[this.toi([2, row])];
        ret = "[" + l.__repr__() + ", " + m.__repr__() + ", " + r.__repr__() +
"]\n";
        return ret;
    }

    func str __repr__() {
                    return  this.__str_of_row(0)  +  this.__str_of_row(1)  +
this.__str_of_row(2);
    }
};

group Mark(Piece) {
    func group Mark __init__(str s, int player) {
        this.s = s;
        this.owner = player;
        return this;
    }
};

int N_PLAYERS = 2;
```

```
group ttb b;

}

@turns
{

func void begin() {
    group line v0; group line v1; group line v2; group line v3;
    group line v4; group line v5; group line v6; group line v7;
    int x; int y; int z;
    # list[group line] victory_conds;

    b = ttb();

    x = b.toi([0, 0]); y = b.toi([1, 0]); z = b.toi([2, 0]);
    v0 = line([x, y, z]);
    x = b.toi([0, 1]); y = b.toi([1, 1]); z = b.toi([2, 1]);
    v1 = line([x, y, z]);
    x = b.toi([0, 2]); y = b.toi([1, 2]); z = b.toi([2, 2]);
    v2 = line([x, y, z]);
    x = b.toi([0, 0]); y = b.toi([0, 1]); z = b.toi([0, 2]);
    v3 = line([x, y, z]);
    x = b.toi([1, 0]); y = b.toi([1, 1]); z = b.toi([1, 2]);
    v4 = line([x, y, z]);
    x = b.toi([2, 0]); y = b.toi([2, 1]); z = b.toi([2, 2]);
    v5 = line([x, y, z]);
    x = b.toi([0, 0]); y = b.toi([1, 1]); z = b.toi([2, 2]);
    v6 = line([x, y, z]);
    x = b.toi([0, 2]); y = b.toi([1, 1]); z = b.toi([2, 0]);
    v7 = line([x, y, z]);

    # print("DEBUG: v7 ok\n");

    victory_conds = [v0, v1, v2, v3, v4, v5, v6, v7];

    # print("DEBUG: built victory_conds\n");

    # b = ttb();

    # print("DEBUG: built b\n");

    pass(prompt, 0);
}

func void prompt() {
    int a;
    int c;
    group Mark m;
    int i;

    if (PLAYER_ON_MOVE % 2 == 0) {
```

```
        m = Mark("X", PLAYER_ON_MOVE);
    } else {
        m = Mark("O", PLAYER_ON_MOVE);
    }

    # print("DEBUG: start of prompt()\n");

    print("\n"); print(b); print("\n");

    # print("DEBUG: printed b\n");

    # players input moves by typing coordinates, e.g. "11" or "02"
    print("PLAYER "); print(PLAYER_ON_MOVE); print(": ");
    print("Input coordinates of square to place ");
    print("in i.e. \"22\" or \"10\".\n");
    a = stoi(read(1));
    c = stoi(read(1));
    clear_input();
    # print("index of input: ["); print(a); print(", "); print(c); print("] ->
");
    # print(b.toi([a, c])); print("\n");

    if (m >> b >> [a, c]) {

        # PLAYER_ON_MOVE is the index of the player
        if (b.won(PLAYER_ON_MOVE)) {
            pass(winner, PLAYER_ON_MOVE);
        }
        if (b.draw()) {
            pass(nowinner, PLAYER_ON_MOVE);
        }
    } else {
        # A piece is already at [a, c]
        print("Cannot place a mark there, try again.\n\n");
        pass(prompt, PLAYER_ON_MOVE);
    }
    # if the move was legal, went through successfully,
    # and the game is not over, pass the turn to the next player
    pass(prompt, (PLAYER_ON_MOVE + 1) % N_PLAYERS);
}

func void winner() {
    print("\n"); print(b); print("\n");
    print("Player "); print(PLAYER_ON_MOVE); print(" wins.\n");
    print("Congratulations!\n");
    end;
}

func void nowinner() {
    print("\n"); print(b); print("\n");
    print("Game ends in a draw.\n");
    end;
}
```

```
}
```

4. Project Plan

4.1 Process

4.1.1 Planning

During the first half of the semester our group met once a week on Fridays. We met in the mornings and worked as long as we needed to in order to complete the necessary work for each assignment that we had to turn in. This served as a good guide for the progression of our project throughout the semester. We set weekly tasks based on the deliverables and tried to complete most of them during our meetings. This allowed us to brainstorm ideas, share our knowledge, and contribute to various aspects of the project. There were weeks when we met with our mentor Lixin Yao in order to make sure that we were on the right track or consult about problems that we weren't able to figure out ourselves.

During the second half of the semester we met twice a week. We found it was necessary because we started working on developing and testing our code. Once again, we worked together during those meetings and although our weekly goals weren't as clearly outlined as before we progressed by building on our first "hello world" test case.

4.1.2 Specification

Even though we had a good idea of what we wanted our language to do and the types of building blocks required, the construction of the specification for our language was not an easy project and was one that took the whole semester to refine. The task of writing the Language Reference Manual helped us with figuring out the syntax and semantics but it wasn't until we started coding each aspect of the compiler that we realized the mistakes we had made initially. We kept updating our LRM almost every time something changed as we were developing each component (scanner, AST, parser, etc.). We also defined a standard library for our language and did a second major revision of the LRM after we had coded and tested everything.

4.1.3 Development and Testing

Development followed the stages specified in the class lectures. We started with developing the scanner, parser, AST, and enough compiling to C in order to print out the "hello world" message. From there, we expanded the code within the components to handle all the necessary test cases and include the required functionality to build a working program.

4.2 Style Guide

We generally used the following conventions while programming our compiler in order to ensure consistency, readability, and transparency:

- Ocaml editing and formatting style was generally followed when writing Ocaml code.
- C style editing and formatting style was used when writing Senet code.

4.3 Project Timeline

Date	Task
September 30th	Submitted Project Proposal
October 26th	Submitted Language Reference Manual
October 26th	Compiler Front End Complete
November 13th	Code Generation for Hello World Complete
November 13th	Hello World Runs
November 16th	Hello World Presentation
December 20th	Code Generation Complete
December 20th	Testing and Debugging Complete
December 21st	Presentation Complete
December 22nd	Submitted Final Project

4.4 Roles and Responsibilities

A big part of our project was done by the team working in the same physical space and a lot of our roles overlapped because everyone did a little bit of everything. The table below provides information about each team member's main responsibilities as outlined in the beginning of the semester along with specific aspects of the project they were involved in:

Team Member	Role	Responsibilities
Lilia Nikolova	Manager	Timely completion of deliverables, team organization, documentation, testing

Maxim Sigalov	Language Guru	Language design, semantics
Dhruvkumar Motwani	Language Guru	Language design, standard library
Srihari Sridhar	System Architect	Compiler architecture, standard library, code generation, semantics
Richard Muñoz	Verification and Validation	Test plan, test suites, code generation, semantics

4.5 Development Environment

- Github repository - version control; contains compiler code, tests, program examples, and standard library.
- OCaml 4.02.1 - parsing and semantic checking.
- gcc-5 - building C output of compiler.
- Valgrind - debugging C code.

4.6 Project Log

Date	**Task**
September 18th	Determined language function and purpose
September 25th	Determined main language features
September 30th	Completed language proposal
October 2nd	Defined grammar
October 9th	Defined grammar
October 16th	Compiler front end, semantics
October 23rd	Compiler front end, semantics, type checking
October 26th	Completed Language Reference Manual
October 26th	Compiler front end complete
October 30th	Hello World code generation
November 3rd	Hello World code generation
November 6th	Hello World code debugging
November 13th	Code generation for Hello World complete

November 13th	Hello World runs
November 16th	Hello World presentation
November 20th	Code generation, testing
December 24th	Code generation, testing
December 1st	Code generation, testing
December 4th	Code generation, testing
December 8th	Code generation, testing, standard library
December 11th	Code generation, testing, standard library
December 15th	Code generation, testing, debugging
December 18th	Code generation, testing, debugging
December 20th	Code generation complete
December 20th	Testing and debugging complete
December 21st	Presentation complete
December 22nd	Submitted final project

5. Architectural Design

5.1 Compiler Block Diagram

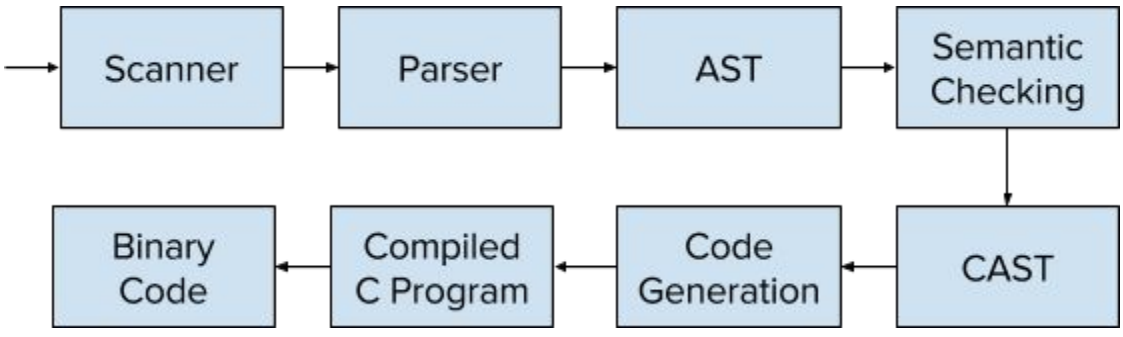

Figure 1 : Architecture of Senet compiler

5.2 Scanner

The scanner, implemented in OCamlLex takes Senet code as input and tokenizes it into defined keywords in the language, literals and constants. It also discards white spaces and designated comments in the language. These tokens are then passed on to the parser for building the Abstract Syntax Tree.

5.3 Parser

The parser takes the tokens from the scanner and produces the Abstract Syntax Tree using the grammar definitions in the parser.mly file. Additionally, this checks for consistency between the data types used in the program and the valid definitions listed in the ast.ml file.

5.4 AST and Semantic Checks

The Abstract Syntax Tree is the intermediary representation used by the parser and the subsequent stages of the compilation process which includes only the most relevant aspects of the tokenized code received from the scanner. Each individual AST type is listed as a node in the a tree representation. Semantic checks are run on this representation to check for consistency with the grammar rules defined for the language.

5.5 CAST and Code Generation

These two components encompass the backend of the compilation process where the semantically checked AST representation is converted modified into another AST that aligns with the C syntax to help ease the transition from Senet to C code that happens in the code generation process. In particular, the ordering of group attributes is standardized so that C can cast pointers between them. In addition, expressions with a group type are filled in with the full SAST group declaration in order to facilitate accessing attributes and functions and casting. Also, some list literals are tagged with temporary names for the C code to use to build linked lists (since the C program needs memory locations to use the address operator). From this intermediary representation, relevant C code is generated which can then be compiled and executed the same way as any generic C program.

6. Test Plan

First, below is the test file for assert functions, test-assert-func.snt:

```
@setup {
assert test() {
    True;
```

```
}

assert test2() {
    2 + 2 > 3;
    4 - 2 < 0;
    2 == 2;
}

assert test3(int x) {
    if (x > 3) { True; } else { False; }
}

}

@turns {

func void begin() {
    print(True); print("\n");
    print(test()); print("\n");
    print(test2()); print("\n");
    print(test3(2)); print("\n");
    end;
}

}

@turns {

func void begin() {
    print(True); print("\n");
    print(test()); print("\n");
    print(test2()); print("\n");
    print(test3(2)); print("\n");
    end;
}

}
```

The following shows the C output for compiling test-assert-func.snt:

```
// @senet_header
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include "temp/sen_linked_list.h"
#include "temp/sen_print_base_grps.h"
#include "temp/sen_init_base_grps.h"
#include "temp/sen_read.h"

struct SENET_NONE {
  } SENET_NONE;

struct Sen_list snt_SEN_EMPTY_LIST;

char *SENET_STR_CONCAT(char* s1, char* s2) {
  char *temp = (char * ) malloc(strlen(s1)+ strlen(s2) +1);
  strcpy(temp, s1);
  strcat(temp, s2);
  return temp;
}

void (*CUR_TURN)();
int snt_PLAYER_ON_MOVE = 0;

// @setup

bool  snt_test() {

if (!(true)) { return false; }

return true;
}

bool  snt_test2() {

if (!(((2 + 2) > 3))) { return false; }

if (!(((4 - 2) < 0))) { return false; }

if (!((2 == 2))) { return false; }

return true;
}

bool  snt_test3(int snt_x) {

if ((snt_x > 3) ) {
  if (!(true)) { return false; }

} else {
  if (!(false)) { return false; }
```

```
}

return true;
}

// @turns
void  snt_begin();

void  snt_begin() {

printf("%s", true ? "true" : "false");
printf("%s", "\n");
printf("%s", snt_test() ? "true" : "false");
printf("%s", "\n");
printf("%s", snt_test2() ? "true" : "false");
printf("%s", "\n");
printf("%s", snt_test3(2) ? "true" : "false");
printf("%s", "\n");
exit(0);
}

// @senet_footer
int main() {
  CUR_TURN = &snt_begin;
  snt_PLAYER_ON_MOVE = 0;
  while (true) {
    CUR_TURN();
  }
  return 0;
}
```

Now, here is a second example of a test script, test-inherit-three-deg.snt, which tests inheriting functions through a grandchild.

```
@setup {

group A(Object) {
    int x;

    func group A __init__(int x) {
        this.x = x;
        return this;
    }

    func int test() {
```

```
        return this.x + 5;
    }

};

group B(A) {
    int y;
};

group C(B) {
    int z;
};

group A obj;
group B obj2;
group C obj3;

}

@turns {

func void begin() {
    int y;

    obj.__init__(1);
    obj2.__init__(2);
    obj3.__init__(3);

    print(obj.x); print("\n");
    print(obj2.x); print("\n");
    print(obj3.x); print("\n");

    print(obj.test()); print("\n");
    print(obj2.test()); print("\n");
    print(obj3.test()); print("\n");

    end;
}

}
```

The C output for test-test-inherit-three-deg.snt:

```
// @senet_header
#include <stdbool.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>
#include "temp/sen_linked_list.h"
#include "temp/sen_print_base_grps.h"
#include "temp/sen_init_base_grps.h"
#include "temp/sen_read.h"

struct SENET_NONE {
  } SENET_NONE;

struct Sen_list snt_SEN_EMPTY_LIST;

char *SENET_STR_CONCAT(char* s1, char* s2) {
  char *temp = (char * ) malloc(strlen(s1)+ strlen(s2) +1);
  strcpy(temp, s1);
  strcat(temp, s2);
  return temp;
}

void (*CUR_TURN)();
int snt_PLAYER_ON_MOVE = 0;

// @setup
struct snt_A snt_obj;
struct snt_B snt_obj2;
struct snt_C snt_obj3;

struct snt_A{
  int snt_x;
} snt_A;

char*  snt_A_snt___repr__(struct snt_A *this) {

return "<Group A instance>";
}

struct snt_A  snt_A_snt___init__(struct snt_A *this, int snt_x) {

(*this).snt_x = snt_x;
return (*this);
}

int  snt_A_snt_test(struct snt_A *this) {

return ((*this).snt_x + 5);
}
```

```
struct snt_B{
  int snt_x;
  int snt_y;
} snt_B;

char*  snt_B_snt___repr__(struct snt_B *this) {

return "<Group B instance>";
}

struct snt_B  snt_B_snt___init__(struct snt_B *this, int snt_x) {

(*this).snt_x = snt_x;
return (*this);
}

struct snt_C{
  int snt_x;
  int snt_y;
  int snt_z;
} snt_C;

char*  snt_C_snt___repr__(struct snt_C *this) {

return "<Group C instance>";
}

struct snt_C  snt_C_snt___init__(struct snt_C *this, int snt_x) {

(*this).snt_x = snt_x;
return (*this);
}

// @turns
void  snt_begin();

void  snt_begin() {

int snt_y;
snt_A_snt___init__((struct snt_A *) &snt_obj, 1);
snt_B_snt___init__((struct snt_B *) &snt_obj2, 2);
snt_C_snt___init__((struct snt_C *) &snt_obj3, 3);
printf("%d", snt_obj.snt_x);
printf("%s", "\n");
printf("%d", snt_obj2.snt_x);
printf("%s", "\n");
printf("%d", snt_obj3.snt_x);
```

```
printf("%s", "\n");
printf("%d", snt_A_snt_test((struct snt_A *) &snt_obj));
printf("%s", "\n");
printf("%d", snt_A_snt_test((struct snt_A *) &snt_obj2));
printf("%s", "\n");
printf("%d", snt_A_snt_test((struct snt_A *) &snt_obj3));
printf("%s", "\n");
exit(0);
}

// @senet_footer
int main() {
  CUR_TURN = &snt_begin;
  snt_PLAYER_ON_MOVE = 0;
  while (true) {
    CUR_TURN();
  }
  return 0;
}
```

In total, the test suites used are below. We generally added tests for each new feature we added to the language or when we discovered that features that appeared to be work did not. The test automation uses a modified version of the `testall.sh` script from microc.

7. Lessons Learned

7.1 Lilia Nikolova

Scheduling and communication within a team are crucial elements when working on a large-scale group project. You have to make sure that everyone is on the same page at all times and not assume or expect that everyone knows exactly what to do. This is achieved by being proactive (in terms of reaching out and coordinating times to meet) and assertive (in terms of assigning tasks and making sure they are complete). There's no room for shyness.

7.2 Maxim Sigalov

Ambition needs to be checked by realistic expectations. A desire for more and more features can thin out work and make it harder to assemble a working subset compliant with the desired specification. At the same time, forward thinking necessary in order to avoid having to change large amounts of code to allow for more advanced features. Classes with true inheritance and virtual tables are not as easy as languages like C++ and Python make, and the temptation to overuse macros is hard to resist.

7.3 Richard Muñoz

Creating test scripts was essential to making progress. It was nice to see that small pieces of the compiler began to work. Translating code to C proved to be more difficult that I anticipated. I would advise next year's groups to spend time during the LRM stage to specify as many details as possible, since they are needed in order to translate code.

7.4 Srihari Sridhar

It is hard to write good code. It is infinitely harder to build a compiler that recognizes what good code is. Delving into the nuances of what used to be a mere click of the Compile button was a really enlightening experience. From here on, I am fairly sure a miniaturized version of our wrestling matches with OCaml and C will flash before my eyes every time I compile a piece of code. On the management side, when you have a bunch of really talented people to work with, fixed roles and responsibilities can end up being restrictive. Working together as often as possible and solving each other's problems was crucial to our success.
Lastly, when you think you know C, think again!

7.5 Dhruvkumar Motwani

Creating a programming languages is a daunting task in a semester. I learned that the best way to move forward in any project is to allow the best developer(s) to lead and support him/her in all possible ways. The project also allowed me to learn a lot about the production level code since we ended up breaking our code multiple times for small errors and had to spend hours debugging it.

8. Appendix

./ast.ml

```
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater |
    Geq
          | Mod | And | Or

type id_type =
    Int
  | Bool
  | Str
  | Void
  | List of id_type
  | Group of string

type bool_lit =
    True
  | False

type field_expr =
    Id of string
  | This
  | FieldCall of field_expr * string

type list_lit =
    Elems of expr list
  | EmptyList

and expr =
    IntLiteral of int
  | StrLiteral of string
  | ListLiteral of list_lit
  | BoolLiteral of bool_lit
  | VoidLiteral
  | Field of field_expr
  | Binop of expr * op * expr
  | Assign of field_expr * expr
  | Call of field_expr * expr list
  | Element of expr * expr
  | Uminus of expr
  | Not of expr
  | Noexpr
  | Remove of field_expr * list_lit
  | Place of field_expr * field_expr * list_lit

type stmt =
    Block of stmt list
  | Expr of expr
  | Return of expr
  | Break
  | Continue
  | If of expr * stmt * expr option * stmt
  | For of var_decl * expr list * stmt
```

```
  | While of expr * stmt
  | End
  | Pass of string * expr

and init =
  | ExprInit of expr
  | NoInit

and var_decl = {
    vname : string;
    vtype : id_type;
    vinit : init
}

type basic_func_decl = {
    ftype : id_type;
    fname : string;
    formals : var_decl list;
    locals : var_decl list;
    body : stmt list;
  }

type assert_decl = {
    fname : string;
    formals : var_decl list;
    locals : var_decl list;
    body : stmt list;
  }

type func_decl =
    BasicFunc of basic_func_decl
  | AssertFunc of assert_decl

type group_decl = {
    gname : string;
    extends : field_expr option;
    par_actuals : expr list option;
    attributes : var_decl list;
    methods : func_decl list;
  }

type setup = var_decl list * func_decl list * group_decl list

type turns = func_decl list

type program = setup * turns

let rec escaped_string s =
  Printf.sprintf "%S" s

let rec string_of_vtype = function
    Int -> "int"
  | Bool -> "bool"
  | Str -> "str"
```

```
  | Void -> "void"
  | List(vt) ->
      "list[" ^ string_of_vtype vt ^ "]"
  | Group(s) -> s

let rec string_of_field = function
    Id(s) -> s
  | This -> "this"
  | FieldCall(f,s) -> string_of_field f ^ "." ^ s

let rec string_of_list_lit = function
    EmptyList -> "[]"
  | Elems(e) ->
        "[" ^ String.concat ", " (List.map string_of_expr e) ^ "]"

and string_of_expr = function
    IntLiteral(l) -> string_of_int l (*[?  |  ?]*)
  | Field(f) -> string_of_field f
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^
      (match o with
        Add -> "+" | Sub -> "-" | Mult -> "*" | Div -> "/"
      | Equal -> "==" | Neq -> "!="
      | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">="
      | Mod -> "%"
      | And -> "and" | Or -> "or" ) ^ " " ^
      string_of_expr e2
  | Assign(f, e) -> string_of_field f ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      string_of_field f ^
      "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""
  | StrLiteral(s) -> escaped_string s
  | Uminus(e) -> "-" ^ string_of_expr e
  | Not(e) -> "not" ^ string_of_expr e
  | Element(e1, e2) ->
    string_of_expr e1 ^ "[" ^ string_of_expr e2 ^ "]"
  | ListLiteral(l) -> string_of_list_lit l
  | BoolLiteral(b) -> (match b with True -> "True" | False -> "False")
  | VoidLiteral -> "None"
  | Place(f1, f2, l) ->
      string_of_field f1 ^ " >> " ^ string_of_field f2 ^ " >> " ^
        string_of_list_lit l
  | Remove(f1, l) ->
      string_of_field f1 ^ " << " ^
        string_of_list_lit l

let rec string_of_vinit = function
    NoInit -> ""
  | ExprInit(e) -> " = " ^ string_of_expr e

let rec string_of_vdecl vdecl =
  string_of_vtype vdecl.vtype ^ " " ^ vdecl.vname ^
  string_of_vinit vdecl.vinit
```

```
158
159 let rec string_of_stmt = function
160     Block(stmts) ->
161       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
162   | Expr(expr) -> string_of_expr expr ^ ";\n";
163   | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
164   | If(e, s, None, Block([])) ->
165       "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
166   | If(e, s1, e2, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
167       string_of_stmt s1 ^
168       (match e2 with
169         None -> "else\n"
170       | Some(expr) -> string_of_expr e) ^
171       string_of_stmt s2
172   | For(vd, elist, s) ->
173       "for (" ^ string_of_vdecl vd  ^ " in " ^
174             "{\n" ^ String.concat ", " (List.map string_of_expr elist) ^ "
    }\n" ^
175           ") " ^ string_of_stmt s
176   | While(e, s) -> "while (" ^ string_of_expr e ^ ") {\n" ^ string_of_stmt
      s ^ "\n}\n"
177   | Pass(s, e) -> "pass (" ^ s ^ ", " ^ string_of_expr e ^ ")\n"
178   | Break -> "break;\n"
179   | Continue -> "continue;\n"
180   | End -> "end();\n"
181
182 let string_of_basic_fdecl fdecl =
183   "func" ^ " " ^ string_of_vtype fdecl.ftype ^ " " ^
184   fdecl.fname ^ "(" ^
185         String.concat ", " (List.map string_of_vdecl fdecl.formals) ^ ")\n
    {\n" ^
186   String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") fdecl.
    locals) ^
187   String.concat "" (List.map string_of_stmt fdecl.body) ^
188   "}\n"
189
190 let string_of_assert_decl fdecl =
191   "assert " ^
192   fdecl.fname ^ "(" ^
193         String.concat ", " (List.map string_of_vdecl fdecl.formals) ^ ")\n
    {\n" ^
194   String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") fdecl.
    locals) ^
195   String.concat "" (List.map string_of_stmt fdecl.body) ^
196   "}\n"
197
198 let string_of_fdecl = function
199     BasicFunc(f) -> string_of_basic_fdecl f
200   | AssertFunc(f) -> string_of_assert_decl f
201
202 let string_of_gdecl gdecl =
203   "group " ^ gdecl.gname ^ "(" ^
204       (match gdecl.extends with
205           Some(par) -> string_of_field par ^
```

```
              (match gdecl.par_actuals with
                  Some(acts) ->
                    "(" ^ String.concat ", " (List.map string_of_expr acts
   ) ^ ")"
                | None -> "")
        | None -> "") ^ ")\n{\n" ^
  String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") gdecl.
    attributes) ^
  String.concat "" (List.map string_of_fdecl gdecl.methods) ^
  "};\n"

let string_of_setup (vars, funcs, groups) =
  "@setup {\n" ^
  String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") vars) ^
  String.concat "\n" (List.map string_of_fdecl funcs) ^
  String.concat "\n" (List.map string_of_gdecl groups) ^
  "}\n"

let string_of_turns (funcs) =
  "@turns {\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs) ^
  "}\n"

let string_of_program (s, t) =
  string_of_setup s ^ string_of_turns t
```

./c_files/headers/all_headers.h

```
#ifndef __ALL_HEADERS__
#define __ALL_HEADERS__
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <setjmp.h>
#include <string.h>
#include "sen_basic_type.h"
#include "sen_int.h"
#include "sen_object.h"
#include "sen_bool.h"
#include "sen_string.h"
#include "sen_array.h"
#include "sen_array_int.h"
#include "sen_board.h"
#include "sen_board_square.h"
#endif
```

./c_files/headers/sen_array.h

```
#include "sen_object.h"
#include "sen_basic_type.h"
#include "sen_int.h"

#ifndef SEN_ARRAY_H
#define SEN_ARRAY_H

```

```
#if !defined(ARRAY_SIZE)
    #define ARRAY_SIZE(x) (sizeof((x)) / sizeof((x)[0]))
#endif

struct Sen_array_vtable;
typedef struct Sen_array_vtable Sen_array_vtable;

struct Sen_array_class;
typedef struct Sen_array_class Sen_array_class;

struct Sen_array;
typedef struct Sen_array Sen_array;

struct Sen_array_vtable {
    void (*print) (Sen_object *);
    //Sen_array *(*construct) (Sen_object **, int);
    Sen_array *(*construct) (int);
    void (*destruct) (Sen_array *);
    Sen_array *(*copy) (Sen_array *);
    Sen_object *(*access) (Sen_array *, Sen_int *);
    Sen_array *(*concat) (Sen_array *, Sen_array *);
};

struct Sen_array_class {
    Sen_object_class *superp;
    Sen_array_vtable *tablep;
};

struct Sen_array {
    bool bound;
    Sen_array_class *classp;
    Sen_object *superp;
    Sen_object **arr;
    int len;
    char print_sep;
};

extern Sen_array_class Sen_array_class_;
extern Sen_array_vtable Sen_array_vtable_;

//Sen_array *construct_array (Sen_object **, int);
Sen_array *construct_array (int);
void destruct_array (Sen_array *);
Sen_array *copy_array (Sen_array *);
Sen_object *access_array (Sen_array *, Sen_int *);
Sen_array *concat_array (Sen_array *, Sen_array *);

#define CONCAT_ARRAY(x,y) ({                                                \
            __auto_type __temp__ = x;                                       \
            __temp__->classp->tablep->concat((Sen_array *)temp, (Sen_array
    *)y); \
        })

//#define CONSTRUCT_ARRAY(x) ((Sen_array *) construct_array(x))
```

```
#define CONSTRUCT_ARRAY(array, length) ({                                    \
            __auto_type input_arr = array;                                   \
            __auto_type __temp_arr__ = construct_array(length);              \
            __temp_arr__->len=length;                                        \
            for (int i=0; i<length; i++) {                                   \
                __auto_type __temp_elem__ = input_arr[i];                    \
                __temp_arr__->arr[i] = (Sen_object *) COPY(input_arr[i]);
           \
                __temp_arr__->bound = true;\
                if (!__temp_elem__->bound) {                                 \
                    DESTRUCT(__temp_elem__);                                 \
                }                                                            \
            }                                                                \
            __temp_arr__;                                                    \
        })
#endif
```

./c_files/headers/sen_array_int.h

```
#include "sen_object.h"
#include "sen_basic_type.h"
#include "sen_int.h"

#ifndef SEN_ARRAY_INT_H
#define SEN_ARRAY_INT_H

#if !defined(ARRAY_INT_SIZE)
    #define ARRAY_INT_SIZE(x) (sizeof((x)) / sizeof((x)[0]))
#endif

struct Sen_array_int_vtable;
typedef struct Sen_array_int_vtable Sen_array_int_vtable;

struct Sen_array_int_class;
typedef struct Sen_array_int_class Sen_array_int_class;

struct Sen_array_int;
typedef struct Sen_array_int Sen_array_int;

struct Sen_array_int_vtable {
    void (*print) (Sen_object *);
    //Sen_array_int *(*construct) (Sen_object **, int);
    Sen_array_int *(*construct) (int);
    void (*destruct) (Sen_array_int *);
    Sen_array_int *(*copy) (Sen_array_int *);
    Sen_int *(*access) (Sen_array_int *, Sen_int *);
    Sen_array_int *(*concat) (Sen_array_int *, Sen_array_int *);
};

struct Sen_array_int_class {
    Sen_object_class *superp;
    Sen_array_int_vtable *tablep;
};

struct Sen_array_int {
```

```
    bool bound;
    Sen_array_int_class *classp;
    Sen_object *superp;
    Sen_int **arr;
    int len;
    char print_sep;
};

extern Sen_array_int_class Sen_array_int_class_;
extern Sen_array_int_vtable Sen_array_int_vtable_;

//Sen_array_int *construct_array_int (Sen_object **, int);
Sen_array_int *construct_array_int (int);
void destruct_array_int (Sen_array_int *);
Sen_array_int *copy_array_int (Sen_array_int *);
Sen_int *access_array_int (Sen_array_int *, Sen_int *);
Sen_array_int *concat_array_int (Sen_array_int *, Sen_array_int *);

#define CONCAT_ARRAY_INT(x,y) ({\
    __auto_type __temp__ = x;\
    __temp__->classp->tablep->concat((Sen_array_int *)temp, (Sen_array_int
    *)y);\
        })

//#define CONSTRUCT_ARRAY_INT(x) ((Sen_array_int *) construct_array_int(x)
   )
#define CONSTRUCT_ARRAY_INT(array_int, length) ({
         \
            __auto_type input_arr = array_int;
     \
            __auto_type __temp_arr__ = construct_array_int(length);
     \
            __temp_arr__->len=length;                                    \
            for (int i=0; i<length; i++) {                               \
                __auto_type __temp_elem__ = input_arr[i];          \
                __auto_type __temp_cpy__ = COPY(input_arr[i]);     \
                __temp_arr__->arr[i] = __temp_cpy__;\
                if (!__temp_elem__->bound) {                             \
                    DESTRUCT(__temp_elem__);                             \
                }                                                        \
            }                                                            \
            __temp_arr__;                                                \
        })
#endif
```

./c_files/headers/sen_basic_type.h

```
#include "sen_object.h"

#ifndef SEN_BASIC_TYPE_H
#define SEN_BASIC_TYPE_H

struct Sen_basic_type_vtable;
typedef struct Sen_basic_type_vtable Sen_basic_type_vtable;

```

```
struct Sen_basic_type_class;
typedef struct Sen_basic_type_class Sen_basic_type_class;

struct Sen_basic_type;
typedef struct Sen_basic_type Sen_basic_type;

typedef enum {BOOL, INT, STR, UNK} Type;

struct Sen_basic_type_vtable {
    void (*print) (Sen_object *);
    Sen_basic_type *(*construct) (void *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_basic_type_class {
    Sen_object_class *superp;
    Sen_basic_type_vtable *tablep;
    Type type;
};

struct Sen_basic_type {
    bool bound;
    Sen_basic_type_class *classp;
    Sen_object *superp;
};

extern Sen_basic_type_class Sen_basic_type_class_;
extern Sen_basic_type_vtable Sen_basic_type_vtable_;

void * get_val_basic_type (Sen_basic_type *);
void * set_val_basic_type (Sen_basic_type *, void *);
Sen_basic_type * add_basic_type (Sen_basic_type *, Sen_basic_type *);

//#define ADD_BASIC_TYPE(x,y, target) x->classp->tablep->add((
   Sen_basic_type *)x, (Sen_basic_type *)y)
#define ADD_BASIC_TYPE(x,y) ({                  \
    __auto_type  __temp__ = x;\
    __temp__->classp->tablep->add((Sen_basic_type *)__temp__, (
   Sen_basic_type *)y);\
    })

#endif
```

./c_files/headers/sen_board.h

```
#include "sen_object.h"
#include "sen_basic_type.h"
#include "sen_int.h"
#include "sen_array.h"

#ifndef SEN_BOARD_H
#define SEN_BOARD_H

```

```
struct Sen_board_vtable;
typedef struct Sen_board_vtable Sen_board_vtable;

struct Sen_board_class;
typedef struct Sen_board_class Sen_board_class;

struct Sen_board;
typedef struct Sen_board Sen_board;

struct Sen_board_vtable {
    void (*print) (Sen_object *);
    //Sen_board *(*construct) (Sen_object **, int);
    Sen_board *(*construct) (int);
    void (*destruct) (Sen_board *);
    Sen_board *(*copy) (Sen_board *);
    int *(*index) (Sen_array *);
};

struct Sen_board_class {
    Sen_object_class *superp;
    Sen_board_vtable *tablep;
};

struct Sen_board {
    bool bound;
    Sen_board_class *classp;
    Sen_object *superp;
    Sen_array *data;
    int len;
    char print_sep;
};

extern Sen_board_class Sen_board_class_;
extern Sen_board_vtable Sen_board_vtable_;

//Sen_board *construct_board (Sen_object **, int);
Sen_board *construct_board (int);
int board_index (Sen_array *);

#define CONSTRUCT_BOARD(array) ({                                       \
            __auto_type input_arr = array;                              \
            __auto_type __temp_board__ = construct_board(input_arr->len);
   \
            __temp_board__->len=input_arr->len;                         \
            printf("OKAY\n");                                           \
            __temp_board__->data=COPY(input_arr);                       \
            if (!input_arr->bound) {                                    \
                DESTRUCT(input_arr);                                    \
            }                                                           \
            __temp_board__;                                             \
        })
#endif
```

./c_files/headers/sen_board_square.h

```
#include "sen_object.h"
#include "sen_basic_type.h"
#include "sen_int.h"
#include "sen_array.h"

#ifndef SEN_BOARD_SQUARE_H
#define SEN_BOARD_SQUARE_H

struct Sen_board_square_vtable;
typedef struct Sen_board_square_vtable Sen_board_square_vtable;

struct Sen_board_square_class;
typedef struct Sen_board_square_class Sen_board_square_class;

struct Sen_board_square;
typedef struct Sen_board_square Sen_board_square;

struct Sen_board_square_vtable {
    void (*print) (Sen_object *);
    //Sen_board_square *(*construct) (Sen_object **, int);
    Sen_board_square *(*construct) (int);
    void (*destruct) (Sen_board_square *);
    Sen_board_square *(*copy) (Sen_board_square *);
    int (*index) (Sen_array *, int);
};

struct Sen_board_square_class {
    Sen_object_class *superp;
    Sen_board_square_vtable *tablep;
};

struct Sen_board_square {
    bool bound;
    Sen_board_square_class *classp;
    Sen_object *superp;
    Sen_array *data;
    int len;
    char print_sep;
};

extern Sen_board_square_class Sen_board_square_class_;
extern Sen_board_square_vtable Sen_board_square_vtable_;

//Sen_board_square *construct_board_square (Sen_object **, int);
Sen_board_square *construct_board_square (int);
int board_square_index (Sen_array *);

#define CONSTRUCT_BOARD_SQUARE(array) ({
        \
            __auto_type input_arr = array;                                  \
            __auto_type __temp_board_square__ = construct_board_square(
   input_arr->len); \
            __temp_board_square__->len=input_arr->len;
        \
```

```
            printf("OKAY\n");\
            __temp_board_square__->data=COPY(input_arr);
        \
            if (!input_arr->bound) {                                    \
                DESTRUCT(input_arr);                                    \
            }                                                           \
            __temp_board_square__;
        \
        })
#endif
```

./c_files/headers/sen_board_square_tictac.h

```
#include "sen_object.h"
#include "sen_basic_type.h"
#include "sen_int.h"
#include "sen_array.h"

#ifndef SEN_BOARD_SQUARE_TICTAC_H
#define SEN_BOARD_SQUARE_TICTAC_H

struct Sen_board_square_tictac_vtable;
typedef struct Sen_board_square_tictac_vtable
   Sen_board_square_tictac_vtable;

struct Sen_board_square_tictac_class;
typedef struct Sen_board_square_tictac_class Sen_board_square_tictac_class
   ;

struct Sen_board_square_tictac;
typedef struct Sen_board_square_tictac Sen_board_square_tictac;

struct Sen_board_square_tictac_vtable {
    void (*print) (Sen_object *);
    //Sen_board_square_tictac *(*construct) (Sen_object **, int);
    Sen_board_square_tictac *(*construct) (int);
    void (*destruct) (Sen_board_square_tictac *);
    Sen_board_square_tictac *(*copy) (Sen_board_square_tictac *);
    int (*index) (Sen_array *, int);
};

struct Sen_board_square_tictac_class {
    Sen_object_class *superp;
    Sen_board_square_tictac_vtable *tablep;
};

struct Sen_board_square_tictac {
    bool bound;
    Sen_board_square_tictac_class *classp;
    Sen_object *superp;
    Sen_array *data;
    int len;
    char print_sep;
};

```

```
extern Sen_board_square_tictac_class Sen_board_square_tictac_class_;
extern Sen_board_square_tictac_vtable Sen_board_square_tictac_vtable_;

//Sen_board_square_tictac *construct_board_square_tictac (Sen_object **,
   int);
Sen_board_square_tictac *construct_board_square_tictac (int);
int board_square_tictac_index (Sen_array *);

#define CONSTRUCT_BOARD_SQUARE_TICTAC(array) ({                          \
            __auto_type input_arr = array;                               \
            __auto_type __temp_board_square_tictac__ =
   construct_board_square_tictac(input_arr->len); \
            __temp_board_square_tictac__->len=input_arr->len;            \
            printf("OKAY\n");                                            \
            __temp_board_square_tictac__->data=COPY(input_arr);          \
            if (!input_arr->bound) {                                     \
                DESTRUCT(input_arr);                                     \
            }                                                            \
            __temp_board_square_tictac__;                                \
        })
#endif
```

./c_files/headers/sen_bool.h

```
#include "sen_basic_type.h"

#ifndef SEN_BOOL_H
#define SEN_BOOL_H

struct Sen_bool_vtable;
typedef struct Sen_bool_vtable Sen_bool_vtable;

struct Sen_bool_class;
typedef struct Sen_bool_class Sen_bool_class;

struct Sen_bool;
typedef struct Sen_bool Sen_bool;

struct Sen_bool_vtable {
    void (*print) (Sen_object *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_bool *(*construct) (bool);
    void (*destruct) (Sen_bool *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_bool_class {
    Sen_basic_type_class *superp;
    Sen_bool_vtable *tablep;
    Type type;
};

struct Sen_bool {
    bool bound;
```

```
    Sen_bool_class *classp;
    Sen_basic_type *superp;
    bool val;
};

extern Sen_bool_class Sen_bool_class_;
extern Sen_bool_vtable Sen_bool_vtable_;

void print_bool (Sen_object *);
Sen_bool * construct_bool (bool);
void *get_val_bool (Sen_basic_type *);
void *set_val_bool (Sen_basic_type *, void *);

#define CONSTRUCT_BOOL(val) (Sen_bool*) construct_bool(val)

#endif
```

./c_files/headers/sen_int.h

```
#include "sen_basic_type.h"

#ifndef SEN_INT_H
#define SEN_INT_H

struct Sen_int_vtable;
typedef struct Sen_int_vtable Sen_int_vtable;

struct Sen_int_class;
typedef struct Sen_int_class Sen_int_class;

struct Sen_int;
typedef struct Sen_int Sen_int;

struct Sen_int_vtable {
    void (*print) (Sen_object *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_int *(*construct) (int);
    void (*destruct) (Sen_int *);
    Sen_int *(*copy) (Sen_int *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_int_class {
    Sen_basic_type_class *superp;
    Sen_int_vtable *tablep;
    Type type;
};

struct Sen_int {
    bool bound;
    Sen_int_class *classp;
    Sen_basic_type *superp;
    int val;
};
```

```

extern Sen_int_class Sen_int_class_;
extern Sen_int_vtable Sen_int_vtable_;

void print_int (Sen_object *);
Sen_int * construct_int (int);
void *get_val_int (Sen_basic_type *);
void *set_val_int (Sen_basic_type *, void *);

#define CONSTRUCT_INT(val) ((Sen_int*) construct_int(val))

#endif
```

./c_files/headers/sen_object.h

```
//#include "stdio.h"
//#include "stdlib.h"

#ifndef SEN_OBJECT_H
#define SEN_OBJECT_H

struct Sen_object_vtable;
typedef struct Sen_object_vtable Sen_object_vtable;

struct Sen_object_class;
typedef struct Sen_object_class Sen_object_class;

struct Sen_object;
typedef struct Sen_object Sen_object;

struct Sen_object_vtable {
    void (*print) (Sen_object *);
    Sen_object *(*construct) (void *);
    void (*destruct) (Sen_object *);
    Sen_object *(*copy) (Sen_object *);
};

//static Sen_object_vtable _Sen_object_vtable;

struct Sen_object_class {
    void * superp;
    Sen_object_vtable *tablep;
};

//static Sen_object_class _Sen_object_class;

struct Sen_object {
    bool bound;
    Sen_object_class *classp;
};

extern Sen_object_class Sen_object_class_;
extern Sen_object_vtable Sen_object_vtable_;

void print_object (Sen_object *);
```

```
Sen_object * construct_object (void *);
void destruct_object (Sen_object *);
Sen_object * copy_object (Sen_object *);

#define PRINT(self) {                                                   \
        typeof(self) __temp__ = self;                                   \
        __temp__-> classp-> tablep-> print(((Sen_object *)__temp__));  \
}

#define DESTRUCT(self) ({                                       \
            __auto_type __temp__ = self;                        \
            __temp__->classp->tablep->destruct(__temp__);   \
        })

#define COPY(self) ({                                                   \
            typeof(self) __temp__ = self;                               \
            (typeof (__temp__)) __temp__->classp->tablep->copy(__temp__);
   \
        })
#endif
```

./c_files/headers/sen_string.h

```
#include "sen_basic_type.h"

#ifndef SEN_STRING_H
#define SEN_STRING_H

struct Sen_string_vtable;
typedef struct Sen_string_vtable Sen_string_vtable;

struct Sen_string_class;
typedef struct Sen_string_class Sen_string_class;

struct Sen_string;
typedef struct Sen_string Sen_string;

struct Sen_string_vtable {
    void (*print) (Sen_object *);
    Sen_string *(*construct) (char *);
    void (*destruct) (Sen_string *);
    Sen_string *(*copy) (Sen_string *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_string_class {
    Sen_basic_type_class *superp;
    Sen_string_vtable *tablep;
    Type type;
};

struct Sen_string {
    bool bound;
```

```
    Sen_string_class *classp;
    Sen_basic_type *superp;
    char *val;
};

extern Sen_string_class Sen_string_class_;
extern Sen_string_vtable Sen_string_vtable_;

void print_string (Sen_object *);
Sen_string * construct_string (char *);
void destruct (Sen_string *);
Sen_string * copy_string  (Sen_string *);
void *get_val_string (Sen_basic_type *);
void *set_val_string (Sen_basic_type *, void *);


#define CONSTRUCT_STRING(val) ((Sen_string*) construct_string(val))

#endif
```

./c_files/headers/sen_tictac.h

```
struct GameBoard
{
  int length;
  int width;
  int *board;
  int game_step;
};
typedef struct GameBoard GameBoard;

//Functions
GameBoard* init_board( int length, int width);
void display(GameBoard);
int board_evaluate(GameBoard);
int valid_turn(GameBoard,int x,int y);
void turn(GameBoard* game_board);
void reset_game(GameBoard* game_board);
```

./c_files/main.c

```
#include "headers/all_headers.h"

int main() {
    __auto_type x = CONSTRUCT_INT(100);
    x->bound = true;
    __auto_type f = CONSTRUCT_INT(50);
    f->bound=true;
    printf("%d\n", x->val);
    PRINT(x);
    printf("\n");
    PRINT(f);
    printf("%d %d\n", x->bound, f->bound);
    {
        typeof(x) __temp__ = (typeof(x)) ADD_BASIC_TYPE(x, ((Sen_int *)
   ADD_BASIC_TYPE(x, f)));
```

```
        free(x);
        x = __temp__;
        x->bound=true;
    }
    printf("%d %d\n", x->bound, f->bound);
    x->bound=true;
    PRINT(x);
    printf("\n");
    printf("%d %d\n", x->bound, f->bound);
    DESTRUCT(x);
    DESTRUCT(f);
    __auto_type s = CONSTRUCT_STRING("tttestingggg ");
    s->bound=true;
    __auto_type ss = CONSTRUCT_STRING("hooray!!\n");
    ss->bound=true;
    //PRINT(arr_[0]);

    PRINT(((Sen_string *) ss));
    PRINT(((Sen_string *)ADD_BASIC_TYPE(s, ss)));
    PRINT(((Sen_string *)ADD_BASIC_TYPE(((Sen_string *)ADD_BASIC_TYPE(s,
ss)), ((Sen_string *)ADD_BASIC_TYPE(s, ss)))));
    DESTRUCT(((Sen_string *) s));
    DESTRUCT(((Sen_string *) ss));
    //Sen_array *arr = (Sen_array_vtable_.construct(arr_, ARRAY_SIZE(arr_)
));
    //Sen_array *arr = (Sen_array_vtable_.construct((Sen_int*[]){
CONSTRUCT_INT(100), CONSTRUCT_INT(50)}, 2));
    Sen_array *arr = CONSTRUCT_ARRAY(((Sen_int*[]){CONSTRUCT_INT(1900),
CONSTRUCT_INT(50)}), 2);
    arr->bound = true;
    //Sen_array *arr = CONSTRUCT_ARRAY(arr_, 2);
    __auto_type xx = CONSTRUCT_INT(123);
    xx->bound=true;
    __auto_type yy = COPY(xx);
    PRINT(yy);
    PRINT(xx);
    DESTRUCT(xx);
    printf("%d %d asdsad\n", ((Sen_int *)((arr->arr)[0]))->val, arr->len);
    printf("OKAY\n");
    __auto_type xxx = arr->arr[1];
    PRINT(((Sen_int *)xxx));
    printf("\nOKAY\n");
    PRINT(xxx);
    printf("\nOKAY\n");

    printf("OKAY\n");
    PRINT((arr->arr)[1]);
    printf("OKAY\n");
    //DESTRUCT(xxx);
    Sen_board *board = CONSTRUCT_BOARD(arr);
    printf("%d %d\n", ((Sen_int *)((arr->arr)[0]))->val, arr->len);
    //DESTRUCT(arr);
    printf("OKAYFINALS\n");

```

```
    //DESTRUCT(board);
    //printf("OKAY\n");
    return 0;
}
```

./c_files/Makefile

```
CC=gcc
CFLAGS= -Wall -g

INC= -I ./headers

SRC=$(wildcard *.c)
SRC:=$(filter-out tictactoe.c, $(SRC))

all: main

main: $(SRC)
    $(CC) -o out $(CFLAGS) $(INC) $(SRC)

clean:
    rm -f *.o out
```

./c_files/sen_array.c

```
#include "headers/all_headers.h"

/*
Sen_array *construct_array(Sen_object *val[], int len) {
    Sen_array *ret = malloc (sizeof(Sen_array));
    ret->len = len;
    printf("%d %d\n", (int)ret->len, (int)sizeof(Sen_object *));
    ret->arr = malloc(sizeof(typeof (val[0])) * (len));
    for (int i=0; i<len; i++) {
        (ret->arr)[i] = COPY(((typeof (val[i])) val[i]));
        (ret->arr)[i]->bound=true;
        if (!val[i]->bound) {
            DESTRUCT(val[i]);
        }
    }
    ret->bound=false;
    return ret;
}
*/

Sen_array *construct_array(int len) {
    Sen_array *ret = malloc(sizeof(Sen_array));
    ret->len = len;
    ret->classp = &Sen_array_class_;
    ret->bound = false;
    ret->arr = malloc(sizeof(Sen_object *) * len);
    ret->print_sep=' ';
    return ret;
}

void destruct_array(Sen_array *self) {
```

```
    for (int i=0; i<self->len; i++) {
        free(self->arr[i]);
    }
    free(self);
}

Sen_array *copy_array(Sen_array *other) {
    Sen_array *ret = construct_array(other->len);
    for (int i=0; i<ret->len; i++) {
        //printf("%d %d\n", other->len, ((Sen_int *)(other->arr[i]))->val)
   ;
        //printf("OKAY1\n");
        ret->arr[i] = COPY(((other->arr)[i]));
        ret->arr[i]->bound=true;
        //printf("OKAY2\n");
    }
    //printf("OKAY1\n");
    return ret;
}

Sen_object *access_array(Sen_array *self, Sen_int *index) {
    return self->arr[index->val];
}

Sen_array *add_array(Sen_array *x, Sen_array *y) {
    // Need to check that types are the same
    Sen_array *ret = malloc(sizeof(ret));
    *ret=*x;
    return ret;
}

Sen_array_vtable Sen_array_vtable_ = {
    print_object,
    construct_array,
    destruct_array,
    copy_array,
    access_array,
    add_array
};

Sen_array_class Sen_array_class_ = {
    &Sen_object_class_,
    &Sen_array_vtable_,
};
```

./c_files/sen_array_int.c

```
#include "headers/all_headers.h"

/*
Sen_array_int *construct_array_int(Sen_object *val[], int len) {
    Sen_array_int *ret = malloc (sizeof(Sen_array_int));
    ret->len = len;
    printf("%d %d\n", (int)ret->len, (int)sizeof(Sen_object *));
    ret->arr = malloc(sizeof(typeof (val[0])) * (len));
```

```
    for (int i=0; i<len; i++) {
        (ret->arr)[i] = COPY(((typeof (val[i])) val[i]));
        (ret->arr)[i]->bound=true;
        if (!val[i]->bound) {
            DESTRUCT(val[i]);
        }
    }
    ret->bound=false;
    return ret;
}
*/

Sen_array_int *construct_array_int(int len) {
    Sen_array_int *ret = malloc(sizeof(Sen_array_int));
    ret->len = len;
    ret->classp = &Sen_array_int_class_;
    ret->bound = false;
    ret->arr = malloc(sizeof(Sen_int *) * len);
    ret->print_sep=' ';
    return ret;
}

void destruct_array_int(Sen_array_int *self) {
    for (int i=0; i<self->len; i++) {
        free(self->arr[i]);
    }
    free(self);
}

Sen_array_int *copy_array_int(Sen_array_int *other) {
    Sen_array_int *ret = construct_array_int(other->len);
    for (int i=0; i<ret->len; i++) {
        printf("%d %d\n", other->len, ((Sen_int *)(other->arr[i]))->val);
        printf("OKAY1\n");
        ret->arr[i] = COPY(((other->arr)[i]));
        printf("OKAY2\n");
    }
    printf("OKAY1\n");
    return ret;
}

Sen_int *access_array_int(Sen_array_int *self, Sen_int *index) {
    return self->arr[index->val];
}

Sen_array_int *add_array_int(Sen_array_int *x, Sen_array_int *y) {
    // Need to check that types are the same
    Sen_array_int *ret = malloc(sizeof(ret));
    *ret=*x;
    return ret;
}

Sen_array_int_vtable Sen_array_int_vtable_ = {
    print_object,
```

```
    construct_array_int,
    destruct_array_int,
    copy_array_int,
    access_array_int,
    add_array_int
};

Sen_array_int_class Sen_array_int_class_ = {
    &Sen_object_class_,
    &Sen_array_int_vtable_,
};
```

./c_files/sen_basic_type.c

```
#include "headers/all_headers.h"

void *get_val_basic_type(Sen_basic_type *self) {
    int *ret = malloc(sizeof *ret);
    *ret=((Sen_int *) self)->val;
    return ret;
}

void *set_val_basic_type(Sen_basic_type *self, void *val) {

    //self->val=val;
    return val;
}

Sen_basic_type *construct_basic_type(void *val) {
    Sen_basic_type *ret = malloc(sizeof(ret));
    return ret;
}

Sen_basic_type *add_basic_type(Sen_basic_type *x, Sen_basic_type *y) {
    // Need to check that types are the same
    Sen_basic_type *ret = malloc(sizeof(ret));
    *ret=*x;
    return ret;
}

Sen_basic_type_vtable Sen_basic_type_vtable_ = {
    print_object,
    construct_basic_type,
    get_val_basic_type,
    set_val_basic_type,
    add_basic_type
};

Sen_basic_type_class Sen_basic_type_class_ = {
    &Sen_object_class_,
    &Sen_basic_type_vtable_,
    UNK
};
```

./c_files/sen_board.c

```c
#include "headers/all_headers.h"

/*
Sen_board *construct_board(Sen_object *val[], int len) {
    Sen_board *ret = malloc (sizeof(Sen_board));
    ret->len = len;
    printf("%d %d\n", (int)ret->len, (int)sizeof(Sen_object *));
    ret->arr = malloc(sizeof(typeof (val[0])) * (len));
    for (int i=0; i<len; i++) {
        (ret->arr)[i] = COPY(((typeof (val[i])) val[i]));
        (ret->arr)[i]->bound=true;
        if (!val[i]->bound) {
            DESTRUCT(val[i]);
        }
    }
    ret->bound=false;
    return ret;
}
*/

Sen_board *construct_board(int len) {
    Sen_board *ret = malloc(sizeof(Sen_board));
    ret->len   = len;
    ret->bound = false;
    ret->data  = construct_array(len);
    ret->print_sep = ' ';
    printf("hi\n");
    return ret;
}

void destruct_board(Sen_board *self) {
    DESTRUCT(self->data);
    free(self);
}

Sen_board *copy_board(Sen_board *other) {
    Sen_board *ret = malloc(sizeof(Sen_board));
    ret->len = other->len;
    ret->data = COPY(other->data);
    return ret;
}

Sen_board_vtable Sen_board_vtable_ = {
    print_object,
    construct_board,
    destruct_board,
};

Sen_board_class Sen_board_class_ = {
    &Sen_object_class_,
    &Sen_board_vtable_,
};
```

./c_files/sen_board_square.c

```c
#include "headers/all_headers.h"

/*
Sen_board_square *construct_board_square(Sen_object *val[], int len) {
    Sen_board_square *ret = malloc (sizeof(Sen_board_square));
    ret->len = len
    printf("%d %d\n", (int)ret->len, (int)sizeof(Sen_object *));
    ret->arr = malloc(sizeof(typeof (val[0])) * (len));
    for (int i=0; i<len; i++) {
        (ret->arr)[i] = COPY(((typeof (val[i])) val[i]));
        (ret->arr)[i]->bound=true;
        if (!val[i]->bound) {
            DESTRUCT(val[i]);
        }
    }
    ret->bound=false;
    return ret;
}
*/

Sen_board_square *construct_board_square(int len) {
    Sen_board_square *ret = malloc(sizeof(Sen_board_square));
    ret->classp = &Sen_board_square_class_;
    ret->len   = len;
    ret->bound = false;
    ret->data  = construct_array(len);
    ret->print_sep = ' ';
    printf("hi\n");
    return ret;
}

void destruct_board_square(Sen_board_square *self) {
    DESTRUCT(self->data);
    free(self);
}


Sen_board_square *copy_board_square(Sen_board_square *other) {
    Sen_board_square *ret = malloc(sizeof(Sen_board_square));
    ret->len = other->len;
    ret->data = COPY(other->data);
    return ret;
}

int index_board_square(Sen_array *coord, int len) {
    int x = ((Sen_int *)coord->arr[0])->val;
    int y = ((Sen_int *)coord->arr[1])->val;
    return x + y % len;
}

Sen_board_square_vtable Sen_board_square_vtable_ = {
    print_object,
    construct_board_square,
    destruct_board_square,
```

```
    copy_board_square,
    index_board_square
};

Sen_board_square_class Sen_board_square_class_ = {
    &Sen_object_class_,
    &Sen_board_square_vtable_,
};
```

./c_files/sen_bool.c

```
#include "headers/all_headers.h"

Sen_bool *construct_bool(bool val) {
    Sen_bool *ret = malloc(sizeof(Sen_bool));
    ret->classp = &Sen_bool_class_;
    ret->val = val;
    ret->bound = false;
    return ret;
}

void destruct_bool(Sen_bool *self) {
    free(self);
}

void print_bool(Sen_object *self) {
    if (((Sen_bool*) self)->val==true) {
        printf("True");
    } else {
        printf("False");
    }
    if (!((Sen_bool *)self)->bound) {
        free(self);
    }
}

void *get_val_bool(Sen_basic_type *self) {
    bool *ret = malloc(sizeof *ret);
    *ret=((Sen_bool *) self)->val;
    if (!((Sen_bool *)self)->bound) {
        free(self);
    }
    return ret;
}

void *set_val_bool(Sen_basic_type *self, void *val) {
    ((Sen_bool *)self)->val=*(bool*)val;
    if (!((Sen_bool *)self)->bound) {
        free(self);
    }
    return val;
}

Sen_basic_type *add_bool(Sen_basic_type *x, Sen_basic_type *y) {
    // Need to check types for safety
```

```
    Sen_bool *ret = construct_bool(((Sen_bool *)x)->val);
    ret->val = ret->val != ((Sen_bool *) y)->val;
    if (!x->bound) {
        free(x);
    }
    if (!y->bound) {
        free(y);
    }
    return (Sen_basic_type *) ret;
}

Sen_bool_vtable Sen_bool_vtable_ = {
    print_bool,
    get_val_bool,
    set_val_bool,
    construct_bool,
    destruct_bool,
    add_bool
};

//Sen_basic_type_class temp; /* NEED TO FIX */
Sen_bool_class Sen_bool_class_ = {
    &Sen_basic_type_class_,
    &Sen_bool_vtable_,
    BOOL
};
```

./c_files/sen_int.c

```
#include "headers/all_headers.h"

Sen_int *construct_int(int val) {
    Sen_int *ret = malloc(sizeof(Sen_int));
    ret->classp = &Sen_int_class_;
    ret->val = val;
    ret->bound = false;
    return ret;
}

void destruct_int(Sen_int *self) {
    free(self);
}

Sen_int *copy_int(Sen_int *self) {
    Sen_int *ret = construct_int(self->val);
    return ret;
}

void print_int(Sen_object *self) {
    printf("%d", ((Sen_int *) self)->val);
    if (!((Sen_int *)self)->bound) {
        free(self);
    }
}

```

```
void *get_val_int(Sen_basic_type *self) {
    int *ret = malloc(sizeof *ret);
    *ret=((Sen_int *) self)->val;
    if (!self->bound) {
        free(self);
    }
    return ret;
}

void *set_val_int(Sen_basic_type *self, void *val) {
    ((Sen_int *)self)->val=*(int*)val;
    if (!self->bound) {
        free(self);
    }
    return val;
}

Sen_basic_type *add_int(Sen_basic_type *x, Sen_basic_type *y) {
    // Need to check types for safety
    Sen_int *ret = construct_int(((Sen_int *)x)->val);
    ret->val+=((Sen_int *) y)->val;
    if (!x->bound) {
        free(x);
    }
    if (!y->bound) {
        free(y);
    }
    return (Sen_basic_type *) ret;
}

Sen_int_vtable Sen_int_vtable_ = {
    print_int,
    get_val_int,
    set_val_int,
    construct_int,
    destruct_int,
    copy_int,
    add_int
};

//Sen_basic_type_class temp; /* NEED TO FIX */
Sen_int_class Sen_int_class_ = {
    &Sen_basic_type_class_,
    &Sen_int_vtable_,
    INT
};
```

./c_files/sen_object.c

```
#include "headers/all_headers.h"

void print_object(Sen_object *self) {

}

```

```
Sen_object *construct_object(void *val) {
    Sen_object *ret = malloc (sizeof (Sen_object *));
    return ret;
}

void destruct_object(Sen_object *val) {
    free(val);
}

Sen_object *copy_object(Sen_object *self) {
    Sen_object *ret=malloc(sizeof(Sen_object *));
    ret->classp=&Sen_object_class_;
    return ret;
}

Sen_object_vtable Sen_object_vtable_ = {
    print_object,
    construct_object,
    destruct_object,
    copy_object
};

Sen_object_class Sen_object_class_ = {
    NULL,
    &Sen_object_vtable_
};
```

./c_files/sen_string.c

```
#include "headers/all_headers.h"

Sen_string *construct_string(char *val) {
    int l = strlen(val);
    Sen_string *ret = malloc(sizeof(Sen_string));
    ret->classp = &Sen_string_class_;
    ret->val = (char *)malloc(l+1);
    strncpy(ret->val, val, l+1);
    ret->bound = false;
    return ret;
}

void destruct_string(Sen_string *self) {
    free(self->val);
    self->val = NULL;
    free(self);
    self=NULL;
}


Sen_string *copy_string(Sen_string *other) {
    return construct_string(other->val);
}

void print_string(Sen_object *self) {
    printf("%s", ((Sen_string *) self)->val);
```

```
    if (!self->bound) {
        destruct_string((Sen_string *) self);
    }
}

void *get_val_string(Sen_basic_type *self) {
    char **ret = malloc(sizeof *ret);
    *ret=((Sen_string *) self)->val;
    if (!self->bound) {
        destruct_string((Sen_string *) self);
    }
    return ret;
}

void *set_val_string(Sen_basic_type *self, void *val) {
    ((Sen_string *)self)->val=*(char **)val;
    if (!self->bound) {
        destruct_string((Sen_string *) self);
    }
    return val;
}

Sen_basic_type *add_string(Sen_basic_type *x, Sen_basic_type *y) {
    // Need to check types for safety
    char *new_string = malloc(strlen(((Sen_string*)x)->val) + strlen(((
   Sen_string*)y)->val) + 1);
    strcpy(new_string, ((Sen_string*) x)->val);
    strcat(new_string, ((Sen_string*) y)->val);
    Sen_string *ret = construct_string(new_string);
    free(new_string);
    if (!x->bound) {
        destruct_string((Sen_string*) x);
    }
    if (!y->bound) {
        destruct_string((Sen_string*) y);
    }
    return (Sen_basic_type *) ret;
}

Sen_string_vtable Sen_string_vtable_ = {
    print_string,
    construct_string,
    destruct_string,
    copy_string,
    get_val_string,
    set_val_string,
    add_string
};

//Sen_basic_type_class temp; /* NEED TO FIX */
Sen_string_class Sen_string_class_ = {
    &Sen_basic_type_class_,
    &Sen_string_vtable_,
    STR
```

```
};
```

./c_files/temp

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 1 "<command-line>" 2
# 1 "main.c"
# 1 "headers/all_headers.h" 1


# 1 "/usr/include/stdlib.h" 1 3 4
# 24 "/usr/include/stdlib.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 374 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
# 385 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 386 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
# 375 "/usr/include/features.h" 2 3 4
# 398 "/usr/include/features.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
# 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
# 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
# 399 "/usr/include/features.h" 2 3 4
# 25 "/usr/include/stdlib.h" 2 3 4









# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 216 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4

# 216 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4
typedef long unsigned int size_t;
# 328 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4
typedef int wchar_t;
# 33 "/usr/include/stdlib.h" 2 3 4









# 1 "/usr/include/x86_64-linux-gnu/bits/waitflags.h" 1 3 4
# 50 "/usr/include/x86_64-linux-gnu/bits/waitflags.h" 3 4
typedef enum
```

```
{
  P_ALL,
  P_PID,
  P_PGID
} idtype_t;
# 42 "/usr/include/stdlib.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 1 3 4
# 64 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 3 4
# 1 "/usr/include/endian.h" 1 3 4
# 36 "/usr/include/endian.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/endian.h" 1 3 4
# 37 "/usr/include/endian.h" 2 3 4
# 60 "/usr/include/endian.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 28 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;


typedef signed char __int8_t;
typedef unsigned char __uint8_t;
typedef signed short int __int16_t;
typedef unsigned short int __uint16_t;
typedef signed int __int32_t;
typedef unsigned int __uint32_t;

typedef signed long int __int64_t;
typedef unsigned long int __uint64_t;









typedef long int __quad_t;
typedef unsigned long int __u_quad_t;
# 121 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/typesizes.h" 1 3 4
# 122 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned long int __dev_t;
typedef unsigned int __uid_t;
typedef unsigned int __gid_t;
typedef unsigned long int __ino_t;
```

```
106 typedef unsigned long int __ino64_t;
107 typedef unsigned int __mode_t;
108 typedef unsigned long int __nlink_t;
109 typedef long int __off_t;
110 typedef long int __off64_t;
111 typedef int __pid_t;
112 typedef struct { int __val[2]; } __fsid_t;
113 typedef long int __clock_t;
114 typedef unsigned long int __rlim_t;
115 typedef unsigned long int __rlim64_t;
116 typedef unsigned int __id_t;
117 typedef long int __time_t;
118 typedef unsigned int __useconds_t;
119 typedef long int __suseconds_t;
120
121 typedef int __daddr_t;
122 typedef int __key_t;
123
124
125 typedef int __clockid_t;
126
127
128 typedef void * __timer_t;
129
130
131 typedef long int __blksize_t;
132
133
134
135
136 typedef long int __blkcnt_t;
137 typedef long int __blkcnt64_t;
138
139
140 typedef unsigned long int __fsblkcnt_t;
141 typedef unsigned long int __fsblkcnt64_t;
142
143
144 typedef unsigned long int __fsfilcnt_t;
145 typedef unsigned long int __fsfilcnt64_t;
146
147
148 typedef long int __fsword_t;
149
150 typedef long int __ssize_t;
151
152
153 typedef long int __syscall_slong_t;
154
155 typedef unsigned long int __syscall_ulong_t;
156
157
158
159 typedef __off64_t __loff_t;
```

```
typedef __quad_t *__qaddr_t;
typedef char *__caddr_t;


typedef long int __intptr_t;


typedef unsigned int __socklen_t;
# 28 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 29 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4









# 1 "/usr/include/x86_64-linux-gnu/bits/byteswap-16.h" 1 3 4
# 36 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4
# 44 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
static __inline unsigned int
__bswap_32 (unsigned int __bsx)
{
  return __builtin_bswap32 (__bsx);
}
# 108 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
static __inline __uint64_t
__bswap_64 (__uint64_t __bsx)
{
  return __builtin_bswap64 (__bsx);
}
# 61 "/usr/include/endian.h" 2 3 4
# 65 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 2 3 4

union wait
  {
    int w_status;
    struct
      {

 unsigned int __w_termsig:7;
 unsigned int __w_coredump:1;
 unsigned int __w_retcode:8;
 unsigned int:16;









      } __wait_terminated;
    struct
      {
```

```
214
215  unsigned int __w_stopval:8;
216  unsigned int __w_stopsig:8;
217  unsigned int:16;
218
219
220
221
222
223
224      } __wait_stopped;
225    };
226 # 43 "/usr/include/stdlib.h" 2 3 4
227 # 67 "/usr/include/stdlib.h" 3 4
228 typedef union
229    {
230      union wait *__uptr;
231      int *__iptr;
232    } __WAIT_STATUS __attribute__ ((__transparent_union__));
233 # 95 "/usr/include/stdlib.h" 3 4
234
235
236 typedef struct
237    {
238      int quot;
239      int rem;
240    } div_t;
241
242
243
244 typedef struct
245    {
246      long int quot;
247      long int rem;
248    } ldiv_t;
249
250
251
252
253
254
255
256 __extension__ typedef struct
257    {
258      long long int quot;
259      long long int rem;
260    } lldiv_t;
261
262
263 # 139 "/usr/include/stdlib.h" 3 4
264 extern size_t __ctype_get_mb_cur_max (void) __attribute__ ((__nothrow__ ,
       __leaf__)) ;
265
266
```

```
267
268
269 extern double atof (const char *__nptr)
270      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
271
272 extern int atoi (const char *__nptr)
273      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
274
275 extern long int atol (const char *__nptr)
276      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
277
278
279
280
281
282 __extension__ extern long long int atoll (const char *__nptr)
283      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
284
285
286
287
288
289 extern double strtod (const char *__restrict __nptr,
290         char **__restrict __endptr)
291      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
292
293
294
295
296
297 extern float strtof (const char *__restrict __nptr,
298        char **__restrict __endptr) __attribute__ ((__nothrow__ , __leaf__)
    ) __attribute__ ((__nonnull__ (1)));
299
300 extern long double strtold (const char *__restrict __nptr,
301        char **__restrict __endptr)
302      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
303
304
305
306
307
308 extern long int strtol (const char *__restrict __nptr,
309    char **__restrict __endptr, int __base)
310      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
311
312 extern unsigned long int strtoul (const char *__restrict __nptr,
```

```
313       char **__restrict __endptr, int __base)
314     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
315
316
317
318
319 __extension__
320 extern long long int strtoq (const char *__restrict __nptr,
321         char **__restrict __endptr, int __base)
322     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
323
324 __extension__
325 extern unsigned long long int strtouq (const char *__restrict __nptr,
326           char **__restrict __endptr, int __base)
327     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
328
329
330
331
332
333 __extension__
334 extern long long int strtoll (const char *__restrict __nptr,
335         char **__restrict __endptr, int __base)
336     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
337
338 __extension__
339 extern unsigned long long int strtoull (const char *__restrict __nptr,
340     char **__restrict __endptr, int __base)
341     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
342
343 # 305 "/usr/include/stdlib.h" 3 4
344 extern char *l64a (long int __n) __attribute__ ((__nothrow__ , __leaf__))
    ;
345
346
347 extern long int a64l (const char *__s)
348     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
349
350
351
352
353 # 1 "/usr/include/x86_64-linux-gnu/sys/types.h" 1 3 4
354 # 27 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
355
356
357
358
359
```

```
360
361 typedef __u_char u_char;
362 typedef __u_short u_short;
363 typedef __u_int u_int;
364 typedef __u_long u_long;
365 typedef __quad_t quad_t;
366 typedef __u_quad_t u_quad_t;
367 typedef __fsid_t fsid_t;
368
369
370
371
372 typedef __loff_t loff_t;
373
374
375
376 typedef __ino_t ino_t;
377 # 60 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
378 typedef __dev_t dev_t;
379
380
381
382
383 typedef __gid_t gid_t;
384
385
386
387
388 typedef __mode_t mode_t;
389
390
391
392
393 typedef __nlink_t nlink_t;
394
395
396
397
398 typedef __uid_t uid_t;
399
400
401
402
403
404 typedef __off_t off_t;
405 # 98 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
406 typedef __pid_t pid_t;
407
408
409
410
411
412 typedef __id_t id_t;
413
```

```
414
415
416
417 typedef __ssize_t ssize_t;
418
419
420
421
422
423 typedef __daddr_t daddr_t;
424 typedef __caddr_t caddr_t;
425
426
427
428
429
430 typedef __key_t key_t;
431 # 132 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
432 # 1 "/usr/include/time.h" 1 3 4
433 # 57 "/usr/include/time.h" 3 4
434
435
436 typedef __clock_t clock_t;
437
438
439
440 # 73 "/usr/include/time.h" 3 4
441
442
443 typedef __time_t time_t;
444
445
446
447 # 91 "/usr/include/time.h" 3 4
448 typedef __clockid_t clockid_t;
449 # 103 "/usr/include/time.h" 3 4
450 typedef __timer_t timer_t;
451 # 133 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
452 # 146 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
453 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
454 # 147 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
455
456
457
458 typedef unsigned long int ulong;
459 typedef unsigned short int ushort;
460 typedef unsigned int uint;
461 # 194 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
462 typedef int int8_t __attribute__ ((__mode__ (__QI__)));
463 typedef int int16_t __attribute__ ((__mode__ (__HI__)));
464 typedef int int32_t __attribute__ ((__mode__ (__SI__)));
465 typedef int int64_t __attribute__ ((__mode__ (__DI__)));
466
467
```

```
typedef unsigned int u_int8_t __attribute__ ((__mode__ (__QI__)));
typedef unsigned int u_int16_t __attribute__ ((__mode__ (__HI__)));
typedef unsigned int u_int32_t __attribute__ ((__mode__ (__SI__)));
typedef unsigned int u_int64_t __attribute__ ((__mode__ (__DI__)));

typedef int register_t __attribute__ ((__mode__ (__word__)));
# 219 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/select.h" 1 3 4
# 30 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/select.h" 1 3 4
# 22 "/usr/include/x86_64-linux-gnu/bits/select.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 23 "/usr/include/x86_64-linux-gnu/bits/select.h" 2 3 4
# 31 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4


# 1 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 1 3 4
# 22 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 3 4
typedef int __sig_atomic_t;




typedef struct
  {
    unsigned long int __val[(1024 / (8 * sizeof (unsigned long int)))];
  } __sigset_t;
# 34 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4



typedef __sigset_t sigset_t;




# 1 "/usr/include/time.h" 1 3 4
# 120 "/usr/include/time.h" 3 4
struct timespec
  {
    __time_t tv_sec;
    __syscall_slong_t tv_nsec;
  };
# 44 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4

# 1 "/usr/include/x86_64-linux-gnu/bits/time.h" 1 3 4
# 30 "/usr/include/x86_64-linux-gnu/bits/time.h" 3 4
struct timeval
  {
    __time_t tv_sec;
    __suseconds_t tv_usec;
  };
# 46 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4
```

```
522
523
524 typedef __suseconds_t suseconds_t;
525
526
527
528
529
530 typedef long int __fd_mask;
531 # 64 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
532 typedef struct
533   {
534
535
536
537
538
539
540     __fd_mask __fds_bits[1024 / (8 * (int) sizeof (__fd_mask))];
541
542
543   } fd_set;
544
545
546
547
548
549
550 typedef __fd_mask fd_mask;
551 # 96 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
552
553 # 106 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
554 extern int select (int __nfds, fd_set *__restrict __readfds,
555      fd_set *__restrict __writefds,
556      fd_set *__restrict __exceptfds,
557      struct timeval *__restrict __timeout);
558 # 118 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
559 extern int pselect (int __nfds, fd_set *__restrict __readfds,
560       fd_set *__restrict __writefds,
561       fd_set *__restrict __exceptfds,
562       const struct timespec *__restrict __timeout,
563       const __sigset_t *__restrict __sigmask);
564 # 131 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
565
566 # 220 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
567
568
569 # 1 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 1 3 4
570 # 24 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 3 4
571
572
573 __extension__
574 extern unsigned int gnu_dev_major (unsigned long long int __dev)
575      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
```

```
__extension__
extern unsigned int gnu_dev_minor (unsigned long long int __dev)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
__extension__
extern unsigned long long int gnu_dev_makedev (unsigned int __major,
            unsigned int __minor)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
# 58 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 3 4

# 223 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4




typedef __blksize_t blksize_t;




typedef __blkcnt_t blkcnt_t;


typedef __fsblkcnt_t fsblkcnt_t;


typedef __fsfilcnt_t fsfilcnt_t;
# 270 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 1 3 4
# 21 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 22 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 2 3 4
# 60 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
typedef unsigned long int pthread_t;


union pthread_attr_t
{
  char __size[56];
  long int __align;
};

typedef union pthread_attr_t pthread_attr_t;




typedef struct __pthread_internal_list
{
```

```
  struct __pthread_internal_list *__prev;
  struct __pthread_internal_list *__next;
} __pthread_list_t;
# 90 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
typedef union
{
  struct __pthread_mutex_s
  {
    int __lock;
    unsigned int __count;
    int __owner;

    unsigned int __nusers;



    int __kind;

    short __spins;
    short __elision;
    __pthread_list_t __list;
# 124 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
  } __data;
  char __size[40];
  long int __align;
} pthread_mutex_t;

typedef union
{
  char __size[4];
  int __align;
} pthread_mutexattr_t;




typedef union
{
  struct
  {
    int __lock;
    unsigned int __futex;
    __extension__ unsigned long long int __total_seq;
    __extension__ unsigned long long int __wakeup_seq;
    __extension__ unsigned long long int __woken_seq;
    void *__mutex;
    unsigned int __nwaiters;
    unsigned int __broadcast_seq;
  } __data;
  char __size[48];
  __extension__ long long int __align;
} pthread_cond_t;

typedef union
```

```
{
  char __size[4];
  int __align;
} pthread_condattr_t;



typedef unsigned int pthread_key_t;



typedef int pthread_once_t;






typedef union
{

  struct
  {
    int __lock;
    unsigned int __nr_readers;
    unsigned int __readers_wakeup;
    unsigned int __writer_wakeup;
    unsigned int __nr_readers_queued;
    unsigned int __nr_writers_queued;
    int __writer;
    int __shared;
    unsigned long int __pad1;
    unsigned long int __pad2;


    unsigned int __flags;

  } __data;
# 211 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
  char __size[56];
  long int __align;
} pthread_rwlock_t;

typedef union
{
  char __size[8];
  long int __align;
} pthread_rwlockattr_t;






typedef volatile int pthread_spinlock_t;

```

```
738
739
740
741 typedef union
742 {
743   char __size[32];
744   long int __align;
745 } pthread_barrier_t;
746
747 typedef union
748 {
749   char __size[4];
750   int __align;
751 } pthread_barrierattr_t;
752 # 271 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
753
754
755
756 # 315 "/usr/include/stdlib.h" 2 3 4
757
758
759
760
761
762
763 extern long int random (void) __attribute__ ((__nothrow__ , __leaf__));
764
765
766 extern void srandom (unsigned int __seed) __attribute__ ((__nothrow__ ,
       __leaf__));
767
768
769
770
771
772 extern char *initstate (unsigned int __seed, char *__statebuf,
773    size_t __statelen) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__nonnull__ (2)));
774
775
776
777 extern char *setstate (char *__statebuf) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
778
779
780
781
782
783
784
785 struct random_data
786   {
787     int32_t *fptr;
788     int32_t *rptr;
```

```
    int32_t *state;
    int rand_type;
    int rand_deg;
    int rand_sep;
    int32_t *end_ptr;
  };

extern int random_r (struct random_data *__restrict __buf,
       int32_t *__restrict __result) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (1, 2)));

extern int srandom_r (unsigned int __seed, struct random_data *__buf)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (2)));

extern int initstate_r (unsigned int __seed, char *__restrict __statebuf,
   size_t __statelen,
   struct random_data *__restrict __buf)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (2, 4)));

extern int setstate_r (char *__restrict __statebuf,
         struct random_data *__restrict __buf)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));









extern int rand (void) __attribute__ ((__nothrow__ , __leaf__));

extern void srand (unsigned int __seed) __attribute__ ((__nothrow__ ,
   __leaf__));




extern int rand_r (unsigned int *__seed) __attribute__ ((__nothrow__ ,
   __leaf__));









extern double drand48 (void) __attribute__ ((__nothrow__ , __leaf__));
extern double erand48 (unsigned short int __xsubi[3]) __attribute__ ((
   __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));


extern long int lrand48 (void) __attribute__ ((__nothrow__ , __leaf__));
```

```
836 extern long int nrand48 (unsigned short int __xsubi[3])
837      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
838
839
840 extern long int mrand48 (void) __attribute__ ((__nothrow__ , __leaf__));
841 extern long int jrand48 (unsigned short int __xsubi[3])
842      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
843
844
845 extern void srand48 (long int __seedval) __attribute__ ((__nothrow__ ,
    __leaf__));
846 extern unsigned short int *seed48 (unsigned short int __seed16v[3])
847      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
848 extern void lcong48 (unsigned short int __param[7]) __attribute__ ((
    __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
849
850
851
852
853
854 struct drand48_data
855   {
856     unsigned short int __x[3];
857     unsigned short int __old_x[3];
858     unsigned short int __c;
859     unsigned short int __init;
860     __extension__ unsigned long long int __a;
861
862   };
863
864
865 extern int drand48_r (struct drand48_data *__restrict __buffer,
866         double *__restrict __result) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
867 extern int erand48_r (unsigned short int __xsubi[3],
868         struct drand48_data *__restrict __buffer,
869         double *__restrict __result) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
870
871
872 extern int lrand48_r (struct drand48_data *__restrict __buffer,
873         long int *__restrict __result)
874      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
875 extern int nrand48_r (unsigned short int __xsubi[3],
876         struct drand48_data *__restrict __buffer,
877         long int *__restrict __result)
878      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
879
880
```

```
extern int mrand48_r (struct drand48_data *__restrict __buffer,
        long int *__restrict __result)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));
extern int jrand48_r (unsigned short int __xsubi[3],
        struct drand48_data *__restrict __buffer,
        long int *__restrict __result)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));


extern int srand48_r (long int __seedval, struct drand48_data *__buffer)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (2)));

extern int seed48_r (unsigned short int __seed16v[3],
       struct drand48_data *__buffer) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (1, 2)));

extern int lcong48_r (unsigned short int __param[7],
        struct drand48_data *__buffer)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));









extern void *malloc (size_t __size) __attribute__ ((__nothrow__ , __leaf__
   )) __attribute__ ((__malloc__)) ;

extern void *calloc (size_t __nmemb, size_t __size)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
    ;









extern void *realloc (void *__ptr, size_t __size)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
   __warn_unused_result__));

extern void free (void *__ptr) __attribute__ ((__nothrow__ , __leaf__));

```

```
927
928
929
930 extern void cfree (void *__ptr) __attribute__ ((__nothrow__ , __leaf__));
931
932
933
934 # 1 "/usr/include/alloca.h" 1 3 4
935 # 24 "/usr/include/alloca.h" 3 4
936 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
937 # 25 "/usr/include/alloca.h" 2 3 4
938
939
940
941
942
943
944
945 extern void *alloca (size_t __size) __attribute__ ((__nothrow__ , __leaf__
    ));
946
947
948
949
950
951
952 # 493 "/usr/include/stdlib.h" 2 3 4
953
954
955
956
957
958 extern void *valloc (size_t __size) __attribute__ ((__nothrow__ , __leaf__
    )) __attribute__ ((__malloc__)) ;
959
960
961
962
963 extern int posix_memalign (void **__memptr, size_t __alignment, size_t
    __size)
964      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1))) ;
965
966
967
968
969 extern void *aligned_alloc (size_t __alignment, size_t __size)
970      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
     __attribute__ ((__alloc_size__ (2))) ;
971
972
973
974
```

```
975 extern void abort (void) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__noreturn__));
976
977
978
979 extern int atexit (void (*__func) (void)) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
980
981
982
983
984
985
986
987 extern int at_quick_exit (void (*__func) (void)) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
988
989
990
991
992
993
994
995 extern int on_exit (void (*__func) (int __status, void *__arg), void *
       __arg)
996   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1)));
997
998
999
1000
1001
1002
1003 extern void exit (int __status) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__noreturn__));
1004
1005
1006
1007
1008
1009 extern void quick_exit (int __status) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__noreturn__));
1010
1011
1012
1013
1014
1015
1016
1017 extern void _Exit (int __status) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__noreturn__));
1018
1019
1020
```

```
1021
1022
1023
1024 extern char *getenv (const char *__name) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (1))) ;
1025
1026 # 578 "/usr/include/stdlib.h" 3 4
1027 extern int putenv (char *__string) __attribute__ ((__nothrow__ , __leaf__)
        ) __attribute__ ((__nonnull__ (1)));
1028
1029
1030
1031
1032
1033 extern int setenv (const char *__name, const char *__value, int __replace)
1034      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (2)));
1035
1036
1037 extern int unsetenv (const char *__name) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (1)));
1038
1039
1040
1041
1042
1043
1044 extern int clearenv (void) __attribute__ ((__nothrow__ , __leaf__));
1045 # 606 "/usr/include/stdlib.h" 3 4
1046 extern char *mktemp (char *__template) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (1)));
1047 # 620 "/usr/include/stdlib.h" 3 4
1048 extern int mkstemp (char *__template) __attribute__ ((__nonnull__ (1))) ;
1049 # 642 "/usr/include/stdlib.h" 3 4
1050 extern int mkstemps (char *__template, int __suffixlen) __attribute__ ((
        __nonnull__ (1))) ;
1051 # 663 "/usr/include/stdlib.h" 3 4
1052 extern char *mkdtemp (char *__template) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (1))) ;
1053 # 712 "/usr/include/stdlib.h" 3 4
1054
1055
1056
1057
1058
1059 extern int system (const char *__command) ;
1060
1061 # 734 "/usr/include/stdlib.h" 3 4
1062 extern char *realpath (const char *__restrict __name,
1063          char *__restrict __resolved) __attribute__ ((__nothrow__ ,
        __leaf__)) ;
1064
1065
1066
```

```
1067
1068
1069
1070 typedef int (*__compar_fn_t) (const void *, const void *);
1071 # 752 "/usr/include/stdlib.h" 3 4
1072
1073
1074
1075 extern void *bsearch (const void *__key, const void *__base,
1076         size_t __nmemb, size_t __size, __compar_fn_t __compar)
1077      __attribute__ ((__nonnull__ (1, 2, 5))) ;
1078
1079
1080
1081
1082
1083
1084
1085 extern void qsort (void *__base, size_t __nmemb, size_t __size,
1086      __compar_fn_t __compar) __attribute__ ((__nonnull__ (1, 4)));
1087 # 775 "/usr/include/stdlib.h" 3 4
1088 extern int abs (int __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)) ;
1089 extern long int labs (long int __x) __attribute__ ((__nothrow__ , __leaf__
       )) __attribute__ ((__const__)) ;
1090
1091
1092
1093 __extension__ extern long long int llabs (long long int __x)
1094      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
       ;
1095
1096
1097
1098
1099
1100
1101
1102 extern div_t div (int __numer, int __denom)
1103      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
       ;
1104 extern ldiv_t ldiv (long int __numer, long int __denom)
1105      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
       ;
1106
1107
1108
1109
1110 __extension__ extern lldiv_t lldiv (long long int __numer,
1111         long long int __denom)
1112      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
       ;
1113
1114 # 812 "/usr/include/stdlib.h" 3 4
```

```
1115 extern char *ecvt (double __value, int __ndigit, int *__restrict __decpt,
1116      int *__restrict __sign) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__nonnull__ (3, 4))) ;
1117
1118
1119
1120
1121 extern char *fcvt (double __value, int __ndigit, int *__restrict __decpt,
1122      int *__restrict __sign) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__nonnull__ (3, 4))) ;
1123
1124
1125
1126
1127 extern char *gcvt (double __value, int __ndigit, char *__buf)
1128      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3))) ;
1129
1130
1131
1132
1133 extern char *qecvt (long double __value, int __ndigit,
1134       int *__restrict __decpt, int *__restrict __sign)
1135      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3, 4))) ;
1136 extern char *qfcvt (long double __value, int __ndigit,
1137       int *__restrict __decpt, int *__restrict __sign)
1138      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3, 4))) ;
1139 extern char *qgcvt (long double __value, int __ndigit, char *__buf)
1140      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3))) ;
1141
1142
1143
1144
1145 extern int ecvt_r (double __value, int __ndigit, int *__restrict __decpt,
1146      int *__restrict __sign, char *__restrict __buf,
1147      size_t __len) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
     ((__nonnull__ (3, 4, 5)));
1148 extern int fcvt_r (double __value, int __ndigit, int *__restrict __decpt,
1149      int *__restrict __sign, char *__restrict __buf,
1150      size_t __len) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
     ((__nonnull__ (3, 4, 5)));
1151
1152 extern int qecvt_r (long double __value, int __ndigit,
1153       int *__restrict __decpt, int *__restrict __sign,
1154       char *__restrict __buf, size_t __len)
1155      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3, 4, 5)));
1156 extern int qfcvt_r (long double __value, int __ndigit,
1157       int *__restrict __decpt, int *__restrict __sign,
1158       char *__restrict __buf, size_t __len)
```

```
1159      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (3, 4, 5)));
1160
1161
1162
1163
1164
1165
1166 extern int mblen (const char *__s, size_t __n) __attribute__ ((__nothrow__
     , __leaf__));
1167
1168
1169 extern int mbtowc (wchar_t *__restrict __pwc,
1170      const char *__restrict __s, size_t __n) __attribute__ ((__nothrow__ ,
     __leaf__));
1171
1172
1173 extern int wctomb (char *__s, wchar_t __wchar) __attribute__ ((__nothrow__
     , __leaf__));
1174
1175
1176
1177 extern size_t mbstowcs (wchar_t *__restrict __pwcs,
1178    const char *__restrict __s, size_t __n) __attribute__ ((__nothrow__ ,
    __leaf__));
1179
1180 extern size_t wcstombs (char *__restrict __s,
1181    const wchar_t *__restrict __pwcs, size_t __n)
1182      __attribute__ ((__nothrow__ , __leaf__));
1183
1184
1185
1186
1187
1188
1189
1190
1191 extern int rpmatch (const char *__response) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1))) ;
1192 # 899 "/usr/include/stdlib.h" 3 4
1193 extern int getsubopt (char **__restrict __optionp,
1194         char *const *__restrict __tokens,
1195         char **__restrict __valuep)
1196      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2, 3))) ;
1197 # 951 "/usr/include/stdlib.h" 3 4
1198 extern int getloadavg (double __loadavg[], int __nelem)
1199      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
1200
1201
1202 # 1 "/usr/include/x86_64-linux-gnu/bits/stdlib-float.h" 1 3 4
1203 # 956 "/usr/include/stdlib.h" 2 3 4
1204 # 968 "/usr/include/stdlib.h" 3 4
```

```
1205
1206 # 4 "headers/all_headers.h" 2
1207 # 1 "/usr/include/stdio.h" 1 3 4
1208 # 29 "/usr/include/stdio.h" 3 4
1209
1210
1211
1212
1213 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
1214 # 34 "/usr/include/stdio.h" 2 3 4
1215 # 44 "/usr/include/stdio.h" 3 4
1216 struct _IO_FILE;
1217
1218
1219
1220 typedef struct _IO_FILE FILE;
1221
1222
1223
1224
1225
1226 # 64 "/usr/include/stdio.h" 3 4
1227 typedef struct _IO_FILE __FILE;
1228 # 74 "/usr/include/stdio.h" 3 4
1229 # 1 "/usr/include/libio.h" 1 3 4
1230 # 31 "/usr/include/libio.h" 3 4
1231 # 1 "/usr/include/_G_config.h" 1 3 4
1232 # 15 "/usr/include/_G_config.h" 3 4
1233 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
1234 # 16 "/usr/include/_G_config.h" 2 3 4
1235
1236
1237
1238
1239 # 1 "/usr/include/wchar.h" 1 3 4
1240 # 82 "/usr/include/wchar.h" 3 4
1241 typedef struct
1242 {
1243   int __count;
1244   union
1245   {
1246
1247     unsigned int __wch;
1248
1249
1250
1251     char __wchb[4];
1252   } __value;
1253 } __mbstate_t;
1254 # 21 "/usr/include/_G_config.h" 2 3 4
1255 typedef struct
1256 {
1257   __off_t __pos;
1258   __mbstate_t __state;
```

```
} _G_fpos_t;
typedef struct
{
  __off64_t __pos;
  __mbstate_t __state;
} _G_fpos64_t;
# 32 "/usr/include/libio.h" 2 3 4
# 49 "/usr/include/libio.h" 3 4
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdarg.h" 1 3 4
# 40 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdarg.h" 3 4
typedef __builtin_va_list __gnuc_va_list;
# 50 "/usr/include/libio.h" 2 3 4
# 144 "/usr/include/libio.h" 3 4
struct _IO_jump_t; struct _IO_FILE;
# 154 "/usr/include/libio.h" 3 4
typedef void _IO_lock_t;




struct _IO_marker {
  struct _IO_marker *_next;
  struct _IO_FILE *_sbuf;



  int _pos;
# 177 "/usr/include/libio.h" 3 4
};


enum __codecvt_result
{
  __codecvt_ok,
  __codecvt_partial,
  __codecvt_error,
  __codecvt_noconv
};
# 245 "/usr/include/libio.h" 3 4
struct _IO_FILE {
  int _flags;




  char* _IO_read_ptr;
  char* _IO_read_end;
  char* _IO_read_base;
  char* _IO_write_base;
  char* _IO_write_ptr;
  char* _IO_write_end;
  char* _IO_buf_base;
  char* _IO_buf_end;
```

```
1313
1314   char *_IO_save_base;
1315   char *_IO_backup_base;
1316   char *_IO_save_end;
1317
1318   struct _IO_marker *_markers;
1319
1320   struct _IO_FILE *_chain;
1321
1322   int _fileno;
1323
1324
1325
1326   int _flags2;
1327
1328   __off_t _old_offset;
1329
1330
1331
1332   unsigned short _cur_column;
1333   signed char _vtable_offset;
1334   char _shortbuf[1];
1335
1336
1337
1338   _IO_lock_t *_lock;
1339 # 293 "/usr/include/libio.h" 3 4
1340   __off64_t _offset;
1341 # 302 "/usr/include/libio.h" 3 4
1342   void *__pad1;
1343   void *__pad2;
1344   void *__pad3;
1345   void *__pad4;
1346   size_t __pad5;
1347
1348   int _mode;
1349
1350   char _unused2[15 * sizeof (int) - 4 * sizeof (void *) - sizeof (size_t)
       ];
1351
1352 };
1353
1354
1355 typedef struct _IO_FILE _IO_FILE;
1356
1357
1358 struct _IO_FILE_plus;
1359
1360 extern struct _IO_FILE_plus _IO_2_1_stdin_;
1361 extern struct _IO_FILE_plus _IO_2_1_stdout_;
1362 extern struct _IO_FILE_plus _IO_2_1_stderr_;
1363 # 338 "/usr/include/libio.h" 3 4
1364 typedef __ssize_t __io_read_fn (void *__cookie, char *__buf, size_t
       __nbytes);
```

```
1365
1366
1367
1368
1369
1370
1371
1372 typedef __ssize_t __io_write_fn (void *__cookie, const char *__buf,
1373      size_t __n);
1374
1375
1376
1377
1378
1379
1380
1381 typedef int __io_seek_fn (void *__cookie, __off64_t *__pos, int __w);
1382
1383
1384 typedef int __io_close_fn (void *__cookie);
1385 # 390 "/usr/include/libio.h" 3 4
1386 extern int __underflow (_IO_FILE *);
1387 extern int __uflow (_IO_FILE *);
1388 extern int __overflow (_IO_FILE *, int);
1389 # 434 "/usr/include/libio.h" 3 4
1390 extern int _IO_getc (_IO_FILE *__fp);
1391 extern int _IO_putc (int __c, _IO_FILE *__fp);
1392 extern int _IO_feof (_IO_FILE *__fp) __attribute__ ((__nothrow__ ,
       __leaf__));
1393 extern int _IO_ferror (_IO_FILE *__fp) __attribute__ ((__nothrow__ ,
       __leaf__));
1394
1395 extern int _IO_peekc_locked (_IO_FILE *__fp);
1396
1397
1398
1399
1400
1401 extern void _IO_flockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
       __leaf__));
1402 extern void _IO_funlockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
       __leaf__));
1403 extern int _IO_ftrylockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
       __leaf__));
1404 # 464 "/usr/include/libio.h" 3 4
1405 extern int _IO_vfscanf (_IO_FILE * __restrict, const char * __restrict,
       __gnuc_va_list, int *__restrict);
1406 extern int _IO_vfprintf (_IO_FILE *__restrict, const char *__restrict,
1407      __gnuc_va_list);
1408 extern __ssize_t _IO_padn (_IO_FILE *, int, __ssize_t);
1409 extern size_t _IO_sgetn (_IO_FILE *, void *, size_t);
1410
1411 extern __off64_t _IO_seekoff (_IO_FILE *, __off64_t, int, int);
1412 extern __off64_t _IO_seekpos (_IO_FILE *, __off64_t, int);
1413
```

```
1414
1415 extern void _IO_free_backup_area (_IO_FILE *) __attribute__ ((__nothrow__
     , __leaf__));
1416 # 75 "/usr/include/stdio.h" 2 3 4
1417
1418
1419
1420
1421 typedef __gnuc_va_list va_list;
1422 # 108 "/usr/include/stdio.h" 3 4
1423
1424
1425 typedef _G_fpos_t fpos_t;
1426
1427
1428
1429
1430 # 164 "/usr/include/stdio.h" 3 4
1431 # 1 "/usr/include/x86_64-linux-gnu/bits/stdio_lim.h" 1 3 4
1432 # 165 "/usr/include/stdio.h" 2 3 4
1433
1434
1435
1436 extern struct _IO_FILE *stdin;
1437 extern struct _IO_FILE *stdout;
1438 extern struct _IO_FILE *stderr;
1439
1440
1441
1442
1443
1444
1445
1446 extern int remove (const char *__filename) __attribute__ ((__nothrow__ ,
     __leaf__));
1447
1448 extern int rename (const char *__old, const char *__new) __attribute__ ((
     __nothrow__ , __leaf__));
1449
1450
1451
1452
1453 extern int renameat (int __oldfd, const char *__old, int __newfd,
1454        const char *__new) __attribute__ ((__nothrow__ , __leaf__));
1455
1456
1457
1458
1459
1460
1461
1462
1463 extern FILE *tmpfile (void) ;
1464 # 209 "/usr/include/stdio.h" 3 4
```

```
extern char *tmpnam (char *__s) __attribute__ ((__nothrow__ , __leaf__)) ;
extern char *tmpnam_r (char *__s) __attribute__ ((__nothrow__ , __leaf__))
    ;
# 227 "/usr/include/stdio.h" 3 4
extern char *tempnam (const char *__dir, const char *__pfx)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
    ;
extern int fclose (FILE *__stream);
extern int fflush (FILE *__stream);
# 252 "/usr/include/stdio.h" 3 4
extern int fflush_unlocked (FILE *__stream);
# 266 "/usr/include/stdio.h" 3 4
extern FILE *fopen (const char *__restrict __filename,
      const char *__restrict __modes) ;
extern FILE *freopen (const char *__restrict __filename,
        const char *__restrict __modes,
        FILE *__restrict __stream) ;
# 295 "/usr/include/stdio.h" 3 4
# 306 "/usr/include/stdio.h" 3 4
extern FILE *fdopen (int __fd, const char *__modes) __attribute__ ((
   __nothrow__ , __leaf__)) ;
# 319 "/usr/include/stdio.h" 3 4
extern FILE *fmemopen (void *__s, size_t __len, const char *__modes)
  __attribute__ ((__nothrow__ , __leaf__)) ;
```

```
1516
1517
1518
1519 extern FILE *open_memstream (char **__bufloc, size_t *__sizeloc)
       __attribute__ ((__nothrow__ , __leaf__)) ;
1520
1521
1522
1523
1524
1525
1526 extern void setbuf (FILE *__restrict __stream, char *__restrict __buf)
       __attribute__ ((__nothrow__ , __leaf__));
1527
1528
1529
1530 extern int setvbuf (FILE *__restrict __stream, char *__restrict __buf,
1531       int __modes, size_t __n) __attribute__ ((__nothrow__ , __leaf__));
1532
1533
1534
1535
1536
1537 extern void setbuffer (FILE *__restrict __stream, char *__restrict __buf,
1538         size_t __size) __attribute__ ((__nothrow__ , __leaf__));
1539
1540
1541 extern void setlinebuf (FILE *__stream) __attribute__ ((__nothrow__ ,
       __leaf__));
1542
1543
1544
1545
1546
1547
1548
1549
1550 extern int fprintf (FILE *__restrict __stream,
1551       const char *__restrict __format, ...);
1552
1553
1554
1555
1556 extern int printf (const char *__restrict __format, ...);
1557
1558 extern int sprintf (char *__restrict __s,
1559       const char *__restrict __format, ...) __attribute__ ((__nothrow__));
1560
1561
1562
1563
1564
1565 extern int vfprintf (FILE *__restrict __s, const char *__restrict __format
       ,
```

```
1566        __gnuc_va_list __arg);
1567
1568
1569
1570
1571 extern int vprintf (const char *__restrict __format, __gnuc_va_list __arg)
       ;
1572
1573 extern int vsprintf (char *__restrict __s, const char *__restrict __format
       ,
1574        __gnuc_va_list __arg) __attribute__ ((__nothrow__));
1575
1576
1577
1578
1579
1580 extern int snprintf (char *__restrict __s, size_t __maxlen,
1581        const char *__restrict __format, ...)
1582      __attribute__ ((__nothrow__)) __attribute__ ((__format__ (__printf__,
       3, 4)));
1583
1584 extern int vsnprintf (char *__restrict __s, size_t __maxlen,
1585         const char *__restrict __format, __gnuc_va_list __arg)
1586      __attribute__ ((__nothrow__)) __attribute__ ((__format__ (__printf__,
       3, 0)));
1587
1588 # 412 "/usr/include/stdio.h" 3 4
1589 extern int vdprintf (int __fd, const char *__restrict __fmt,
1590        __gnuc_va_list __arg)
1591      __attribute__ ((__format__ (__printf__, 2, 0)));
1592 extern int dprintf (int __fd, const char *__restrict __fmt, ...)
1593      __attribute__ ((__format__ (__printf__, 2, 3)));
1594
1595
1596
1597
1598
1599
1600
1601
1602 extern int fscanf (FILE *__restrict __stream,
1603      const char *__restrict __format, ...) ;
1604
1605
1606
1607
1608 extern int scanf (const char *__restrict __format, ...) ;
1609
1610 extern int sscanf (const char *__restrict __s,
1611      const char *__restrict __format, ...) __attribute__ ((__nothrow__ ,
       __leaf__));
1612 # 443 "/usr/include/stdio.h" 3 4
1613 extern int fscanf (FILE *__restrict __stream, const char *__restrict
       __format, ...) __asm__ ("" "__isoc99_fscanf")
```

```
1614
1615                                ;
1616 extern int scanf (const char *__restrict __format, ...) __asm__ ("" "
       __isoc99_scanf")
1617                                ;
1618 extern int sscanf (const char *__restrict __s, const char *__restrict
       __format, ...) __asm__ ("" "__isoc99_sscanf") __attribute__ ((
       __nothrow__ , __leaf__))
1619
1620                       ;
1621 # 463 "/usr/include/stdio.h" 3 4
1622
1623
1624
1625
1626
1627
1628
1629
1630 extern int vfscanf (FILE *__restrict __s, const char *__restrict __format,
1631       __gnuc_va_list __arg)
1632      __attribute__ ((__format__ (__scanf__, 2, 0))) ;
1633
1634
1635
1636
1637
1638 extern int vscanf (const char *__restrict __format, __gnuc_va_list __arg)
1639      __attribute__ ((__format__ (__scanf__, 1, 0))) ;
1640
1641
1642 extern int vsscanf (const char *__restrict __s,
1643       const char *__restrict __format, __gnuc_va_list __arg)
1644      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__format__ (
       __scanf__, 2, 0)));
1645 # 494 "/usr/include/stdio.h" 3 4
1646 extern int vfscanf (FILE *__restrict __s, const char *__restrict __format,
       __gnuc_va_list __arg) __asm__ ("" "__isoc99_vfscanf")
1647
1648
1649
1650      __attribute__ ((__format__ (__scanf__, 2, 0))) ;
1651 extern int vscanf (const char *__restrict __format, __gnuc_va_list __arg)
       __asm__ ("" "__isoc99_vscanf")
1652
1653      __attribute__ ((__format__ (__scanf__, 1, 0))) ;
1654 extern int vsscanf (const char *__restrict __s, const char *__restrict
       __format, __gnuc_va_list __arg) __asm__ ("" "__isoc99_vsscanf")
       __attribute__ ((__nothrow__ , __leaf__))
1655
1656
1657
1658      __attribute__ ((__format__ (__scanf__, 2, 0)));
1659 # 522 "/usr/include/stdio.h" 3 4
```

```
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669 extern int fgetc (FILE *__stream);
1670 extern int getc (FILE *__stream);
1671
1672
1673
1674
1675
1676 extern int getchar (void);
1677
1678 # 550 "/usr/include/stdio.h" 3 4
1679 extern int getc_unlocked (FILE *__stream);
1680 extern int getchar_unlocked (void);
1681 # 561 "/usr/include/stdio.h" 3 4
1682 extern int fgetc_unlocked (FILE *__stream);
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694 extern int fputc (int __c, FILE *__stream);
1695 extern int putc (int __c, FILE *__stream);
1696
1697
1698
1699
1700
1701 extern int putchar (int __c);
1702
1703 # 594 "/usr/include/stdio.h" 3 4
1704 extern int fputc_unlocked (int __c, FILE *__stream);
1705
1706
1707
1708
1709
1710
1711
1712 extern int putc_unlocked (int __c, FILE *__stream);
1713 extern int putchar_unlocked (int __c);
```

```
1714
1715
1716
1717
1718
1719
1720 extern int getw (FILE *__stream);
1721
1722
1723 extern int putw (int __w, FILE *__stream);
1724
1725
1726
1727
1728
1729
1730
1731
1732 extern char *fgets (char *__restrict __s, int __n, FILE *__restrict
        __stream)
1733      ;
1734 # 640 "/usr/include/stdio.h" 3 4
1735
1736 # 665 "/usr/include/stdio.h" 3 4
1737 extern __ssize_t __getdelim (char **__restrict __lineptr,
1738           size_t *__restrict __n, int __delimiter,
1739           FILE *__restrict __stream) ;
1740 extern __ssize_t getdelim (char **__restrict __lineptr,
1741          size_t *__restrict __n, int __delimiter,
1742          FILE *__restrict __stream) ;
1743
1744
1745
1746
1747
1748
1749
1750 extern __ssize_t getline (char **__restrict __lineptr,
1751        size_t *__restrict __n,
1752        FILE *__restrict __stream) ;
1753
1754
1755
1756
1757
1758
1759
1760
1761 extern int fputs (const char *__restrict __s, FILE *__restrict __stream);
1762
1763
1764
1765
1766
```

```
1767 extern int puts (const char *__s);
1768
1769
1770
1771
1772
1773
1774 extern int ungetc (int __c, FILE *__stream);
1775
1776
1777
1778
1779
1780
1781 extern size_t fread (void *__restrict __ptr, size_t __size,
1782        size_t __n, FILE *__restrict __stream) ;
1783
1784
1785
1786
1787 extern size_t fwrite (const void *__restrict __ptr, size_t __size,
1788         size_t __n, FILE *__restrict __s);
1789
1790 # 737 "/usr/include/stdio.h" 3 4
1791 extern size_t fread_unlocked (void *__restrict __ptr, size_t __size,
1792          size_t __n, FILE *__restrict __stream) ;
1793 extern size_t fwrite_unlocked (const void *__restrict __ptr, size_t __size
    ,
1794          size_t __n, FILE *__restrict __stream);
1795
1796
1797
1798
1799
1800
1801
1802
1803 extern int fseek (FILE *__stream, long int __off, int __whence);
1804
1805
1806
1807
1808 extern long int ftell (FILE *__stream) ;
1809
1810
1811
1812
1813 extern void rewind (FILE *__stream);
1814
1815 # 773 "/usr/include/stdio.h" 3 4
1816 extern int fseeko (FILE *__stream, __off_t __off, int __whence);
1817
1818
1819
```

```
1820
1821 extern __off_t ftello (FILE *__stream) ;
1822 # 792 "/usr/include/stdio.h" 3 4
1823
1824
1825
1826
1827
1828
1829 extern int fgetpos (FILE *__restrict __stream, fpos_t *__restrict __pos);
1830
1831
1832
1833
1834 extern int fsetpos (FILE *__stream, const fpos_t *__pos);
1835 # 815 "/usr/include/stdio.h" 3 4
1836
1837 # 824 "/usr/include/stdio.h" 3 4
1838
1839
1840 extern void clearerr (FILE *__stream) __attribute__ ((__nothrow__ ,
       __leaf__));
1841
1842 extern int feof (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__))
       ;
1843
1844 extern int ferror (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)
       ) ;
1845
1846
1847
1848
1849 extern void clearerr_unlocked (FILE *__stream) __attribute__ ((__nothrow__
        , __leaf__));
1850 extern int feof_unlocked (FILE *__stream) __attribute__ ((__nothrow__ ,
       __leaf__)) ;
1851 extern int ferror_unlocked (FILE *__stream) __attribute__ ((__nothrow__ ,
       __leaf__)) ;
1852
1853
1854
1855
1856
1857
1858
1859
1860 extern void perror (const char *__s);
1861
1862
1863
1864
1865
1866
1867 # 1 "/usr/include/x86_64-linux-gnu/bits/sys_errlist.h" 1 3 4
```

```
# 26 "/usr/include/x86_64-linux-gnu/bits/sys_errlist.h" 3 4
extern int sys_nerr;
extern const char *const sys_errlist[];
# 854 "/usr/include/stdio.h" 2 3 4



extern int fileno (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)
   ) ;



extern int fileno_unlocked (FILE *__stream) __attribute__ ((__nothrow__ ,
   __leaf__)) ;
# 873 "/usr/include/stdio.h" 3 4
extern FILE *popen (const char *__command, const char *__modes) ;




extern int pclose (FILE *__stream);




extern char *ctermid (char *__s) __attribute__ ((__nothrow__ , __leaf__));
# 913 "/usr/include/stdio.h" 3 4
extern void flockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
   __leaf__));



extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
   __leaf__)) ;


extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
   __leaf__));
# 943 "/usr/include/stdio.h" 3 4

# 5 "headers/all_headers.h" 2
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdbool.h" 1 3 4
# 6 "headers/all_headers.h" 2
# 1 "/usr/include/math.h" 1 3 4
# 28 "/usr/include/math.h" 3 4



# 1 "/usr/include/x86_64-linux-gnu/bits/huge_val.h" 1 3 4
```

```
1917 # 33 "/usr/include/math.h" 2 3 4
1918
1919 # 1 "/usr/include/x86_64-linux-gnu/bits/huge_valf.h" 1 3 4
1920 # 35 "/usr/include/math.h" 2 3 4
1921 # 1 "/usr/include/x86_64-linux-gnu/bits/huge_vall.h" 1 3 4
1922 # 36 "/usr/include/math.h" 2 3 4
1923
1924
1925 # 1 "/usr/include/x86_64-linux-gnu/bits/inf.h" 1 3 4
1926 # 39 "/usr/include/math.h" 2 3 4
1927
1928
1929 # 1 "/usr/include/x86_64-linux-gnu/bits/nan.h" 1 3 4
1930 # 42 "/usr/include/math.h" 2 3 4
1931
1932
1933
1934 # 1 "/usr/include/x86_64-linux-gnu/bits/mathdef.h" 1 3 4
1935 # 28 "/usr/include/x86_64-linux-gnu/bits/mathdef.h" 3 4
1936 typedef float float_t;
1937 typedef double double_t;
1938 # 46 "/usr/include/math.h" 2 3 4
1939 # 69 "/usr/include/math.h" 3 4
1940 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
1941 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
1942
1943
1944 extern double acos (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __acos (double __x) __attribute__ ((__nothrow__ ,
       __leaf__));
1945
1946 extern double asin (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __asin (double __x) __attribute__ ((__nothrow__ ,
       __leaf__));
1947
1948 extern double atan (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __atan (double __x) __attribute__ ((__nothrow__ ,
       __leaf__));
1949
1950 extern double atan2 (double __y, double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern double __atan2 (double __y, double __x)
       __attribute__ ((__nothrow__ , __leaf__));
1951
1952
1953 extern double cos (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __cos (double __x) __attribute__ ((__nothrow__ , __leaf__
       ));
1954
1955 extern double sin (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __sin (double __x) __attribute__ ((__nothrow__ , __leaf__
       ));
1956
1957 extern double tan (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __tan (double __x) __attribute__ ((__nothrow__ , __leaf__
```

```
   ));
1958
1959
1960
1961
1962 extern double cosh (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __cosh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1963
1964 extern double sinh (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __sinh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1965
1966 extern double tanh (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __tanh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1967
1968 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
1969
1970
1971 extern double acosh (double __x) __attribute__ ((__nothrow__ , __leaf__));
    extern double __acosh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1972
1973 extern double asinh (double __x) __attribute__ ((__nothrow__ , __leaf__));
    extern double __asinh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1974
1975 extern double atanh (double __x) __attribute__ ((__nothrow__ , __leaf__));
    extern double __atanh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1976
1977
1978
1979
1980
1981
1982
1983 extern double exp (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __exp (double __x) __attribute__ ((__nothrow__ , __leaf__
   ));
1984
1985
1986 extern double frexp (double __x, int *__exponent) __attribute__ ((
   __nothrow__ , __leaf__)); extern double __frexp (double __x, int *
   __exponent) __attribute__ ((__nothrow__ , __leaf__));
1987
1988
1989 extern double ldexp (double __x, int __exponent) __attribute__ ((
   __nothrow__ , __leaf__)); extern double __ldexp (double __x, int
   __exponent) __attribute__ ((__nothrow__ , __leaf__));
1990
1991
```

```
1992 extern double log (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __log (double __x) __attribute__ ((__nothrow__ , __leaf__
     ));
1993
1994
1995 extern double log10 (double __x) __attribute__ ((__nothrow__ , __leaf__));
      extern double __log10 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1996
1997
1998 extern double modf (double __x, double *__iptr) __attribute__ ((
     __nothrow__ , __leaf__)); extern double __modf (double __x, double *
     __iptr) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
     __nonnull__ (2)));
1999
2000 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2001
2002
2003 extern double expm1 (double __x) __attribute__ ((__nothrow__ , __leaf__));
      extern double __expm1 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2004
2005
2006 extern double log1p (double __x) __attribute__ ((__nothrow__ , __leaf__));
      extern double __log1p (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2007
2008
2009 extern double logb (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __logb (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2010
2011
2012
2013
2014
2015
2016 extern double exp2 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __exp2 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2017
2018
2019 extern double log2 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __log2 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2020
2021
2022
2023
2024
2025
2026
2027
```

```
2028 extern double pow (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)); extern double __pow (double __x, double __y) __attribute__
        ((__nothrow__ , __leaf__));
2029
2030
2031 extern double sqrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
        extern double __sqrt (double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2032
2033
2034
2035
2036
2037 extern double hypot (double __x, double __y) __attribute__ ((__nothrow__ ,
         __leaf__)); extern double __hypot (double __x, double __y)
        __attribute__ ((__nothrow__ , __leaf__));
2038
2039
2040
2041
2042
2043
2044 extern double cbrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
        extern double __cbrt (double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2045
2046
2047
2048
2049
2050
2051
2052
2053 extern double ceil (double __x) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__)); extern double __ceil (double __x)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2054
2055
2056 extern double fabs (double __x) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__)); extern double __fabs (double __x)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2057
2058
2059 extern double floor (double __x) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__)); extern double __floor (double __x)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2060
2061
2062 extern double fmod (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)); extern double __fmod (double __x, double __y) __attribute__
         ((__nothrow__ , __leaf__));
2063
2064
2065
```

```
2066
2067 extern int __isinf (double __value) __attribute__ ((__nothrow__ , __leaf__
     )) __attribute__ ((__const__));
2068
2069
2070 extern int __finite (double __value) __attribute__ ((__nothrow__ ,
     __leaf__)) __attribute__ ((__const__));
2071
2072
2073
2074
2075
2076 extern int isinf (double __value) __attribute__ ((__nothrow__ , __leaf__))
      __attribute__ ((__const__));
2077
2078
2079 extern int finite (double __value) __attribute__ ((__nothrow__ , __leaf__)
     ) __attribute__ ((__const__));
2080
2081
2082 extern double drem (double __x, double __y) __attribute__ ((__nothrow__ ,
     __leaf__)); extern double __drem (double __x, double __y) __attribute__
      ((__nothrow__ , __leaf__));
2083
2084
2085
2086 extern double significand (double __x) __attribute__ ((__nothrow__ ,
     __leaf__)); extern double __significand (double __x) __attribute__ ((
     __nothrow__ , __leaf__));
2087
2088
2089
2090
2091
2092 extern double copysign (double __x, double __y) __attribute__ ((
     __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
     __copysign (double __x, double __y) __attribute__ ((__nothrow__ ,
     __leaf__)) __attribute__ ((__const__));
2093
2094
2095
2096
2097
2098
2099 extern double nan (const char *__tagb) __attribute__ ((__nothrow__ ,
     __leaf__)) __attribute__ ((__const__)); extern double __nan (const char
      *__tagb) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
     __const__));
2100
2101
2102
2103
2104
```

```
2105 extern int __isnan (double __value) __attribute__ ((__nothrow__ , __leaf__
     )) __attribute__ ((__const__));
2106
2107
2108
2109 extern int isnan (double __value) __attribute__ ((__nothrow__ , __leaf__))
         __attribute__ ((__const__));
2110
2111
2112 extern double j0 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
         double __j0 (double) __attribute__ ((__nothrow__ , __leaf__));
2113 extern double j1 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
         double __j1 (double) __attribute__ ((__nothrow__ , __leaf__));
2114 extern double jn (int, double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __jn (int, double) __attribute__ ((__nothrow__ , __leaf__
     ));
2115 extern double y0 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
         double __y0 (double) __attribute__ ((__nothrow__ , __leaf__));
2116 extern double y1 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
         double __y1 (double) __attribute__ ((__nothrow__ , __leaf__));
2117 extern double yn (int, double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __yn (int, double) __attribute__ ((__nothrow__ , __leaf__
     ));
2118
2119
2120
2121
2122
2123
2124 extern double erf (double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __erf (double) __attribute__ ((__nothrow__ , __leaf__));
2125 extern double erfc (double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __erfc (double) __attribute__ ((__nothrow__ , __leaf__));
2126 extern double lgamma (double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __lgamma (double) __attribute__ ((__nothrow__ , __leaf__)
     );
2127
2128
2129
2130
2131
2132
2133 extern double tgamma (double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __tgamma (double) __attribute__ ((__nothrow__ , __leaf__)
     );
2134
2135
2136
2137
2138
2139 extern double gamma (double) __attribute__ ((__nothrow__ , __leaf__));
     extern double __gamma (double) __attribute__ ((__nothrow__ , __leaf__))
     ;
2140
```

```
2141
2142
2143
2144
2145
2146 extern double lgamma_r (double, int *__signgamp) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __lgamma_r (double, int *
       __signgamp) __attribute__ ((__nothrow__ , __leaf__));
2147
2148
2149
2150
2151
2152
2153
2154 extern double rint (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern double __rint (double __x) __attribute__ ((__nothrow__ ,
       __leaf__));
2155
2156
2157 extern double nextafter (double __x, double __y) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
       __nextafter (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2158
2159 extern double nexttoward (double __x, long double __y) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
       __nexttoward (double __x, long double __y) __attribute__ ((__nothrow__
       , __leaf__)) __attribute__ ((__const__));
2160
2161
2162
2163 extern double remainder (double __x, double __y) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __remainder (double __x, double
        __y) __attribute__ ((__nothrow__ , __leaf__));
2164
2165
2166
2167 extern double scalbn (double __x, int __n) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __scalbn (double __x, int __n) __attribute__
       ((__nothrow__ , __leaf__));
2168
2169
2170
2171 extern int ilogb (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern int __ilogb (double __x) __attribute__ ((__nothrow__ , __leaf__)
       );
2172
2173
2174
2175
2176 extern double scalbln (double __x, long int __n) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __scalbln (double __x, long int
        __n) __attribute__ ((__nothrow__ , __leaf__));
```

```
2177
2178
2179
2180 extern double nearbyint (double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern double __nearbyint (double __x) __attribute__ ((
        __nothrow__ , __leaf__));
2181
2182
2183
2184 extern double round (double __x) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__)); extern double __round (double __x)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2185
2186
2187
2188 extern double trunc (double __x) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__)); extern double __trunc (double __x)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2189
2190
2191
2192
2193 extern double remquo (double __x, double __y, int *__quo) __attribute__ ((
        __nothrow__ , __leaf__)); extern double __remquo (double __x, double
        __y, int *__quo) __attribute__ ((__nothrow__ , __leaf__));
2194
2195
2196
2197
2198
2199
2200 extern long int lrint (double __x) __attribute__ ((__nothrow__ , __leaf__)
        ); extern long int __lrint (double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2201 __extension__
2202 extern long long int llrint (double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long long int __llrint (double __x) __attribute__ ((
        __nothrow__ , __leaf__));
2203
2204
2205
2206 extern long int lround (double __x) __attribute__ ((__nothrow__ , __leaf__
        )); extern long int __lround (double __x) __attribute__ ((__nothrow__ ,
         __leaf__));
2207 __extension__
2208 extern long long int llround (double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long long int __llround (double __x) __attribute__
        ((__nothrow__ , __leaf__));
2209
2210
2211
2212 extern double fdim (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)); extern double __fdim (double __x, double __y) __attribute__
         ((__nothrow__ , __leaf__));
```

```
2213
2214
2215 extern double fmax (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern double __fmax (double
        __x, double __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
         ((__const__));
2216
2217
2218 extern double fmin (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern double __fmin (double
        __x, double __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
         ((__const__));
2219
2220
2221
2222 extern int __fpclassify (double __value) __attribute__ ((__nothrow__ ,
        __leaf__))
2223      __attribute__ ((__const__));
2224
2225
2226 extern int __signbit (double __value) __attribute__ ((__nothrow__ ,
        __leaf__))
2227      __attribute__ ((__const__));
2228
2229
2230
2231 extern double fma (double __x, double __y, double __z) __attribute__ ((
        __nothrow__ , __leaf__)); extern double __fma (double __x, double __y,
        double __z) __attribute__ ((__nothrow__ , __leaf__));
2232
2233
2234
2235
2236 # 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2237 extern double scalb (double __x, double __n) __attribute__ ((__nothrow__ ,
         __leaf__)); extern double __scalb (double __x, double __n)
        __attribute__ ((__nothrow__ , __leaf__));
2238 # 70 "/usr/include/math.h" 2 3 4
2239 # 88 "/usr/include/math.h" 3 4
2240 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
2241 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2242
2243
2244 extern float acosf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __acosf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2245
2246 extern float asinf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __asinf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2247
2248 extern float atanf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __atanf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
```

```
2249
2250 extern float atan2f (float __y, float __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern float __atan2f (float __y, float __x) __attribute__
        ((__nothrow__ , __leaf__));
2251
2252
2253 extern float cosf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __cosf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2254
2255 extern float sinf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __sinf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2256
2257 extern float tanf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __tanf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2258
2259
2260
2261
2262 extern float coshf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __coshf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2263
2264 extern float sinhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __sinhf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2265
2266 extern float tanhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __tanhf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2267
2268 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2269
2270
2271 extern float acoshf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __acoshf (float __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2272
2273 extern float asinhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __asinhf (float __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2274
2275 extern float atanhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __atanhf (float __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2276
2277
2278
2279
2280
2281
2282
```

```
2283 extern float expf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __expf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2284
2285
2286 extern float frexpf (float __x, int *__exponent) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __frexpf (float __x, int *
     __exponent) __attribute__ ((__nothrow__ , __leaf__));
2287
2288
2289 extern float ldexpf (float __x, int __exponent) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __ldexpf (float __x, int
     __exponent) __attribute__ ((__nothrow__ , __leaf__));
2290
2291
2292 extern float logf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __logf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2293
2294
2295 extern float log10f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __log10f (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2296
2297
2298 extern float modff (float __x, float *__iptr) __attribute__ ((__nothrow__
     , __leaf__)); extern float __modff (float __x, float *__iptr)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2)));
2299
2300 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2301
2302
2303 extern float expm1f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __expm1f (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2304
2305
2306 extern float log1pf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __log1pf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2307
2308
2309 extern float logbf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __logbf (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2310
2311
2312
2313
2314
2315
2316 extern float exp2f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __exp2f (float __x) __attribute__ ((__nothrow__ , __leaf__
```

```
   ));
2317
2318
2319 extern float log2f (float __x) __attribute__ ((__nothrow__ , __leaf__));
   extern float __log2f (float __x) __attribute__ ((__nothrow__ , __leaf__
   ));
2320
2321
2322
2323
2324
2325
2326
2327
2328 extern float powf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __powf (float __x, float __y) __attribute__ ((
   __nothrow__ , __leaf__));
2329
2330
2331 extern float sqrtf (float __x) __attribute__ ((__nothrow__ , __leaf__));
   extern float __sqrtf (float __x) __attribute__ ((__nothrow__ , __leaf__
   ));
2332
2333
2334
2335
2336
2337 extern float hypotf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __hypotf (float __x, float __y) __attribute__
   ((__nothrow__ , __leaf__));
2338
2339
2340
2341
2342
2343
2344 extern float cbrtf (float __x) __attribute__ ((__nothrow__ , __leaf__));
   extern float __cbrtf (float __x) __attribute__ ((__nothrow__ , __leaf__
   ));
2345
2346
2347
2348
2349
2350
2351
2352
2353 extern float ceilf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __ceilf (float __x)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2354
2355
2356 extern float fabsf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __fabsf (float __x)
```

```
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2357
2358
2359 extern float floorf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __floorf (float __x)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2360
2361
2362 extern float fmodf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __fmodf (float __x, float __y) __attribute__
   ((__nothrow__ , __leaf__));
2363
2364
2365
2366
2367 extern int __isinff (float __value) __attribute__ ((__nothrow__ , __leaf__
   )) __attribute__ ((__const__));
2368
2369
2370 extern int __finitef (float __value) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__));
2371
2372
2373
2374
2375
2376 extern int isinff (float __value) __attribute__ ((__nothrow__ , __leaf__))
    __attribute__ ((__const__));
2377
2378
2379 extern int finitef (float __value) __attribute__ ((__nothrow__ , __leaf__)
   ) __attribute__ ((__const__));
2380
2381
2382 extern float dremf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __dremf (float __x, float __y) __attribute__
   ((__nothrow__ , __leaf__));
2383
2384
2385
2386 extern float significandf (float __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __significandf (float __x) __attribute__ ((
   __nothrow__ , __leaf__));
2387
2388
2389
2390
2391
2392 extern float copysignf (float __x, float __y) __attribute__ ((__nothrow__
   , __leaf__)) __attribute__ ((__const__)); extern float __copysignf (
   float __x, float __y) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__));
2393
2394
```

```
2395
2396
2397
2398
2399 extern float nanf (const char *__tagb) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern float __nanf (const char
        *__tagb) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
       __const__));
2400
2401
2402
2403
2404
2405 extern int __isnanf (float __value) __attribute__ ((__nothrow__ , __leaf__
       )) __attribute__ ((__const__));
2406
2407
2408
2409 extern int isnanf (float __value) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__));
2410
2411
2412 extern float j0f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
       float __j0f (float) __attribute__ ((__nothrow__ , __leaf__));
2413 extern float j1f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
       float __j1f (float) __attribute__ ((__nothrow__ , __leaf__));
2414 extern float jnf (int, float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __jnf (int, float) __attribute__ ((__nothrow__ , __leaf__)
       );
2415 extern float y0f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
       float __y0f (float) __attribute__ ((__nothrow__ , __leaf__));
2416 extern float y1f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
       float __y1f (float) __attribute__ ((__nothrow__ , __leaf__));
2417 extern float ynf (int, float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __ynf (int, float) __attribute__ ((__nothrow__ , __leaf__)
       );
2418
2419
2420
2421
2422
2423
2424 extern float erff (float) __attribute__ ((__nothrow__ , __leaf__)); extern
        float __erff (float) __attribute__ ((__nothrow__ , __leaf__));
2425 extern float erfcf (float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __erfcf (float) __attribute__ ((__nothrow__ , __leaf__));
2426 extern float lgammaf (float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __lgammaf (float) __attribute__ ((__nothrow__ , __leaf__))
       ;
2427
2428
2429
2430
2431
```

```
2432
2433 extern float tgammaf (float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __tgammaf (float) __attribute__ ((__nothrow__ , __leaf__))
       ;
2434
2435
2436
2437
2438
2439 extern float gammaf (float) __attribute__ ((__nothrow__ , __leaf__));
       extern float __gammaf (float) __attribute__ ((__nothrow__ , __leaf__));
2440
2441
2442
2443
2444
2445
2446 extern float lgammaf_r (float, int *__signgamp) __attribute__ ((
       __nothrow__ , __leaf__)); extern float __lgammaf_r (float, int *
       __signgamp) __attribute__ ((__nothrow__ , __leaf__));
2447
2448
2449
2450
2451
2452
2453
2454 extern float rintf (float __x) __attribute__ ((__nothrow__ , __leaf__));
       extern float __rintf (float __x) __attribute__ ((__nothrow__ , __leaf__
       ));
2455
2456
2457 extern float nextafterf (float __x, float __y) __attribute__ ((__nothrow__
        , __leaf__)) __attribute__ ((__const__)); extern float __nextafterf (
       float __x, float __y) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__));
2458
2459 extern float nexttowardf (float __x, long double __y) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern float
       __nexttowardf (float __x, long double __y) __attribute__ ((__nothrow__
       , __leaf__)) __attribute__ ((__const__));
2460
2461
2462
2463 extern float remainderf (float __x, float __y) __attribute__ ((__nothrow__
        , __leaf__)); extern float __remainderf (float __x, float __y)
       __attribute__ ((__nothrow__ , __leaf__));
2464
2465
2466
2467 extern float scalbnf (float __x, int __n) __attribute__ ((__nothrow__ ,
       __leaf__)); extern float __scalbnf (float __x, int __n) __attribute__
       ((__nothrow__ , __leaf__));
2468
```

```
2469
2470
2471 extern int ilogbf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern int __ilogbf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2472
2473
2474
2475
2476 extern float scalblnf (float __x, long int __n) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __scalblnf (float __x, long int
     __n) __attribute__ ((__nothrow__ , __leaf__));
2477
2478
2479
2480 extern float nearbyintf (float __x) __attribute__ ((__nothrow__ , __leaf__
     )); extern float __nearbyintf (float __x) __attribute__ ((__nothrow__ ,
      __leaf__));
2481
2482
2483
2484 extern float roundf (float __x) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__const__)); extern float __roundf (float __x)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2485
2486
2487
2488 extern float truncf (float __x) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__const__)); extern float __truncf (float __x)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2489
2490
2491
2492
2493 extern float remquof (float __x, float __y, int *__quo) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __remquof (float __x, float __y,
      int *__quo) __attribute__ ((__nothrow__ , __leaf__));
2494
2495
2496
2497
2498
2499
2500 extern long int lrintf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     ); extern long int __lrintf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2501 __extension__
2502 extern long long int llrintf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__)); extern long long int __llrintf (float __x) __attribute__ ((
     __nothrow__ , __leaf__));
2503
2504
2505
```

```
2506 extern long int lroundf (float __x) __attribute__ ((__nothrow__ , __leaf__
     )); extern long int __lroundf (float __x) __attribute__ ((__nothrow__ ,
      __leaf__));
2507 __extension__
2508 extern long long int llroundf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__)); extern long long int __llroundf (float __x) __attribute__
     ((__nothrow__ , __leaf__));
2509
2510
2511
2512 extern float fdimf (float __x, float __y) __attribute__ ((__nothrow__ ,
     __leaf__)); extern float __fdimf (float __x, float __y) __attribute__
     ((__nothrow__ , __leaf__));
2513
2514
2515 extern float fmaxf (float __x, float __y) __attribute__ ((__nothrow__ ,
     __leaf__)) __attribute__ ((__const__)); extern float __fmaxf (float __x
     , float __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
     __const__));
2516
2517
2518 extern float fminf (float __x, float __y) __attribute__ ((__nothrow__ ,
     __leaf__)) __attribute__ ((__const__)); extern float __fminf (float __x
     , float __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
     __const__));
2519
2520
2521
2522 extern int __fpclassifyf (float __value) __attribute__ ((__nothrow__ ,
     __leaf__))
2523       __attribute__ ((__const__));
2524
2525
2526 extern int __signbitf (float __value) __attribute__ ((__nothrow__ ,
     __leaf__))
2527       __attribute__ ((__const__));
2528
2529
2530
2531 extern float fmaf (float __x, float __y, float __z) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __fmaf (float __x, float __y,
     float __z) __attribute__ ((__nothrow__ , __leaf__));
2532
2533
2534
2535
2536 # 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2537 extern float scalbf (float __x, float __n) __attribute__ ((__nothrow__ ,
     __leaf__)); extern float __scalbf (float __x, float __n) __attribute__
     ((__nothrow__ , __leaf__));
2538 # 89 "/usr/include/math.h" 2 3 4
2539 # 132 "/usr/include/math.h" 3 4
2540 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
2541 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
```

```
2542
2543
2544 extern long double acosl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __acosl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2545
2546 extern long double asinl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __asinl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2547
2548 extern long double atanl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __atanl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2549
2550 extern long double atan2l (long double __y, long double __x) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __atan2l (long double
       __y, long double __x) __attribute__ ((__nothrow__ , __leaf__));
2551
2552
2553 extern long double cosl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __cosl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2554
2555 extern long double sinl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __sinl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2556
2557 extern long double tanl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __tanl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2558
2559
2560
2561
2562 extern long double coshl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __coshl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2563
2564 extern long double sinhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __sinhl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2565
2566 extern long double tanhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __tanhl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2567
2568 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2569
2570
2571 extern long double acoshl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __acoshl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2572
```

```
2573 extern long double asinhl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __asinhl (long double __x) __attribute__
         ((__nothrow__ , __leaf__));
2574
2575 extern long double atanhl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __atanhl (long double __x) __attribute__
         ((__nothrow__ , __leaf__));
2576
2577
2578
2579
2580
2581
2582
2583 extern long double expl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __expl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2584
2585
2586 extern long double frexpl (long double __x, int *__exponent) __attribute__
         ((__nothrow__ , __leaf__)); extern long double __frexpl (long double
        __x, int *__exponent) __attribute__ ((__nothrow__ , __leaf__));
2587
2588
2589 extern long double ldexpl (long double __x, int __exponent) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __ldexpl (long double
        __x, int __exponent) __attribute__ ((__nothrow__ , __leaf__));
2590
2591
2592 extern long double logl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __logl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2593
2594
2595 extern long double log10l (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __log10l (long double __x) __attribute__
         ((__nothrow__ , __leaf__));
2596
2597
2598 extern long double modfl (long double __x, long double *__iptr)
        __attribute__ ((__nothrow__ , __leaf__)); extern long double __modfl (
        long double __x, long double *__iptr) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (2)));
2599
2600 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2601
2602
2603 extern long double expm1l (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __expm1l (long double __x) __attribute__
         ((__nothrow__ , __leaf__));
2604
2605
2606 extern long double log1pl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __log1pl (long double __x) __attribute__
```

```
    ((__nothrow__ , __leaf__));
2607
2608
2609 extern long double logbl (long double __x) __attribute__ ((__nothrow__ ,
    __leaf__)); extern long double __logbl (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2610
2611
2612
2613
2614
2615
2616 extern long double exp2l (long double __x) __attribute__ ((__nothrow__ ,
    __leaf__)); extern long double __exp2l (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2617
2618
2619 extern long double log2l (long double __x) __attribute__ ((__nothrow__ ,
    __leaf__)); extern long double __log2l (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2620
2621
2622
2623
2624
2625
2626
2627
2628 extern long double powl (long double __x, long double __y) __attribute__
    ((__nothrow__ , __leaf__)); extern long double __powl (long double __x,
     long double __y) __attribute__ ((__nothrow__ , __leaf__));
2629
2630
2631 extern long double sqrtl (long double __x) __attribute__ ((__nothrow__ ,
    __leaf__)); extern long double __sqrtl (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2632
2633
2634
2635
2636
2637 extern long double hypotl (long double __x, long double __y) __attribute__
    ((__nothrow__ , __leaf__)); extern long double __hypotl (long double
    __x, long double __y) __attribute__ ((__nothrow__ , __leaf__));
2638
2639
2640
2641
2642
2643
2644 extern long double cbrtl (long double __x) __attribute__ ((__nothrow__ ,
    __leaf__)); extern long double __cbrtl (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2645
```

```
2646
2647
2648
2649
2650
2651
2652
2653 extern long double ceill (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __ceill (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2654
2655
2656 extern long double fabsl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __fabsl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2657
2658
2659 extern long double floorl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __floorl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2660
2661
2662 extern long double fmodl (long double __x, long double __y) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __fmodl (long double __x
       , long double __y) __attribute__ ((__nothrow__ , __leaf__));
2663
2664
2665
2666
2667 extern int __isinfl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2668
2669
2670 extern int __finitel (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2671
2672
2673
2674
2675
2676 extern int isinfl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2677
2678
2679 extern int finitel (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2680
2681
2682 extern long double dreml (long double __x, long double __y) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __dreml (long double __x
       , long double __y) __attribute__ ((__nothrow__ , __leaf__));
```

```
2683
2684
2685
2686 extern long double significandl (long double __x) __attribute__ ((
       __nothrow__ , __leaf__)); extern long double __significandl (long
       double __x) __attribute__ ((__nothrow__ , __leaf__));
2687
2688
2689
2690
2691
2692 extern long double copysignl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
       extern long double __copysignl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2693
2694
2695
2696
2697
2698
2699 extern long double nanl (const char *__tagb) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern long double __nanl (
       const char *__tagb) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__));
2700
2701
2702
2703
2704
2705 extern int __isnanl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2706
2707
2708
2709 extern int isnanl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2710
2711
2712 extern long double j0l (long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __j0l (long double) __attribute__ ((
       __nothrow__ , __leaf__));
2713 extern long double j1l (long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __j1l (long double) __attribute__ ((
       __nothrow__ , __leaf__));
2714 extern long double jnl (int, long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __jnl (int, long double) __attribute__
       ((__nothrow__ , __leaf__));
2715 extern long double y0l (long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __y0l (long double) __attribute__ ((
       __nothrow__ , __leaf__));
2716 extern long double y1l (long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __y1l (long double) __attribute__ ((
       __nothrow__ , __leaf__));
```

```
2717 extern long double ynl (int, long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __ynl (int, long double) __attribute__
        ((__nothrow__ , __leaf__));
2718
2719
2720
2721
2722
2723
2724 extern long double erfl (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __erfl (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2725 extern long double erfcl (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __erfcl (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2726 extern long double lgammal (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __lgammal (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2727
2728
2729
2730
2731
2732
2733 extern long double tgammal (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __tgammal (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2734
2735
2736
2737
2738
2739 extern long double gammal (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __gammal (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2740
2741
2742
2743
2744
2745
2746 extern long double lgammal_r (long double, int *__signgamp) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __lgammal_r (long double
        , int *__signgamp) __attribute__ ((__nothrow__ , __leaf__));
2747
2748
2749
2750
2751
2752
2753
2754 extern long double rintl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __rintl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
```

```
2755
2756
2757 extern long double nextafterl (long double __x, long double __y)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
        extern long double __nextafterl (long double __x, long double __y)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2758
2759 extern long double nexttowardl (long double __x, long double __y)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
        extern long double __nexttowardl (long double __x, long double __y)
        __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2760
2761
2762
2763 extern long double remainderl (long double __x, long double __y)
        __attribute__ ((__nothrow__ , __leaf__)); extern long double
        __remainderl (long double __x, long double __y) __attribute__ ((
        __nothrow__ , __leaf__));
2764
2765
2766
2767 extern long double scalbnl (long double __x, int __n) __attribute__ ((
        __nothrow__ , __leaf__)); extern long double __scalbnl (long double __x
        , int __n) __attribute__ ((__nothrow__ , __leaf__));
2768
2769
2770
2771 extern int ilogbl (long double __x) __attribute__ ((__nothrow__ , __leaf__
        )); extern int __ilogbl (long double __x) __attribute__ ((__nothrow__ ,
         __leaf__));
2772
2773
2774
2775
2776 extern long double scalblnl (long double __x, long int __n) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __scalblnl (long double
        __x, long int __n) __attribute__ ((__nothrow__ , __leaf__));
2777
2778
2779
2780 extern long double nearbyintl (long double __x) __attribute__ ((
        __nothrow__ , __leaf__)); extern long double __nearbyintl (long double
        __x) __attribute__ ((__nothrow__ , __leaf__));
2781
2782
2783
2784 extern long double roundl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern long double __roundl (
        long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
         ((__const__));
2785
2786
2787
```

```
2788 extern long double truncl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern long double __truncl (
        long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
         ((__const__));
2789
2790
2791
2792
2793 extern long double remquol (long double __x, long double __y, int *__quo)
        __attribute__ ((__nothrow__ , __leaf__)); extern long double __remquol
        (long double __x, long double __y, int *__quo) __attribute__ ((
        __nothrow__ , __leaf__));
2794
2795
2796
2797
2798
2799
2800 extern long int lrintl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long int __lrintl (long double __x) __attribute__ ((
        __nothrow__ , __leaf__));
2801 __extension__
2802 extern long long int llrintl (long double __x) __attribute__ ((__nothrow__
         , __leaf__)); extern long long int __llrintl (long double __x)
        __attribute__ ((__nothrow__ , __leaf__));
2803
2804
2805
2806 extern long int lroundl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long int __lroundl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2807 __extension__
2808 extern long long int llroundl (long double __x) __attribute__ ((
        __nothrow__ , __leaf__)); extern long long int __llroundl (long double
        __x) __attribute__ ((__nothrow__ , __leaf__));
2809
2810
2811
2812 extern long double fdiml (long double __x, long double __y) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __fdiml (long double __x
        , long double __y) __attribute__ ((__nothrow__ , __leaf__));
2813
2814
2815 extern long double fmaxl (long double __x, long double __y) __attribute__
        ((__nothrow__ , __leaf__)) __attribute__ ((__const__)); extern long
        double __fmaxl (long double __x, long double __y) __attribute__ ((
        __nothrow__ , __leaf__)) __attribute__ ((__const__));
2816
2817
2818 extern long double fminl (long double __x, long double __y) __attribute__
        ((__nothrow__ , __leaf__)) __attribute__ ((__const__)); extern long
        double __fminl (long double __x, long double __y) __attribute__ ((
        __nothrow__ , __leaf__)) __attribute__ ((__const__));
2819
```

```
2820
2821
2822 extern int __fpclassifyl (long double __value) __attribute__ ((__nothrow__
         , __leaf__))
2823      __attribute__ ((__const__));
2824
2825
2826 extern int __signbitl (long double __value) __attribute__ ((__nothrow__ ,
        __leaf__))
2827      __attribute__ ((__const__));
2828
2829
2830
2831 extern long double fmal (long double __x, long double __y, long double __z
        ) __attribute__ ((__nothrow__ , __leaf__)); extern long double __fmal (
        long double __x, long double __y, long double __z) __attribute__ ((
        __nothrow__ , __leaf__));
2832
2833
2834
2835
2836 # 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2837 extern long double scalbl (long double __x, long double __n) __attribute__
         ((__nothrow__ , __leaf__)); extern long double __scalbl (long double
        __x, long double __n) __attribute__ ((__nothrow__ , __leaf__));
2838 # 133 "/usr/include/math.h" 2 3 4
2839 # 148 "/usr/include/math.h" 3 4
2840 extern int signgam;
2841 # 189 "/usr/include/math.h" 3 4
2842 enum
2843   {
2844     FP_NAN =
2845
2846       0,
2847     FP_INFINITE =
2848
2849       1,
2850     FP_ZERO =
2851
2852       2,
2853     FP_SUBNORMAL =
2854
2855       3,
2856     FP_NORMAL =
2857
2858       4
2859   };
2860 # 301 "/usr/include/math.h" 3 4
2861 typedef enum
2862 {
2863   _IEEE_ = -1,
2864   _SVID_,
2865   _XOPEN_,
2866   _POSIX_,
```

```
  _ISOC_
} _LIB_VERSION_TYPE;




extern _LIB_VERSION_TYPE _LIB_VERSION;
# 326 "/usr/include/math.h" 3 4
struct exception

  {
    int type;
    char *name;
    double arg1;
    double arg2;
    double retval;
  };




extern int matherr (struct exception *__exc);
# 488 "/usr/include/math.h" 3 4

# 7 "headers/all_headers.h" 2
# 1 "/usr/include/setjmp.h" 1 3 4
# 27 "/usr/include/setjmp.h" 3 4


# 1 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 1 3 4
# 26 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 2 3 4




typedef long int __jmp_buf[8];
# 30 "/usr/include/setjmp.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 1 3 4
# 31 "/usr/include/setjmp.h" 2 3 4


struct __jmp_buf_tag
  {




    __jmp_buf __jmpbuf;
    int __mask_was_saved;
    __sigset_t __saved_mask;
  };
```

```
2921
2922
2923
2924
2925 typedef struct __jmp_buf_tag jmp_buf[1];
2926
2927
2928
2929 extern int setjmp (jmp_buf __env) __attribute__ ((__nothrow__));
2930
2931
2932
2933
2934
2935
2936 extern int __sigsetjmp (struct __jmp_buf_tag __env[1], int __savemask)
        __attribute__ ((__nothrow__));
2937
2938
2939
2940 extern int _setjmp (struct __jmp_buf_tag __env[1]) __attribute__ ((
        __nothrow__));
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951 extern void longjmp (struct __jmp_buf_tag __env[1], int __val)
2952      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
2953
2954
2955
2956
2957
2958
2959
2960 extern void _longjmp (struct __jmp_buf_tag __env[1], int __val)
2961      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
2962
2963
2964
2965
2966
2967
2968
2969 typedef struct __jmp_buf_tag sigjmp_buf[1];
2970 # 102 "/usr/include/setjmp.h" 3 4
2971 extern void siglongjmp (sigjmp_buf __env, int __val)
2972      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
```

```
# 112 "/usr/include/setjmp.h" 3 4

# 8 "headers/all_headers.h" 2
# 1 "/usr/include/string.h" 1 3 4
# 27 "/usr/include/string.h" 3 4




# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 33 "/usr/include/string.h" 2 3 4
# 44 "/usr/include/string.h" 3 4


extern void *memcpy (void *__restrict __dest, const void *__restrict __src
    ,
        size_t __n) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
    ((__nonnull__ (1, 2)));


extern void *memmove (void *__dest, const void *__src, size_t __n)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));







extern void *memccpy (void *__restrict __dest, const void *__restrict
    __src,
         int __c, size_t __n)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));






extern void *memset (void *__s, int __c, size_t __n) __attribute__ ((
    __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));


extern int memcmp (const void *__s1, const void *__s2, size_t __n)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
# 96 "/usr/include/string.h" 3 4
extern void *memchr (const void *__s, int __c, size_t __n)
      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1)));


# 127 "/usr/include/string.h" 3 4
```

```
3019
3020
3021 extern char *strcpy (char *__restrict __dest, const char *__restrict __src
     )
3022      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (1, 2)));
3023
3024 extern char *strncpy (char *__restrict __dest,
3025         const char *__restrict __src, size_t __n)
3026      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (1, 2)));
3027
3028
3029 extern char *strcat (char *__restrict __dest, const char *__restrict __src
     )
3030      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (1, 2)));
3031
3032 extern char *strncat (char *__restrict __dest, const char *__restrict
     __src,
3033         size_t __n) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
      ((__nonnull__ (1, 2)));
3034
3035
3036 extern int strcmp (const char *__s1, const char *__s2)
3037      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3038
3039 extern int strncmp (const char *__s1, const char *__s2, size_t __n)
3040      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3041
3042
3043 extern int strcoll (const char *__s1, const char *__s2)
3044      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3045
3046 extern size_t strxfrm (char *__restrict __dest,
3047          const char *__restrict __src, size_t __n)
3048      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2)));
3049
3050
3051
3052
3053
3054
3055 # 1 "/usr/include/xlocale.h" 1 3 4
3056 # 27 "/usr/include/xlocale.h" 3 4
3057 typedef struct __locale_struct
3058 {
3059
3060   struct __locale_data *__locales[13];
3061
```

```
3062
3063   const unsigned short int *__ctype_b;
3064   const int *__ctype_tolower;
3065   const int *__ctype_toupper;
3066
3067
3068   const char *__names[13];
3069 } *__locale_t;
3070
3071
3072 typedef __locale_t locale_t;
3073 # 164 "/usr/include/string.h" 2 3 4
3074
3075
3076 extern int strcoll_l (const char *__s1, const char *__s2, __locale_t __l)
3077      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2, 3)));
3078
3079 extern size_t strxfrm_l (char *__dest, const char *__src, size_t __n,
3080      __locale_t __l) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
      ((__nonnull__ (2, 4)));
3081
3082
3083
3084
3085
3086 extern char *strdup (const char *__s)
3087      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
      __attribute__ ((__nonnull__ (1)));
3088
3089
3090
3091
3092
3093
3094 extern char *strndup (const char *__string, size_t __n)
3095      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
      __attribute__ ((__nonnull__ (1)));
3096 # 211 "/usr/include/string.h" 3 4
3097
3098 # 236 "/usr/include/string.h" 3 4
3099 extern char *strchr (const char *__s, int __c)
3100      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1)));
3101 # 263 "/usr/include/string.h" 3 4
3102 extern char *strrchr (const char *__s, int __c)
3103      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1)));
3104
3105
3106 # 282 "/usr/include/string.h" 3 4
3107
3108
3109
```

```
3110 extern size_t strcspn (const char *__s, const char *__reject)
3111      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3112
3113
3114 extern size_t strspn (const char *__s, const char *__accept)
3115      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3116 # 315 "/usr/include/string.h" 3 4
3117 extern char *strpbrk (const char *__s, const char *__accept)
3118      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3119 # 342 "/usr/include/string.h" 3 4
3120 extern char *strstr (const char *__haystack, const char *__needle)
3121      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1, 2)));
3122
3123
3124
3125
3126 extern char *strtok (char *__restrict __s, const char *__restrict __delim)
3127      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2)));
3128
3129
3130
3131
3132 extern char *__strtok_r (char *__restrict __s,
3133     const char *__restrict __delim,
3134     char **__restrict __save_ptr)
3135      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2, 3)));
3136
3137 extern char *strtok_r (char *__restrict __s, const char *__restrict
     __delim,
3138           char **__restrict __save_ptr)
3139      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2, 3)));
3140 # 397 "/usr/include/string.h" 3 4
3141
3142
3143 extern size_t strlen (const char *__s)
3144      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1)));
3145
3146
3147
3148
3149
3150 extern size_t strnlen (const char *__string, size_t __maxlen)
3151      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
     __attribute__ ((__nonnull__ (1)));
3152
3153
```

```
3154
3155
3156
3157 extern char *strerror (int __errnum) __attribute__ ((__nothrow__ ,
       __leaf__));
3158
3159 # 427 "/usr/include/string.h" 3 4
3160 extern int strerror_r (int __errnum, char *__buf, size_t __buflen) __asm__
       ("" "__xpg_strerror_r") __attribute__ ((__nothrow__ , __leaf__))
3161
3162                         __attribute__ ((__nonnull__ (2)));
3163 # 445 "/usr/include/string.h" 3 4
3164 extern char *strerror_l (int __errnum, __locale_t __l) __attribute__ ((
       __nothrow__ , __leaf__));
3165
3166
3167
3168
3169
3170 extern void __bzero (void *__s, size_t __n) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
3171
3172
3173
3174 extern void bcopy (const void *__src, void *__dest, size_t __n)
3175      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3176
3177
3178 extern void bzero (void *__s, size_t __n) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
3179
3180
3181 extern int bcmp (const void *__s1, const void *__s2, size_t __n)
3182      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1, 2)));
3183 # 489 "/usr/include/string.h" 3 4
3184 extern char *index (const char *__s, int __c)
3185      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1)));
3186 # 517 "/usr/include/string.h" 3 4
3187 extern char *rindex (const char *__s, int __c)
3188      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1)));
3189
3190
3191
3192
3193 extern int ffs (int __i) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__));
3194 # 534 "/usr/include/string.h" 3 4
3195 extern int strcasecmp (const char *__s1, const char *__s2)
3196      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1, 2)));
```

```
3197
3198
3199 extern int strncasecmp (const char *__s1, const char *__s2, size_t __n)
3200      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3201 # 557 "/usr/include/string.h" 3 4
3202 extern char *strsep (char **__restrict __stringp,
3203        const char *__restrict __delim)
3204      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3205
3206
3207
3208
3209 extern char *strsignal (int __sig) __attribute__ ((__nothrow__ , __leaf__)
       );
3210
3211
3212 extern char *__stpcpy (char *__restrict __dest, const char *__restrict
       __src)
3213      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3214 extern char *stpcpy (char *__restrict __dest, const char *__restrict __src
       )
3215      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3216
3217
3218
3219 extern char *__stpncpy (char *__restrict __dest,
3220        const char *__restrict __src, size_t __n)
3221      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3222 extern char *stpncpy (char *__restrict __dest,
3223         const char *__restrict __src, size_t __n)
3224      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
3225 # 644 "/usr/include/string.h" 3 4
3226
3227 # 9 "headers/all_headers.h" 2
3228 # 1 "headers/sen_basic_type.h" 1
3229 # 1 "headers/sen_object.h" 1
3230
3231
3232
3233
3234
3235
3236
3237 # 7 "headers/sen_object.h"
3238 struct Sen_object_vtable;
3239 typedef struct Sen_object_vtable Sen_object_vtable;
3240
3241 struct Sen_object_class;
```

```
typedef struct Sen_object_class Sen_object_class;

struct Sen_object;
typedef struct Sen_object Sen_object;

struct Sen_object_vtable {
    void (*print) (Sen_object *);
    Sen_object *(*construct) (void *);
    void (*destruct) (Sen_object *);
    Sen_object *(*copy) (Sen_object *);
};



struct Sen_object_class {
    Sen_object_vtable *tablep;
};



struct Sen_object {

# 32 "headers/sen_object.h" 3 4
   _Bool
# 32 "headers/sen_object.h"
        bound;
    Sen_object_class *classp;
};

extern Sen_object_class Sen_object_class_;
extern Sen_object_vtable Sen_object_vtable_;

void print_object (Sen_object *);
Sen_object * construct_object (void *);
void destruct_object (Sen_object *);
Sen_object * copy_object (Sen_object *);
# 2 "headers/sen_basic_type.h" 2




struct Sen_basic_type_vtable;
typedef struct Sen_basic_type_vtable Sen_basic_type_vtable;

struct Sen_basic_type_class;
typedef struct Sen_basic_type_class Sen_basic_type_class;

struct Sen_basic_type;
typedef struct Sen_basic_type Sen_basic_type;

typedef enum {BOOL, INT, STR, UNK} Type;

struct Sen_basic_type_vtable {
    void (*print) (Sen_object *);
```

```
    Sen_basic_type *(*construct) (void *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_basic_type_class {
    Sen_object_class *superp;
    Sen_basic_type_vtable *tablep;
    Type type;
};

struct Sen_basic_type {

# 32 "headers/sen_basic_type.h" 3 4
   _Bool
# 32 "headers/sen_basic_type.h"
        bound;
    Sen_basic_type_class *classp;
    Sen_object *superp;
};

extern Sen_basic_type_class Sen_basic_type_class_;
extern Sen_basic_type_vtable Sen_basic_type_vtable_;

void * get_val_basic_type (Sen_basic_type *);
void * set_val_basic_type (Sen_basic_type *, void *);
Sen_basic_type * add_basic_type (Sen_basic_type *, Sen_basic_type *);
# 10 "headers/all_headers.h" 2
# 1 "headers/sen_int.h" 1
# 1 "headers/sen_basic_type.h" 1
# 2 "headers/sen_int.h" 2




struct Sen_int_vtable;
typedef struct Sen_int_vtable Sen_int_vtable;

struct Sen_int_class;
typedef struct Sen_int_class Sen_int_class;

struct Sen_int;
typedef struct Sen_int Sen_int;

struct Sen_int_vtable {
    void (*print) (Sen_object *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_int *(*construct) (int);
    void (*destruct) (Sen_int *);
    Sen_int *(*copy) (Sen_int *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};
```

```
3350
3351 struct Sen_int_class {
3352     Sen_basic_type_class *superp;
3353     Sen_int_vtable *tablep;
3354     Type type;
3355 };
3356
3357 struct Sen_int {
3358
3359 # 32 "headers/sen_int.h" 3 4
3360    _Bool
3361 # 32 "headers/sen_int.h"
3362         bound;
3363     Sen_int_class *classp;
3364     Sen_basic_type *superp;
3365     int val;
3366 };
3367
3368 extern Sen_int_class Sen_int_class_;
3369 extern Sen_int_vtable Sen_int_vtable_;
3370
3371 void print_int (Sen_object *);
3372 Sen_int * construct_int (int);
3373 void *get_val_int (Sen_basic_type *);
3374 void *set_val_int (Sen_basic_type *, void *);
3375 # 11 "headers/all_headers.h" 2
3376
3377 # 1 "headers/sen_bool.h" 1
3378 # 1 "headers/sen_basic_type.h" 1
3379 # 2 "headers/sen_bool.h" 2
3380
3381
3382
3383
3384 struct Sen_bool_vtable;
3385 typedef struct Sen_bool_vtable Sen_bool_vtable;
3386
3387 struct Sen_bool_class;
3388 typedef struct Sen_bool_class Sen_bool_class;
3389
3390 struct Sen_bool;
3391 typedef struct Sen_bool Sen_bool;
3392
3393 struct Sen_bool_vtable {
3394     void (*print) (Sen_object *);
3395     void *(*get_val) (Sen_basic_type *);
3396     void *(*set_val) (Sen_basic_type *, void *);
3397     Sen_bool *(*construct) (
3398 # 19 "headers/sen_bool.h" 3 4
3399                             _Bool
3400 # 19 "headers/sen_bool.h"
3401                                  );
3402     void (*destruct) (Sen_bool *);
3403     Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
```

```
};

struct Sen_bool_class {
    Sen_basic_type_class *superp;
    Sen_bool_vtable *tablep;
    Type type;
};

struct Sen_bool {

# 31 "headers/sen_bool.h" 3 4
   _Bool
# 31 "headers/sen_bool.h"
        bound;
    Sen_bool_class *classp;
    Sen_basic_type *superp;

# 34 "headers/sen_bool.h" 3 4
   _Bool
# 34 "headers/sen_bool.h"
        val;
};

extern Sen_bool_class Sen_bool_class_;
extern Sen_bool_vtable Sen_bool_vtable_;

void print_bool (Sen_object *);
Sen_bool * construct_bool (
# 41 "headers/sen_bool.h" 3 4
                           _Bool
# 41 "headers/sen_bool.h"
                                );
void *get_val_bool (Sen_basic_type *);
void *set_val_bool (Sen_basic_type *, void *);
# 13 "headers/all_headers.h" 2
# 1 "headers/sen_string.h" 1
# 1 "headers/sen_basic_type.h" 1
# 2 "headers/sen_string.h" 2




struct Sen_string_vtable;
typedef struct Sen_string_vtable Sen_string_vtable;

struct Sen_string_class;
typedef struct Sen_string_class Sen_string_class;

struct Sen_string;
typedef struct Sen_string Sen_string;

struct Sen_string_vtable {
    void (*print) (Sen_object *);
    Sen_string *(*construct) (char *);
```

```
3458     void (*destruct) (Sen_string *);
3459     Sen_string *(*copy) (Sen_string *);
3460     void *(*get_val) (Sen_basic_type *);
3461     void *(*set_val) (Sen_basic_type *, void *);
3462     Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
3463 };
3464 
3465 struct Sen_string_class {
3466     Sen_basic_type_class *superp;
3467     Sen_string_vtable *tablep;
3468     Type type;
3469 };
3470 
3471 struct Sen_string {
3472 
3473 # 32 "headers/sen_string.h" 3 4
3474    _Bool
3475 # 32 "headers/sen_string.h"
3476         bound;
3477     Sen_string_class *classp;
3478     Sen_basic_type *superp;
3479     char *val;
3480 };
3481 
3482 extern Sen_string_class Sen_string_class_;
3483 extern Sen_string_vtable Sen_string_vtable_;
3484 
3485 void print_string (Sen_object *);
3486 Sen_string * construct_string (char *);
3487 void destruct (Sen_string *);
3488 Sen_string * copy_string (Sen_string *);
3489 void *get_val_string (Sen_basic_type *);
3490 void *set_val_string (Sen_basic_type *, void *);
3491 # 14 "headers/all_headers.h" 2
3492 # 1 "headers/sen_array.h" 1
3493 
3494 # 1 "headers/sen_basic_type.h" 1
3495 # 3 "headers/sen_array.h" 2
3496 # 1 "headers/sen_int.h" 1
3497 # 1 "headers/sen_basic_type.h" 1
3498 # 2 "headers/sen_int.h" 2
3499 # 4 "headers/sen_array.h" 2
3500 # 12 "headers/sen_array.h"
3501 struct Sen_array_vtable;
3502 typedef struct Sen_array_vtable Sen_array_vtable;
3503 
3504 struct Sen_array_class;
3505 typedef struct Sen_array_class Sen_array_class;
3506 
3507 struct Sen_array;
3508 typedef struct Sen_array Sen_array;
3509 
3510 struct Sen_array_vtable {
3511     void (*print) (Sen_object *);
```

```
3512     Sen_array *(*construct) (Sen_object **, int);
3513     void (*destruct) (Sen_array *);
3514     Sen_object *(*access) (Sen_array *, Sen_int *);
3515     Sen_array *(*concat) (Sen_array *, Sen_array *);
3516 };
3517 
3518 struct Sen_array_class {
3519     Sen_object_class *superp;
3520     Sen_array_vtable *tablep;
3521 };
3522 
3523 struct Sen_array {
3524 
3525 # 35 "headers/sen_array.h" 3 4
3526    _Bool
3527 # 35 "headers/sen_array.h"
3528         bound;
3529     Sen_array_class *classp;
3530     Sen_object *superp;
3531     void **arr;
3532     int len;
3533     char print_sep;
3534 };
3535 
3536 extern Sen_array_class Sen_array_class_;
3537 extern Sen_array_vtable Sen_array_vtable_;
3538 
3539 Sen_array *construct_array (Sen_object **, int);
3540 Sen_object *access_array (Sen_array *, Sen_int *);
3541 Sen_array *concat_array (Sen_array *, Sen_array *);
3542 # 15 "headers/all_headers.h" 2
3543 # 2 "main.c" 2
3544 
3545 int main() {
3546 # 29 "main.c"
3547     __auto_type s = ((Sen_string*) construct_string("tttesting "));
3548     s->bound=
3549 # 30 "main.c" 3 4
3550             1
3551 # 30 "main.c"
3552                 ;
3553     __auto_type ss = ((Sen_string*) construct_string("hooray!!\n"));
3554     ss->bound=
3555 # 32 "main.c" 3 4
3556              1
3557 # 32 "main.c"
3558                  ;
3559 
3560 
3561     { typeof(((Sen_string *) ss)) __temp__ = ((Sen_string *) ss); __temp__
    ->classp->tablep->print(((Sen_object *) __temp__)); };
3562     { typeof(((Sen_string *)({ __auto_type __temp__ = s; __temp__->classp
    ->tablep->add((Sen_basic_type *)__temp__, (Sen_basic_type *)ss); })))
    __temp__ = ((Sen_string *)({ __auto_type __temp__ = s; __temp__->classp
```

```
    ->tablep->add((Sen_basic_type *)__temp__, (Sen_basic_type *)ss); }));
    __temp__->classp->tablep->print(((Sen_object *) __temp__)); };
3563     { typeof(((Sen_string *)({ __auto_type __temp__ = ((Sen_string *)({
    __auto_type __temp__ = s; __temp__->classp->tablep->add((Sen_basic_type
     *)__temp__, (Sen_basic_type *)ss); })); __temp__->classp->tablep->add
    ((Sen_basic_type *)__temp__, (Sen_basic_type *)((Sen_string *)({
    __auto_type __temp__ = s; __temp__->classp->tablep->add((Sen_basic_type
     *)__temp__, (Sen_basic_type *)ss); }))); })))  __temp__ = ((Sen_string
    *)({ __auto_type __temp__ = ((Sen_string *)({ __auto_type __temp__ = s;
     __temp__->classp->tablep->add((Sen_basic_type *)__temp__, (
    Sen_basic_type *)ss); })); __temp__->classp->tablep->add((
    Sen_basic_type *)__temp__, (Sen_basic_type *)((Sen_string *)({
    __auto_type __temp__ = s; __temp__->classp->tablep->add((Sen_basic_type
     *)__temp__, (Sen_basic_type *)ss); }))); })); __temp__->classp->tablep
    ->print(((Sen_object *) __temp__)); };
3564     ((Sen_string *) s)->classp->tablep->destruct(((Sen_string *) s));
3565     ((Sen_string *) ss)->classp->tablep->destruct(((Sen_string *) ss));
3566     Sen_int *arr_[2] = {((Sen_int*) construct_int(100)), ((Sen_int*)
    construct_int(50))};
3567     printf ("%d %d\n", (int)sizeof(arr_), arr_[0]->bound);
3568     Sen_array *arr = (Sen_array_vtable_.construct(arr_, (sizeof((arr_)) /
    sizeof((arr_)[0]))));
3569     printf ("%d %d\n", (int)sizeof(arr_), arr_[0]->bound);
3570     printf("%d %d %d\n", (int)((arr->arr)[0]), (sizeof((arr_)) / sizeof((
    arr_)[0])), arr_[0]->bound);
3571
3572     return 0;
3573 }
```

./c_files/test

```
1  # 1 "main.c"
2  # 1 "<built-in>"
3  # 1 "<command-line>"
4  # 1 "/usr/include/stdc-predef.h" 1 3 4
5  # 1 "<command-line>" 2
6  # 1 "main.c"
7  # 1 "headers/all_headers.h" 1
8
9
10 # 1 "/usr/include/stdlib.h" 1 3 4
11 # 24 "/usr/include/stdlib.h" 3 4
12 # 1 "/usr/include/features.h" 1 3 4
13 # 374 "/usr/include/features.h" 3 4
14 # 1 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 1 3 4
15 # 385 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 3 4
16 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
17 # 386 "/usr/include/x86_64-linux-gnu/sys/cdefs.h" 2 3 4
18 # 375 "/usr/include/features.h" 2 3 4
19 # 398 "/usr/include/features.h" 3 4
20 # 1 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 1 3 4
21 # 10 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 3 4
22 # 1 "/usr/include/x86_64-linux-gnu/gnu/stubs-64.h" 1 3 4
23 # 11 "/usr/include/x86_64-linux-gnu/gnu/stubs.h" 2 3 4
24 # 399 "/usr/include/features.h" 2 3 4
```

```
# 25 "/usr/include/stdlib.h" 2 3 4
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 216 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4
# 216 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4
typedef long unsigned int size_t;
# 328 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 3 4
typedef int wchar_t;
# 33 "/usr/include/stdlib.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/waitflags.h" 1 3 4
# 50 "/usr/include/x86_64-linux-gnu/bits/waitflags.h" 3 4
typedef enum
{
  P_ALL,
  P_PID,
  P_PGID
} idtype_t;
# 42 "/usr/include/stdlib.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 1 3 4
# 64 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 3 4
# 1 "/usr/include/endian.h" 1 3 4
# 36 "/usr/include/endian.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/endian.h" 1 3 4
# 37 "/usr/include/endian.h" 2 3 4
# 60 "/usr/include/endian.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/types.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 28 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned char __u_char;
typedef unsigned short int __u_short;
typedef unsigned int __u_int;
typedef unsigned long int __u_long;


```

```
typedef signed char __int8_t;
typedef unsigned char __uint8_t;
typedef signed short int __int16_t;
typedef unsigned short int __uint16_t;
typedef signed int __int32_t;
typedef unsigned int __uint32_t;

typedef signed long int __int64_t;
typedef unsigned long int __uint64_t;









typedef long int __quad_t;
typedef unsigned long int __u_quad_t;
# 121 "/usr/include/x86_64-linux-gnu/bits/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/typesizes.h" 1 3 4
# 122 "/usr/include/x86_64-linux-gnu/bits/types.h" 2 3 4


typedef unsigned long int __dev_t;
typedef unsigned int __uid_t;
typedef unsigned int __gid_t;
typedef unsigned long int __ino_t;
typedef unsigned long int __ino64_t;
typedef unsigned int __mode_t;
typedef unsigned long int __nlink_t;
typedef long int __off_t;
typedef long int __off64_t;
typedef int __pid_t;
typedef struct { int __val[2]; } __fsid_t;
typedef long int __clock_t;
typedef unsigned long int __rlim_t;
typedef unsigned long int __rlim64_t;
typedef unsigned int __id_t;
typedef long int __time_t;
typedef unsigned int __useconds_t;
typedef long int __suseconds_t;

typedef int __daddr_t;
typedef int __key_t;


typedef int __clockid_t;


typedef void * __timer_t;


typedef long int __blksize_t;

```

```
133
134
135
136 typedef long int __blkcnt_t;
137 typedef long int __blkcnt64_t;
138
139
140 typedef unsigned long int __fsblkcnt_t;
141 typedef unsigned long int __fsblkcnt64_t;
142
143
144 typedef unsigned long int __fsfilcnt_t;
145 typedef unsigned long int __fsfilcnt64_t;
146
147
148 typedef long int __fsword_t;
149
150 typedef long int __ssize_t;
151
152
153 typedef long int __syscall_slong_t;
154
155 typedef unsigned long int __syscall_ulong_t;
156
157
158
159 typedef __off64_t __loff_t;
160 typedef __quad_t *__qaddr_t;
161 typedef char *__caddr_t;
162
163
164 typedef long int __intptr_t;
165
166
167 typedef unsigned int __socklen_t;
168 # 28 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4
169 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
170 # 29 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4
171
172
173
174
175
176
177 # 1 "/usr/include/x86_64-linux-gnu/bits/byteswap-16.h" 1 3 4
178 # 36 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 2 3 4
179 # 44 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
180 static __inline unsigned int
181 __bswap_32 (unsigned int __bsx)
182 {
183   return __builtin_bswap32 (__bsx);
184 }
185 # 108 "/usr/include/x86_64-linux-gnu/bits/byteswap.h" 3 4
186 static __inline __uint64_t
```

```
__bswap_64 (__uint64_t __bsx)
{
  return __builtin_bswap64 (__bsx);
}
# 61 "/usr/include/endian.h" 2 3 4
# 65 "/usr/include/x86_64-linux-gnu/bits/waitstatus.h" 2 3 4

union wait
  {
    int w_status;
    struct
      {

 unsigned int __w_termsig:7;
 unsigned int __w_coredump:1;
 unsigned int __w_retcode:8;
 unsigned int:16;









      } __wait_terminated;
    struct
      {

 unsigned int __w_stopval:8;
 unsigned int __w_stopsig:8;
 unsigned int:16;









      } __wait_stopped;
  };
# 43 "/usr/include/stdlib.h" 2 3 4
# 67 "/usr/include/stdlib.h" 3 4
typedef union
  {
    union wait *__uptr;
    int *__iptr;
  } __WAIT_STATUS __attribute__ ((__transparent_union__));
# 95 "/usr/include/stdlib.h" 3 4


typedef struct
  {
    int quot;
    int rem;
  } div_t;
```

```
241
242
243
244 typedef struct
245   {
246     long int quot;
247     long int rem;
248   } ldiv_t;
249
250
251
252
253
254
255
256 __extension__ typedef struct
257   {
258     long long int quot;
259     long long int rem;
260   } lldiv_t;
261
262
263 # 139 "/usr/include/stdlib.h" 3 4
264 extern size_t __ctype_get_mb_cur_max (void) __attribute__ ((__nothrow__ ,
       __leaf__)) ;
265
266
267
268
269 extern double atof (const char *__nptr)
270      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1))) ;
271
272 extern int atoi (const char *__nptr)
273      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1))) ;
274
275 extern long int atol (const char *__nptr)
276      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1))) ;
277
278
279
280
281
282 __extension__ extern long long int atoll (const char *__nptr)
283      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
       __attribute__ ((__nonnull__ (1))) ;
284
285
286
287
288
289 extern double strtod (const char *__restrict __nptr,
```

```
        char **__restrict __endptr)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));




extern float strtof (const char *__restrict __nptr,
       char **__restrict __endptr) __attribute__ ((__nothrow__ , __leaf__)
   ) __attribute__ ((__nonnull__ (1)));

extern long double strtold (const char *__restrict __nptr,
       char **__restrict __endptr)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));




extern long int strtol (const char *__restrict __nptr,
   char **__restrict __endptr, int __base)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));

extern unsigned long int strtoul (const char *__restrict __nptr,
      char **__restrict __endptr, int __base)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));




__extension__
extern long long int strtoq (const char *__restrict __nptr,
        char **__restrict __endptr, int __base)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));

__extension__
extern unsigned long long int strtouq (const char *__restrict __nptr,
           char **__restrict __endptr, int __base)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));




__extension__
extern long long int strtoll (const char *__restrict __nptr,
         char **__restrict __endptr, int __base)
```

```
336      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
337
338 __extension__
339 extern unsigned long long int strtoull (const char *__restrict __nptr,
340      char **__restrict __endptr, int __base)
341      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
342
343 # 305 "/usr/include/stdlib.h" 3 4
344 extern char *l64a (long int __n) __attribute__ ((__nothrow__ , __leaf__))
    ;
345
346
347 extern long int a64l (const char *__s)
348      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1))) ;
349
350
351
352
353 # 1 "/usr/include/x86_64-linux-gnu/sys/types.h" 1 3 4
354 # 27 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
355
356
357
358
359
360
361 typedef __u_char u_char;
362 typedef __u_short u_short;
363 typedef __u_int u_int;
364 typedef __u_long u_long;
365 typedef __quad_t quad_t;
366 typedef __u_quad_t u_quad_t;
367 typedef __fsid_t fsid_t;
368
369
370
371
372 typedef __loff_t loff_t;
373
374
375
376 typedef __ino_t ino_t;
377 # 60 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
378 typedef __dev_t dev_t;
379
380
381
382
383 typedef __gid_t gid_t;
384
385
```

```
386
387
388 typedef __mode_t mode_t;
389
390
391
392
393 typedef __nlink_t nlink_t;
394
395
396
397
398 typedef __uid_t uid_t;
399
400
401
402
403
404 typedef __off_t off_t;
405 # 98 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
406 typedef __pid_t pid_t;
407
408
409
410
411
412 typedef __id_t id_t;
413
414
415
416
417 typedef __ssize_t ssize_t;
418
419
420
421
422
423 typedef __daddr_t daddr_t;
424 typedef __caddr_t caddr_t;
425
426
427
428
429
430 typedef __key_t key_t;
431 # 132 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
432 # 1 "/usr/include/time.h" 1 3 4
433 # 57 "/usr/include/time.h" 3 4
434
435
436 typedef __clock_t clock_t;
437
438
439
```

```
# 73 "/usr/include/time.h" 3 4


typedef __time_t time_t;



# 91 "/usr/include/time.h" 3 4
typedef __clockid_t clockid_t;
# 103 "/usr/include/time.h" 3 4
typedef __timer_t timer_t;
# 133 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
# 146 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 147 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4



typedef unsigned long int ulong;
typedef unsigned short int ushort;
typedef unsigned int uint;
# 194 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
typedef int int8_t __attribute__ ((__mode__ (__QI__)));
typedef int int16_t __attribute__ ((__mode__ (__HI__)));
typedef int int32_t __attribute__ ((__mode__ (__SI__)));
typedef int int64_t __attribute__ ((__mode__ (__DI__)));


typedef unsigned int u_int8_t __attribute__ ((__mode__ (__QI__)));
typedef unsigned int u_int16_t __attribute__ ((__mode__ (__HI__)));
typedef unsigned int u_int32_t __attribute__ ((__mode__ (__SI__)));
typedef unsigned int u_int64_t __attribute__ ((__mode__ (__DI__)));

typedef int register_t __attribute__ ((__mode__ (__word__)));
# 219 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/sys/select.h" 1 3 4
# 30 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/select.h" 1 3 4
# 22 "/usr/include/x86_64-linux-gnu/bits/select.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 23 "/usr/include/x86_64-linux-gnu/bits/select.h" 2 3 4
# 31 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4


# 1 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 1 3 4
# 22 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 3 4
typedef int __sig_atomic_t;



typedef struct
  {
    unsigned long int __val[(1024 / (8 * sizeof (unsigned long int)))];
```

```
  } __sigset_t;
# 34 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4



typedef __sigset_t sigset_t;
# 1 "/usr/include/time.h" 1 3 4
# 120 "/usr/include/time.h" 3 4
struct timespec
  {
    __time_t tv_sec;
    __syscall_slong_t tv_nsec;
  };
# 44 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4

# 1 "/usr/include/x86_64-linux-gnu/bits/time.h" 1 3 4
# 30 "/usr/include/x86_64-linux-gnu/bits/time.h" 3 4
struct timeval
  {
    __time_t tv_sec;
    __suseconds_t tv_usec;
  };
# 46 "/usr/include/x86_64-linux-gnu/sys/select.h" 2 3 4


typedef __suseconds_t suseconds_t;



typedef long int __fd_mask;
# 64 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
typedef struct
  {




    __fd_mask __fds_bits[1024 / (8 * (int) sizeof (__fd_mask))];


  } fd_set;
```

```
548
549
550 typedef __fd_mask fd_mask;
551 # 96 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
552
553 # 106 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
554 extern int select (int __nfds, fd_set *__restrict __readfds,
555      fd_set *__restrict __writefds,
556      fd_set *__restrict __exceptfds,
557      struct timeval *__restrict __timeout);
558 # 118 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
559 extern int pselect (int __nfds, fd_set *__restrict __readfds,
560       fd_set *__restrict __writefds,
561       fd_set *__restrict __exceptfds,
562       const struct timespec *__restrict __timeout,
563       const __sigset_t *__restrict __sigmask);
564 # 131 "/usr/include/x86_64-linux-gnu/sys/select.h" 3 4
565
566 # 220 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
567
568
569 # 1 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 1 3 4
570 # 24 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 3 4
571
572
573 __extension__
574 extern unsigned int gnu_dev_major (unsigned long long int __dev)
575      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
576 __extension__
577 extern unsigned int gnu_dev_minor (unsigned long long int __dev)
578      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
579 __extension__
580 extern unsigned long long int gnu_dev_makedev (unsigned int __major,
581             unsigned int __minor)
582      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
583 # 58 "/usr/include/x86_64-linux-gnu/sys/sysmacros.h" 3 4
584
585 # 223 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4
586
587
588
589
590
591 typedef __blksize_t blksize_t;
592
593
594
595
596
597
598 typedef __blkcnt_t blkcnt_t;
599
600
601
```

```
602 typedef __fsblkcnt_t fsblkcnt_t;
603
604
605
606 typedef __fsfilcnt_t fsfilcnt_t;
607 # 270 "/usr/include/x86_64-linux-gnu/sys/types.h" 3 4
608 # 1 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 1 3 4
609 # 21 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
610 # 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
611 # 22 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 2 3 4
612 # 60 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
613 typedef unsigned long int pthread_t;
614
615
616 union pthread_attr_t
617 {
618   char __size[56];
619   long int __align;
620 };
621
622 typedef union pthread_attr_t pthread_attr_t;
623
624
625
626
627
628 typedef struct __pthread_internal_list
629 {
630   struct __pthread_internal_list *__prev;
631   struct __pthread_internal_list *__next;
632 } __pthread_list_t;
633 # 90 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
634 typedef union
635 {
636   struct __pthread_mutex_s
637   {
638     int __lock;
639     unsigned int __count;
640     int __owner;
641
642     unsigned int __nusers;
643
644
645
646     int __kind;
647
648     short __spins;
649     short __elision;
650     __pthread_list_t __list;
651 # 124 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
652   } __data;
653   char __size[40];
654   long int __align;
655 } pthread_mutex_t;
```

```
656
657 typedef union
658 {
659   char __size[4];
660   int __align;
661 } pthread_mutexattr_t;
662
663
664
665
666 typedef union
667 {
668   struct
669   {
670     int __lock;
671     unsigned int __futex;
672     __extension__ unsigned long long int __total_seq;
673     __extension__ unsigned long long int __wakeup_seq;
674     __extension__ unsigned long long int __woken_seq;
675     void *__mutex;
676     unsigned int __nwaiters;
677     unsigned int __broadcast_seq;
678   } __data;
679   char __size[48];
680   __extension__ long long int __align;
681 } pthread_cond_t;
682
683 typedef union
684 {
685   char __size[4];
686   int __align;
687 } pthread_condattr_t;
688
689
690
691 typedef unsigned int pthread_key_t;
692
693
694
695 typedef int pthread_once_t;
696
697
698
699
700
701 typedef union
702 {
703
704   struct
705   {
706     int __lock;
707     unsigned int __nr_readers;
708     unsigned int __readers_wakeup;
709     unsigned int __writer_wakeup;
```

```
    unsigned int __nr_readers_queued;
    unsigned int __nr_writers_queued;
    int __writer;
    int __shared;
    unsigned long int __pad1;
    unsigned long int __pad2;


    unsigned int __flags;

  } __data;
# 211 "/usr/include/x86_64-linux-gnu/bits/pthreadtypes.h" 3 4
  char __size[56];
  long int __align;
} pthread_rwlock_t;

typedef union
{
  char __size[8];
  long int __align;
} pthread_rwlockattr_t;




typedef volatile int pthread_spinlock_t;



typedef union
{
  char __size[32];
  long int __align;
} pthread_barrier_t;

typedef union
{
  char __size[4];
  int __align;
} pthread_barrierattr_t;
# 271 "/usr/include/x86_64-linux-gnu/sys/types.h" 2 3 4



# 315 "/usr/include/stdlib.h" 2 3 4







extern long int random (void) __attribute__ ((__nothrow__ , __leaf__));
```

```
764
765
766 extern void srandom (unsigned int __seed) __attribute__ ((__nothrow__ ,
       __leaf__));
767
768
769
770
771
772 extern char *initstate (unsigned int __seed, char *__statebuf,
773    size_t __statelen) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__nonnull__ (2)));
774
775
776
777 extern char *setstate (char *__statebuf) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
778
779
780
781
782
783
784
785 struct random_data
786   {
787     int32_t *fptr;
788     int32_t *rptr;
789     int32_t *state;
790     int rand_type;
791     int rand_deg;
792     int rand_sep;
793     int32_t *end_ptr;
794   };
795
796 extern int random_r (struct random_data *__restrict __buf,
797        int32_t *__restrict __result) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
798
799 extern int srandom_r (unsigned int __seed, struct random_data *__buf)
800      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (2)));
801
802 extern int initstate_r (unsigned int __seed, char *__restrict __statebuf,
803    size_t __statelen,
804    struct random_data *__restrict __buf)
805      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (2, 4)));
806
807 extern int setstate_r (char *__restrict __statebuf,
808          struct random_data *__restrict __buf)
809      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1, 2)));
810
```

```
811
812
813
814
815
816 extern int rand (void) __attribute__ ((__nothrow__ , __leaf__));
817
818 extern void srand (unsigned int __seed) __attribute__ ((__nothrow__ ,
       __leaf__));
819
820
821
822
823 extern int rand_r (unsigned int *__seed) __attribute__ ((__nothrow__ ,
       __leaf__));
824
825
826
827
828
829
830
831 extern double drand48 (void) __attribute__ ((__nothrow__ , __leaf__));
832 extern double erand48 (unsigned short int __xsubi[3]) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
833
834
835 extern long int lrand48 (void) __attribute__ ((__nothrow__ , __leaf__));
836 extern long int nrand48 (unsigned short int __xsubi[3])
837      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1)));
838
839
840 extern long int mrand48 (void) __attribute__ ((__nothrow__ , __leaf__));
841 extern long int jrand48 (unsigned short int __xsubi[3])
842      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1)));
843
844
845 extern void srand48 (long int __seedval) __attribute__ ((__nothrow__ ,
       __leaf__));
846 extern unsigned short int *seed48 (unsigned short int __seed16v[3])
847      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (1)));
848 extern void lcong48 (unsigned short int __param[7]) __attribute__ ((
       __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
849
850
851
852
853
854 struct drand48_data
855   {
856     unsigned short int __x[3];
```

```
857     unsigned short int __old_x[3];
858     unsigned short int __c;
859     unsigned short int __init;
860     __extension__ unsigned long long int __a;
861
862   };
863
864
865 extern int drand48_r (struct drand48_data *__restrict __buffer,
866         double *__restrict __result) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
867 extern int erand48_r (unsigned short int __xsubi[3],
868         struct drand48_data *__restrict __buffer,
869         double *__restrict __result) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
870
871
872 extern int lrand48_r (struct drand48_data *__restrict __buffer,
873         long int *__restrict __result)
874     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
875 extern int nrand48_r (unsigned short int __xsubi[3],
876         struct drand48_data *__restrict __buffer,
877         long int *__restrict __result)
878     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
879
880
881 extern int mrand48_r (struct drand48_data *__restrict __buffer,
882         long int *__restrict __result)
883     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
884 extern int jrand48_r (unsigned short int __xsubi[3],
885         struct drand48_data *__restrict __buffer,
886         long int *__restrict __result)
887     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
888
889
890 extern int srand48_r (long int __seedval, struct drand48_data *__buffer)
891     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (2)));
892
893 extern int seed48_r (unsigned short int __seed16v[3],
894       struct drand48_data *__buffer) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1, 2)));
895
896 extern int lcong48_r (unsigned short int __param[7],
897         struct drand48_data *__buffer)
898     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1, 2)));
899
900
901
```

```
902
903
904
905
906
907
908 extern void *malloc (size_t __size) __attribute__ ((__nothrow__ , __leaf__
    )) __attribute__ ((__malloc__)) ;
909
910 extern void *calloc (size_t __nmemb, size_t __size)
911      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
     ;
912
913
914
915
916
917
918
919
920
921
922 extern void *realloc (void *__ptr, size_t __size)
923      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
    __warn_unused_result__));
924
925 extern void free (void *__ptr) __attribute__ ((__nothrow__ , __leaf__));
926
927
928
929
930 extern void cfree (void *__ptr) __attribute__ ((__nothrow__ , __leaf__));
931
932
933
934 # 1 "/usr/include/alloca.h" 1 3 4
935 # 24 "/usr/include/alloca.h" 3 4
936 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
937 # 25 "/usr/include/alloca.h" 2 3 4
938
939
940
941
942
943
944
945 extern void *alloca (size_t __size) __attribute__ ((__nothrow__ , __leaf__
    ));
946
947
948
949
950
951
```

```
952 # 493 "/usr/include/stdlib.h" 2 3 4
953
954
955
956
957
958 extern void *valloc (size_t __size) __attribute__ ((__nothrow__ , __leaf__
    )) __attribute__ ((__malloc__)) ;
959
960
961
962
963 extern int posix_memalign (void **__memptr, size_t __alignment, size_t
    __size)
964       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1))) ;
965
966
967
968
969 extern void *aligned_alloc (size_t __alignment, size_t __size)
970       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
     __attribute__ ((__alloc_size__ (2))) ;
971
972
973
974
975 extern void abort (void) __attribute__ ((__nothrow__ , __leaf__))
    __attribute__ ((__noreturn__));
976
977
978
979 extern int atexit (void (*__func) (void)) __attribute__ ((__nothrow__ ,
    __leaf__)) __attribute__ ((__nonnull__ (1)));
980
981
982
983
984
985
986
987 extern int at_quick_exit (void (*__func) (void)) __attribute__ ((
    __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
988
989
990
991
992
993
994
995 extern int on_exit (void (*__func) (int __status, void *__arg), void *
    __arg)
996       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (1)));
```

```
997
998
999
1000
1001
1002
1003 extern void exit (int __status) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__noreturn__));
1004
1005
1006
1007
1008
1009 extern void quick_exit (int __status) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__noreturn__));
1010
1011
1012
1013
1014
1015
1016
1017 extern void _Exit (int __status) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__noreturn__));
1018
1019
1020
1021
1022
1023
1024 extern char *getenv (const char *__name) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1))) ;
1025
1026 # 578 "/usr/include/stdlib.h" 3 4
1027 extern int putenv (char *__string) __attribute__ ((__nothrow__ , __leaf__)
       ) __attribute__ ((__nonnull__ (1)));
1028
1029
1030
1031
1032
1033 extern int setenv (const char *__name, const char *__value, int __replace)
1034      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
       (2)));
1035
1036
1037 extern int unsetenv (const char *__name) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__nonnull__ (1)));
1038
1039
1040
1041
1042
1043
```

```
extern int clearenv (void) __attribute__ ((__nothrow__ , __leaf__));
# 606 "/usr/include/stdlib.h" 3 4
extern char *mktemp (char *__template) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (1)));
# 620 "/usr/include/stdlib.h" 3 4
extern int mkstemp (char *__template) __attribute__ ((__nonnull__ (1))) ;
# 642 "/usr/include/stdlib.h" 3 4
extern int mkstemps (char *__template, int __suffixlen) __attribute__ ((
   __nonnull__ (1))) ;
# 663 "/usr/include/stdlib.h" 3 4
extern char *mkdtemp (char *__template) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (1))) ;
# 712 "/usr/include/stdlib.h" 3 4
```

```
extern int system (const char *__command) ;
```

```
# 734 "/usr/include/stdlib.h" 3 4
extern char *realpath (const char *__restrict __name,
         char *__restrict __resolved) __attribute__ ((__nothrow__ ,
   __leaf__)) ;
```

```
typedef int (*__compar_fn_t) (const void *, const void *);
# 752 "/usr/include/stdlib.h" 3 4
```

```
extern void *bsearch (const void *__key, const void *__base,
        size_t __nmemb, size_t __size, __compar_fn_t __compar)
     __attribute__ ((__nonnull__ (1, 2, 5))) ;
```

```
extern void qsort (void *__base, size_t __nmemb, size_t __size,
     __compar_fn_t __compar) __attribute__ ((__nonnull__ (1, 4)));
# 775 "/usr/include/stdlib.h" 3 4
extern int abs (int __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)) ;
extern long int labs (long int __x) __attribute__ ((__nothrow__ , __leaf__
   )) __attribute__ ((__const__)) ;
```

```
1092
1093 __extension__ extern long long int llabs (long long int __x)
1094      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
     ;
1095
1096
1097
1098
1099
1100
1101
1102 extern div_t div (int __numer, int __denom)
1103      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
     ;
1104 extern ldiv_t ldiv (long int __numer, long int __denom)
1105      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
     ;
1106
1107
1108
1109
1110 __extension__ extern lldiv_t lldiv (long long int __numer,
1111         long long int __denom)
1112      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__))
     ;
1113
1114 # 812 "/usr/include/stdlib.h" 3 4
1115 extern char *ecvt (double __value, int __ndigit, int *__restrict __decpt,
1116      int *__restrict __sign) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__nonnull__ (3, 4))) ;
1117
1118
1119
1120
1121 extern char *fcvt (double __value, int __ndigit, int *__restrict __decpt,
1122      int *__restrict __sign) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__nonnull__ (3, 4))) ;
1123
1124
1125
1126
1127 extern char *gcvt (double __value, int __ndigit, char *__buf)
1128      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3))) ;
1129
1130
1131
1132
1133 extern char *qecvt (long double __value, int __ndigit,
1134       int *__restrict __decpt, int *__restrict __sign)
1135      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (3, 4))) ;
1136 extern char *qfcvt (long double __value, int __ndigit,
1137       int *__restrict __decpt, int *__restrict __sign)
```

```
1138      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (3, 4))) ;
1139 extern char *qgcvt (long double __value, int __ndigit, char *__buf)
1140      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (3))) ;
1141
1142
1143
1144
1145 extern int ecvt_r (double __value, int __ndigit, int *__restrict __decpt,
1146      int *__restrict __sign, char *__restrict __buf,
1147      size_t __len) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
    ((__nonnull__ (3, 4, 5)));
1148 extern int fcvt_r (double __value, int __ndigit, int *__restrict __decpt,
1149      int *__restrict __sign, char *__restrict __buf,
1150      size_t __len) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
    ((__nonnull__ (3, 4, 5)));
1151
1152 extern int qecvt_r (long double __value, int __ndigit,
1153       int *__restrict __decpt, int *__restrict __sign,
1154       char *__restrict __buf, size_t __len)
1155      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (3, 4, 5)));
1156 extern int qfcvt_r (long double __value, int __ndigit,
1157       int *__restrict __decpt, int *__restrict __sign,
1158       char *__restrict __buf, size_t __len)
1159      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (3, 4, 5)));
1160
1161
1162
1163
1164
1165
1166 extern int mblen (const char *__s, size_t __n) __attribute__ ((__nothrow__
     , __leaf__));
1167
1168
1169 extern int mbtowc (wchar_t *__restrict __pwc,
1170      const char *__restrict __s, size_t __n) __attribute__ ((__nothrow__ ,
     __leaf__));
1171
1172
1173 extern int wctomb (char *__s, wchar_t __wchar) __attribute__ ((__nothrow__
     , __leaf__));
1174
1175
1176
1177 extern size_t mbstowcs (wchar_t *__restrict __pwcs,
1178    const char *__restrict __s, size_t __n) __attribute__ ((__nothrow__ ,
    __leaf__));
1179
1180 extern size_t wcstombs (char *__restrict __s,
1181    const wchar_t *__restrict __pwcs, size_t __n)
```

```
     __attribute__ ((__nothrow__ , __leaf__));
```

```
extern int rpmatch (const char *__response) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (1))) ;
# 899 "/usr/include/stdlib.h" 3 4
extern int getsubopt (char **__restrict __optionp,
        char *const *__restrict __tokens,
        char **__restrict __valuep)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2, 3))) ;
# 951 "/usr/include/stdlib.h" 3 4
extern int getloadavg (double __loadavg[], int __nelem)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1)));

# 1 "/usr/include/x86_64-linux-gnu/bits/stdlib-float.h" 1 3 4
# 956 "/usr/include/stdlib.h" 2 3 4
# 968 "/usr/include/stdlib.h" 3 4

# 4 "headers/all_headers.h" 2
# 1 "/usr/include/stdio.h" 1 3 4
# 29 "/usr/include/stdio.h" 3 4


# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 34 "/usr/include/stdio.h" 2 3 4
# 44 "/usr/include/stdio.h" 3 4
struct _IO_FILE;


typedef struct _IO_FILE FILE;


# 64 "/usr/include/stdio.h" 3 4
typedef struct _IO_FILE __FILE;
# 74 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/libio.h" 1 3 4
# 31 "/usr/include/libio.h" 3 4
# 1 "/usr/include/_G_config.h" 1 3 4
# 15 "/usr/include/_G_config.h" 3 4
```

```
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
# 16 "/usr/include/_G_config.h" 2 3 4




# 1 "/usr/include/wchar.h" 1 3 4
# 82 "/usr/include/wchar.h" 3 4
typedef struct
{
  int __count;
  union
  {

    unsigned int __wch;



    char __wchb[4];
  } __value;
} __mbstate_t;
# 21 "/usr/include/_G_config.h" 2 3 4
typedef struct
{
  __off_t __pos;
  __mbstate_t __state;
} _G_fpos_t;
typedef struct
{
  __off64_t __pos;
  __mbstate_t __state;
} _G_fpos64_t;
# 32 "/usr/include/libio.h" 2 3 4
# 49 "/usr/include/libio.h" 3 4
# 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdarg.h" 1 3 4
# 40 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdarg.h" 3 4
typedef __builtin_va_list __gnuc_va_list;
# 50 "/usr/include/libio.h" 2 3 4
# 144 "/usr/include/libio.h" 3 4
struct _IO_jump_t; struct _IO_FILE;
# 154 "/usr/include/libio.h" 3 4
typedef void _IO_lock_t;




struct _IO_marker {
  struct _IO_marker *_next;
  struct _IO_FILE *_sbuf;



  int _pos;
```

```
# 177 "/usr/include/libio.h" 3 4
};


enum __codecvt_result
{
  __codecvt_ok,
  __codecvt_partial,
  __codecvt_error,
  __codecvt_noconv
};
# 245 "/usr/include/libio.h" 3 4
struct _IO_FILE {
  int _flags;
#define _IO_file_flags _flags


  char* _IO_read_ptr;
  char* _IO_read_end;
  char* _IO_read_base;
  char* _IO_write_base;
  char* _IO_write_ptr;
  char* _IO_write_end;
  char* _IO_buf_base;
  char* _IO_buf_end;

  char *_IO_save_base;
  char *_IO_backup_base;
  char *_IO_save_end;

  struct _IO_marker *_markers;

  struct _IO_FILE *_chain;

  int _fileno;
#if 0
  int _blksize;
#else
  int _flags2;
#endif
  __off_t _old_offset;


  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];


  _IO_lock_t *_lock;
# 293 "/usr/include/libio.h" 3 4
  __off64_t _offset;
```

```
# 302 "/usr/include/libio.h" 3 4
  void *__pad1;
  void *__pad2;
  void *__pad3;
  void *__pad4;
  size_t __pad5;

  int _mode;

  char _unused2[15 * sizeof (int) - 4 * sizeof (void *) - sizeof (size_t)
   ];

};


typedef struct _IO_FILE _IO_FILE;


struct _IO_FILE_plus;

extern struct _IO_FILE_plus _IO_2_1_stdin_;
extern struct _IO_FILE_plus _IO_2_1_stdout_;
extern struct _IO_FILE_plus _IO_2_1_stderr_;
# 338 "/usr/include/libio.h" 3 4
typedef __ssize_t __io_read_fn (void *__cookie, char *__buf, size_t
   __nbytes);
```

```
typedef __ssize_t __io_write_fn (void *__cookie, const char *__buf,
     size_t __n);
```

```
typedef int __io_seek_fn (void *__cookie, __off64_t *__pos, int __w);


typedef int __io_close_fn (void *__cookie);
# 390 "/usr/include/libio.h" 3 4
extern int __underflow (_IO_FILE *);
extern int __uflow (_IO_FILE *);
extern int __overflow (_IO_FILE *, int);
# 434 "/usr/include/libio.h" 3 4
extern int _IO_getc (_IO_FILE *__fp);
extern int _IO_putc (int __c, _IO_FILE *__fp);
```

```
extern int _IO_feof (_IO_FILE *__fp) __attribute__ ((__nothrow__ ,
   __leaf__));
extern int _IO_ferror (_IO_FILE *__fp) __attribute__ ((__nothrow__ ,
   __leaf__));

extern int _IO_peekc_locked (_IO_FILE *__fp);




extern void _IO_flockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
   __leaf__));
extern void _IO_funlockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
   __leaf__));
extern int _IO_ftrylockfile (_IO_FILE *) __attribute__ ((__nothrow__ ,
   __leaf__));
# 464 "/usr/include/libio.h" 3 4
extern int _IO_vfscanf (_IO_FILE * __restrict, const char * __restrict,
   __gnuc_va_list, int *__restrict);
extern int _IO_vfprintf (_IO_FILE *__restrict, const char *__restrict,
    __gnuc_va_list);
extern __ssize_t _IO_padn (_IO_FILE *, int, __ssize_t);
extern size_t _IO_sgetn (_IO_FILE *, void *, size_t);

extern __off64_t _IO_seekoff (_IO_FILE *, __off64_t, int, int);
extern __off64_t _IO_seekpos (_IO_FILE *, __off64_t, int);

extern void _IO_free_backup_area (_IO_FILE *) __attribute__ ((__nothrow__
   , __leaf__));
# 75 "/usr/include/stdio.h" 2 3 4




typedef __gnuc_va_list va_list;
# 108 "/usr/include/stdio.h" 3 4


typedef _G_fpos_t fpos_t;




# 164 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/stdio_lim.h" 1 3 4
# 165 "/usr/include/stdio.h" 2 3 4



extern struct _IO_FILE *stdin;
extern struct _IO_FILE *stdout;
extern struct _IO_FILE *stderr;

```

```
1440
1441
1442
1443
1444
1445
1446 extern int remove (const char *__filename) __attribute__ ((__nothrow__ ,
       __leaf__));
1447
1448 extern int rename (const char *__old, const char *__new) __attribute__ ((
       __nothrow__ , __leaf__));
1449
1450
1451
1452
1453 extern int renameat (int __oldfd, const char *__old, int __newfd,
1454        const char *__new) __attribute__ ((__nothrow__ , __leaf__));
1455
1456
1457
1458
1459
1460
1461
1462
1463 extern FILE *tmpfile (void) ;
1464 # 209 "/usr/include/stdio.h" 3 4
1465 extern char *tmpnam (char *__s) __attribute__ ((__nothrow__ , __leaf__)) ;
1466
1467
1468
1469
1470
1471 extern char *tmpnam_r (char *__s) __attribute__ ((__nothrow__ , __leaf__))
       ;
1472 # 227 "/usr/include/stdio.h" 3 4
1473 extern char *tempnam (const char *__dir, const char *__pfx)
1474      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
       ;
1475
1476
1477
1478
1479
1480
1481
1482
1483 extern int fclose (FILE *__stream);
1484
1485
1486
1487
1488 extern int fflush (FILE *__stream);
1489
```

```
# 252 "/usr/include/stdio.h" 3 4
extern int fflush_unlocked (FILE *__stream);
# 266 "/usr/include/stdio.h" 3 4
extern FILE *fopen (const char *__restrict __filename,
      const char *__restrict __modes) ;
extern FILE *freopen (const char *__restrict __filename,
        const char *__restrict __modes,
        FILE *__restrict __stream) ;
# 295 "/usr/include/stdio.h" 3 4
# 306 "/usr/include/stdio.h" 3 4
extern FILE *fdopen (int __fd, const char *__modes) __attribute__ ((
   __nothrow__ , __leaf__)) ;
# 319 "/usr/include/stdio.h" 3 4
extern FILE *fmemopen (void *__s, size_t __len, const char *__modes)
  __attribute__ ((__nothrow__ , __leaf__)) ;
extern FILE *open_memstream (char **__bufloc, size_t *__sizeloc)
   __attribute__ ((__nothrow__ , __leaf__)) ;
extern void setbuf (FILE *__restrict __stream, char *__restrict __buf)
   __attribute__ ((__nothrow__ , __leaf__));
extern int setvbuf (FILE *__restrict __stream, char *__restrict __buf,
      int __modes, size_t __n) __attribute__ ((__nothrow__ , __leaf__));
extern void setbuffer (FILE *__restrict __stream, char *__restrict __buf,
         size_t __size) __attribute__ ((__nothrow__ , __leaf__));
```

```
1541 extern void setlinebuf (FILE *__stream) __attribute__ ((__nothrow__ ,
       __leaf__));
1542
1543
1544
1545
1546
1547
1548
1549
1550 extern int fprintf (FILE *__restrict __stream,
1551       const char *__restrict __format, ...);
1552
1553
1554
1555
1556 extern int printf (const char *__restrict __format, ...);
1557
1558 extern int sprintf (char *__restrict __s,
1559       const char *__restrict __format, ...) __attribute__ ((__nothrow__));
1560
1561
1562
1563
1564
1565 extern int vfprintf (FILE *__restrict __s, const char *__restrict __format
       ,
1566        __gnuc_va_list __arg);
1567
1568
1569
1570
1571 extern int vprintf (const char *__restrict __format, __gnuc_va_list __arg)
       ;
1572
1573 extern int vsprintf (char *__restrict __s, const char *__restrict __format
       ,
1574        __gnuc_va_list __arg) __attribute__ ((__nothrow__));
1575
1576
1577
1578
1579
1580 extern int snprintf (char *__restrict __s, size_t __maxlen,
1581        const char *__restrict __format, ...)
1582      __attribute__ ((__nothrow__)) __attribute__ ((__format__ (__printf__,
       3, 4)));
1583
1584 extern int vsnprintf (char *__restrict __s, size_t __maxlen,
1585         const char *__restrict __format, __gnuc_va_list __arg)
1586      __attribute__ ((__nothrow__)) __attribute__ ((__format__ (__printf__,
       3, 0)));
1587
1588 # 412 "/usr/include/stdio.h" 3 4
```

```
1589 extern int vdprintf (int __fd, const char *__restrict __fmt,
1590        __gnuc_va_list __arg)
1591      __attribute__ ((__format__ (__printf__, 2, 0)));
1592 extern int dprintf (int __fd, const char *__restrict __fmt, ...)
1593      __attribute__ ((__format__ (__printf__, 2, 3)));
1594
1595
1596
1597
1598
1599
1600
1601
1602 extern int fscanf (FILE *__restrict __stream,
1603      const char *__restrict __format, ...) ;
1604
1605
1606
1607
1608 extern int scanf (const char *__restrict __format, ...) ;
1609
1610 extern int sscanf (const char *__restrict __s,
1611      const char *__restrict __format, ...) __attribute__ ((__nothrow__ ,
     __leaf__));
1612 # 443 "/usr/include/stdio.h" 3 4
1613 extern int fscanf (FILE *__restrict __stream, const char *__restrict
     __format, ...) __asm__ ("" "__isoc99_fscanf")
1614
1615                                ;
1616 extern int scanf (const char *__restrict __format, ...) __asm__ ("" "
     __isoc99_scanf")
1617                               ;
1618 extern int sscanf (const char *__restrict __s, const char *__restrict
     __format, ...) __asm__ ("" "__isoc99_sscanf") __attribute__ ((
     __nothrow__ , __leaf__))
1619
1620                        ;
1621 # 463 "/usr/include/stdio.h" 3 4
1622
1623
1624
1625
1626
1627
1628
1629
1630 extern int vfscanf (FILE *__restrict __s, const char *__restrict __format,
1631       __gnuc_va_list __arg)
1632      __attribute__ ((__format__ (__scanf__, 2, 0))) ;
1633
1634
1635
1636
1637
```

```
extern int vscanf (const char *__restrict __format, __gnuc_va_list __arg)
     __attribute__ ((__format__ (__scanf__, 1, 0))) ;


extern int vsscanf (const char *__restrict __s,
      const char *__restrict __format, __gnuc_va_list __arg)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__format__ (
   __scanf__, 2, 0)));
# 494 "/usr/include/stdio.h" 3 4
extern int vfscanf (FILE *__restrict __s, const char *__restrict __format,
     __gnuc_va_list __arg) __asm__ ("" "__isoc99_vfscanf")



     __attribute__ ((__format__ (__scanf__, 2, 0))) ;
extern int vscanf (const char *__restrict __format, __gnuc_va_list __arg)
   __asm__ ("" "__isoc99_vscanf")

     __attribute__ ((__format__ (__scanf__, 1, 0))) ;
extern int vsscanf (const char *__restrict __s, const char *__restrict
   __format, __gnuc_va_list __arg) __asm__ ("" "__isoc99_vsscanf")
   __attribute__ ((__nothrow__ , __leaf__))



     __attribute__ ((__format__ (__scanf__, 2, 0)));
# 522 "/usr/include/stdio.h" 3 4









extern int fgetc (FILE *__stream);
extern int getc (FILE *__stream);




extern int getchar (void);

# 550 "/usr/include/stdio.h" 3 4
extern int getc_unlocked (FILE *__stream);
extern int getchar_unlocked (void);
# 561 "/usr/include/stdio.h" 3 4
extern int fgetc_unlocked (FILE *__stream);



```

```
1687
1688
1689
1690
1691
1692
1693
1694 extern int fputc (int __c, FILE *__stream);
1695 extern int putc (int __c, FILE *__stream);
1696
1697
1698
1699
1700
1701 extern int putchar (int __c);
1702
1703 # 594 "/usr/include/stdio.h" 3 4
1704 extern int fputc_unlocked (int __c, FILE *__stream);
1705
1706
1707
1708
1709
1710
1711
1712 extern int putc_unlocked (int __c, FILE *__stream);
1713 extern int putchar_unlocked (int __c);
1714
1715
1716
1717
1718
1719
1720 extern int getw (FILE *__stream);
1721
1722
1723 extern int putw (int __w, FILE *__stream);
1724
1725
1726
1727
1728
1729
1730
1731
1732 extern char *fgets (char *__restrict __s, int __n, FILE *__restrict
        __stream)
1733       ;
1734 # 640 "/usr/include/stdio.h" 3 4
1735
1736 # 665 "/usr/include/stdio.h" 3 4
1737 extern __ssize_t __getdelim (char **__restrict __lineptr,
1738           size_t *__restrict __n, int __delimiter,
1739           FILE *__restrict __stream) ;
```

```
extern __ssize_t getdelim (char **__restrict __lineptr,
        size_t *__restrict __n, int __delimiter,
        FILE *__restrict __stream) ;
extern __ssize_t getline (char **__restrict __lineptr,
       size_t *__restrict __n,
       FILE *__restrict __stream) ;
extern int fputs (const char *__restrict __s, FILE *__restrict __stream);
extern int puts (const char *__s);
extern int ungetc (int __c, FILE *__stream);
extern size_t fread (void *__restrict __ptr, size_t __size,
       size_t __n, FILE *__restrict __stream) ;
extern size_t fwrite (const void *__restrict __ptr, size_t __size,
        size_t __n, FILE *__restrict __s);
# 737 "/usr/include/stdio.h" 3 4
extern size_t fread_unlocked (void *__restrict __ptr, size_t __size,
         size_t __n, FILE *__restrict __stream) ;
```

```
extern size_t fwrite_unlocked (const void *__restrict __ptr, size_t __size
    ,
           size_t __n, FILE *__restrict __stream);
```

```
extern int fseek (FILE *__stream, long int __off, int __whence);
```

```
extern long int ftell (FILE *__stream) ;
```

```
extern void rewind (FILE *__stream);

# 773 "/usr/include/stdio.h" 3 4
extern int fseeko (FILE *__stream, __off_t __off, int __whence);
```

```
extern __off_t ftello (FILE *__stream) ;
# 792 "/usr/include/stdio.h" 3 4
```

```
extern int fgetpos (FILE *__restrict __stream, fpos_t *__restrict __pos);
```

```
extern int fsetpos (FILE *__stream, const fpos_t *__pos);
# 815 "/usr/include/stdio.h" 3 4

# 824 "/usr/include/stdio.h" 3 4

extern void clearerr (FILE *__stream) __attribute__ ((__nothrow__ ,
    __leaf__));

extern int feof (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__))
    ;

```

```
extern int ferror (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;



extern void clearerr_unlocked (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));
extern int feof_unlocked (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;
extern int ferror_unlocked (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;









extern void perror (const char *__s);




# 1 "/usr/include/x86_64-linux-gnu/bits/sys_errlist.h" 1 3 4
# 26 "/usr/include/x86_64-linux-gnu/bits/sys_errlist.h" 3 4
extern int sys_nerr;
extern const char *const sys_errlist[];
# 854 "/usr/include/stdio.h" 2 3 4



extern int fileno (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;



extern int fileno_unlocked (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;
# 873 "/usr/include/stdio.h" 3 4
extern FILE *popen (const char *__command, const char *__modes) ;




extern int pclose (FILE *__stream);

```

```
1892
1893
1894
1895 extern char *ctermid (char *__s) __attribute__ ((__nothrow__ , __leaf__));
1896 # 913 "/usr/include/stdio.h" 3 4
1897 extern void flockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
        __leaf__));
1898
1899
1900
1901 extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
        __leaf__)) ;
1902
1903
1904 extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ ,
        __leaf__));
1905 # 943 "/usr/include/stdio.h" 3 4
1906
1907 # 5 "headers/all_headers.h" 2
1908 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stdbool.h" 1 3 4
1909 # 6 "headers/all_headers.h" 2
1910 # 1 "/usr/include/math.h" 1 3 4
1911 # 28 "/usr/include/math.h" 3 4
1912
1913
1914
1915
1916 # 1 "/usr/include/x86_64-linux-gnu/bits/huge_val.h" 1 3 4
1917 # 33 "/usr/include/math.h" 2 3 4
1918
1919 # 1 "/usr/include/x86_64-linux-gnu/bits/huge_valf.h" 1 3 4
1920 # 35 "/usr/include/math.h" 2 3 4
1921 # 1 "/usr/include/x86_64-linux-gnu/bits/huge_vall.h" 1 3 4
1922 # 36 "/usr/include/math.h" 2 3 4
1923
1924
1925 # 1 "/usr/include/x86_64-linux-gnu/bits/inf.h" 1 3 4
1926 # 39 "/usr/include/math.h" 2 3 4
1927
1928
1929 # 1 "/usr/include/x86_64-linux-gnu/bits/nan.h" 1 3 4
1930 # 42 "/usr/include/math.h" 2 3 4
1931
1932
1933
1934 # 1 "/usr/include/x86_64-linux-gnu/bits/mathdef.h" 1 3 4
1935 # 28 "/usr/include/x86_64-linux-gnu/bits/mathdef.h" 3 4
1936 typedef float float_t;
1937 typedef double double_t;
1938 # 46 "/usr/include/math.h" 2 3 4
1939 # 69 "/usr/include/math.h" 3 4
1940 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
1941 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
1942
```

```
1943
1944 extern double acos (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __acos (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1945
1946 extern double asin (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __asin (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1947
1948 extern double atan (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __atan (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1949
1950 extern double atan2 (double __y, double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern double __atan2 (double __y, double __x)
     __attribute__ ((__nothrow__ , __leaf__));
1951
1952
1953 extern double cos (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __cos (double __x) __attribute__ ((__nothrow__ , __leaf__
     ));
1954
1955 extern double sin (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __sin (double __x) __attribute__ ((__nothrow__ , __leaf__
     ));
1956
1957 extern double tan (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __tan (double __x) __attribute__ ((__nothrow__ , __leaf__
     ));
1958
1959
1960
1961
1962 extern double cosh (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __cosh (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1963
1964 extern double sinh (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __sinh (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1965
1966 extern double tanh (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __tanh (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1967
1968 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
1969
1970
1971 extern double acosh (double __x) __attribute__ ((__nothrow__ , __leaf__));
      extern double __acosh (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
1972
1973 extern double asinh (double __x) __attribute__ ((__nothrow__ , __leaf__));
      extern double __asinh (double __x) __attribute__ ((__nothrow__ ,
```

```
   __leaf__));
1974
1975 extern double atanh (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __atanh (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1976
1977
1978
1979
1980
1981
1982
1983 extern double exp (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __exp (double __x) __attribute__ ((__nothrow__ , __leaf__
   ));
1984
1985
1986 extern double frexp (double __x, int *__exponent) __attribute__ ((
   __nothrow__ , __leaf__)); extern double __frexp (double __x, int *
   __exponent) __attribute__ ((__nothrow__ , __leaf__));
1987
1988
1989 extern double ldexp (double __x, int __exponent) __attribute__ ((
   __nothrow__ , __leaf__)); extern double __ldexp (double __x, int
   __exponent) __attribute__ ((__nothrow__ , __leaf__));
1990
1991
1992 extern double log (double __x) __attribute__ ((__nothrow__ , __leaf__));
   extern double __log (double __x) __attribute__ ((__nothrow__ , __leaf__
   ));
1993
1994
1995 extern double log10 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __log10 (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
1996
1997
1998 extern double modf (double __x, double *__iptr) __attribute__ ((
   __nothrow__ , __leaf__)); extern double __modf (double __x, double *
   __iptr) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
   __nonnull__ (2)));
1999
2000 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2001
2002
2003 extern double expm1 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __expm1 (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
2004
2005
2006 extern double log1p (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __log1p (double __x) __attribute__ ((__nothrow__ ,
   __leaf__));
2007
```

```
2008
2009 extern double logb (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __logb (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2010
2011
2012
2013
2014
2015
2016 extern double exp2 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __exp2 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2017
2018
2019 extern double log2 (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __log2 (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2020
2021
2022
2023
2024
2025
2026
2027
2028 extern double pow (double __x, double __y) __attribute__ ((__nothrow__ ,
     __leaf__)); extern double __pow (double __x, double __y) __attribute__
     ((__nothrow__ , __leaf__));
2029
2030
2031 extern double sqrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __sqrt (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2032
2033
2034
2035
2036
2037 extern double hypot (double __x, double __y) __attribute__ ((__nothrow__ ,
      __leaf__)); extern double __hypot (double __x, double __y)
     __attribute__ ((__nothrow__ , __leaf__));
2038
2039
2040
2041
2042
2043
2044 extern double cbrt (double __x) __attribute__ ((__nothrow__ , __leaf__));
     extern double __cbrt (double __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2045
2046
2047
```

```
2048
2049
2050
2051
2052
2053 extern double ceil (double __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)); extern double __ceil (double __x)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2054
2055
2056 extern double fabs (double __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)); extern double __fabs (double __x)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2057
2058
2059 extern double floor (double __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)); extern double __floor (double __x)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2060
2061
2062 extern double fmod (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __fmod (double __x, double __y) __attribute__
        ((__nothrow__ , __leaf__));
2063
2064
2065
2066
2067 extern int __isinf (double __value) __attribute__ ((__nothrow__ , __leaf__
       )) __attribute__ ((__const__));
2068
2069
2070 extern int __finite (double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2071
2072
2073
2074
2075
2076 extern int isinf (double __value) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__));
2077
2078
2079 extern int finite (double __value) __attribute__ ((__nothrow__ , __leaf__)
       ) __attribute__ ((__const__));
2080
2081
2082 extern double drem (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __drem (double __x, double __y) __attribute__
        ((__nothrow__ , __leaf__));
2083
2084
2085
2086 extern double significand (double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __significand (double __x) __attribute__ ((
```

```
   __nothrow__ , __leaf__));
2087
2088
2089
2090
2091
2092 extern double copysign (double __x, double __y) __attribute__ ((
   __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
   __copysign (double __x, double __y) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__));
2093
2094
2095
2096
2097
2098
2099 extern double nan (const char *__tagb) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__)); extern double __nan (const char
    *__tagb) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
   __const__));
2100
2101
2102
2103
2104
2105 extern int __isnan (double __value) __attribute__ ((__nothrow__ , __leaf__
   )) __attribute__ ((__const__));
2106
2107
2108
2109 extern int isnan (double __value) __attribute__ ((__nothrow__ , __leaf__))
    __attribute__ ((__const__));
2110
2111
2112 extern double j0 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
    double __j0 (double) __attribute__ ((__nothrow__ , __leaf__));
2113 extern double j1 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
    double __j1 (double) __attribute__ ((__nothrow__ , __leaf__));
2114 extern double jn (int, double) __attribute__ ((__nothrow__ , __leaf__));
   extern double __jn (int, double) __attribute__ ((__nothrow__ , __leaf__
   ));
2115 extern double y0 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
    double __y0 (double) __attribute__ ((__nothrow__ , __leaf__));
2116 extern double y1 (double) __attribute__ ((__nothrow__ , __leaf__)); extern
    double __y1 (double) __attribute__ ((__nothrow__ , __leaf__));
2117 extern double yn (int, double) __attribute__ ((__nothrow__ , __leaf__));
   extern double __yn (int, double) __attribute__ ((__nothrow__ , __leaf__
   ));
2118
2119
2120
2121
2122
2123
```

```
2124 extern double erf (double) __attribute__ ((__nothrow__ , __leaf__));
        extern double __erf (double) __attribute__ ((__nothrow__ , __leaf__));
2125 extern double erfc (double) __attribute__ ((__nothrow__ , __leaf__));
        extern double __erfc (double) __attribute__ ((__nothrow__ , __leaf__));
2126 extern double lgamma (double) __attribute__ ((__nothrow__ , __leaf__));
        extern double __lgamma (double) __attribute__ ((__nothrow__ , __leaf__)
        );
2127
2128
2129
2130
2131
2132
2133 extern double tgamma (double) __attribute__ ((__nothrow__ , __leaf__));
        extern double __tgamma (double) __attribute__ ((__nothrow__ , __leaf__)
        );
2134
2135
2136
2137
2138
2139 extern double gamma (double) __attribute__ ((__nothrow__ , __leaf__));
        extern double __gamma (double) __attribute__ ((__nothrow__ , __leaf__))
        ;
2140
2141
2142
2143
2144
2145
2146 extern double lgamma_r (double, int *__signgamp) __attribute__ ((
        __nothrow__ , __leaf__)); extern double __lgamma_r (double, int *
        __signgamp) __attribute__ ((__nothrow__ , __leaf__));
2147
2148
2149
2150
2151
2152
2153
2154 extern double rint (double __x) __attribute__ ((__nothrow__ , __leaf__));
        extern double __rint (double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2155
2156
2157 extern double nextafter (double __x, double __y) __attribute__ ((
        __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
        __nextafter (double __x, double __y) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__));
2158
2159 extern double nexttoward (double __x, long double __y) __attribute__ ((
        __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern double
        __nexttoward (double __x, long double __y) __attribute__ ((__nothrow__
        , __leaf__)) __attribute__ ((__const__));
```

```
2160
2161
2162
2163 extern double remainder (double __x, double __y) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __remainder (double __x, double
        __y) __attribute__ ((__nothrow__ , __leaf__));
2164
2165
2166
2167 extern double scalbn (double __x, int __n) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __scalbn (double __x, int __n) __attribute__
       ((__nothrow__ , __leaf__));
2168
2169
2170
2171 extern int ilogb (double __x) __attribute__ ((__nothrow__ , __leaf__));
       extern int __ilogb (double __x) __attribute__ ((__nothrow__ , __leaf__)
       );
2172
2173
2174
2175
2176 extern double scalbln (double __x, long int __n) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __scalbln (double __x, long int
        __n) __attribute__ ((__nothrow__ , __leaf__));
2177
2178
2179
2180 extern double nearbyint (double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __nearbyint (double __x) __attribute__ ((
       __nothrow__ , __leaf__));
2181
2182
2183
2184 extern double round (double __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)); extern double __round (double __x)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2185
2186
2187
2188 extern double trunc (double __x) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__)); extern double __trunc (double __x)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2189
2190
2191
2192
2193 extern double remquo (double __x, double __y, int *__quo) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __remquo (double __x, double
       __y, int *__quo) __attribute__ ((__nothrow__ , __leaf__));
2194
2195
2196
2197
```

```
2198
2199
2200 extern long int lrint (double __x) __attribute__ ((__nothrow__ , __leaf__)
       ); extern long int __lrint (double __x) __attribute__ ((__nothrow__ ,
       __leaf__));
2201 __extension__
2202 extern long long int llrint (double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long long int __llrint (double __x) __attribute__ ((
       __nothrow__ , __leaf__));
2203
2204
2205
2206 extern long int lround (double __x) __attribute__ ((__nothrow__ , __leaf__
       )); extern long int __lround (double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
2207 __extension__
2208 extern long long int llround (double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long long int __llround (double __x) __attribute__
       ((__nothrow__ , __leaf__));
2209
2210
2211
2212 extern double fdim (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)); extern double __fdim (double __x, double __y) __attribute__
        ((__nothrow__ , __leaf__));
2213
2214
2215 extern double fmax (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern double __fmax (double
       __x, double __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2216
2217
2218 extern double fmin (double __x, double __y) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern double __fmin (double
       __x, double __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2219
2220
2221
2222 extern int __fpclassify (double __value) __attribute__ ((__nothrow__ ,
       __leaf__))
2223      __attribute__ ((__const__));
2224
2225
2226 extern int __signbit (double __value) __attribute__ ((__nothrow__ ,
       __leaf__))
2227      __attribute__ ((__const__));
2228
2229
2230
2231 extern double fma (double __x, double __y, double __z) __attribute__ ((
       __nothrow__ , __leaf__)); extern double __fma (double __x, double __y,
       double __z) __attribute__ ((__nothrow__ , __leaf__));
```

```
2232
2233
2234
2235
2236 # 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2237 extern double scalb (double __x, double __n) __attribute__ ((__nothrow__ ,
         __leaf__)); extern double __scalb (double __x, double __n)
        __attribute__ ((__nothrow__ , __leaf__));
2238 # 70 "/usr/include/math.h" 2 3 4
2239 # 88 "/usr/include/math.h" 3 4
2240 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
2241 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2242
2243
2244 extern float acosf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __acosf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2245
2246 extern float asinf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __asinf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2247
2248 extern float atanf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __atanf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2249
2250 extern float atan2f (float __y, float __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern float __atan2f (float __y, float __x) __attribute__
        ((__nothrow__ , __leaf__));
2251
2252
2253 extern float cosf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __cosf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2254
2255 extern float sinf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __sinf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2256
2257 extern float tanf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __tanf (float __x) __attribute__ ((__nothrow__ , __leaf__)
        );
2258
2259
2260
2261
2262 extern float coshf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __coshf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2263
2264 extern float sinhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __sinhf (float __x) __attribute__ ((__nothrow__ , __leaf__
        ));
2265
```

```
2266 extern float tanhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __tanhf (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2267
2268 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2269
2270
2271 extern float acoshf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __acoshf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2272
2273 extern float asinhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __asinhf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2274
2275 extern float atanhf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __atanhf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2276
2277
2278
2279
2280
2281
2282
2283 extern float expf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __expf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2284
2285
2286 extern float frexpf (float __x, int *__exponent) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __frexpf (float __x, int *
     __exponent) __attribute__ ((__nothrow__ , __leaf__));
2287
2288
2289 extern float ldexpf (float __x, int __exponent) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __ldexpf (float __x, int
     __exponent) __attribute__ ((__nothrow__ , __leaf__));
2290
2291
2292 extern float logf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __logf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2293
2294
2295 extern float log10f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __log10f (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2296
2297
2298 extern float modff (float __x, float *__iptr) __attribute__ ((__nothrow__
     , __leaf__)); extern float __modff (float __x, float *__iptr)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (2)));
```

```
2299
2300 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2301
2302
2303 extern float expm1f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __expm1f (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2304
2305
2306 extern float log1pf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __log1pf (float __x) __attribute__ ((__nothrow__ ,
     __leaf__));
2307
2308
2309 extern float logbf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __logbf (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2310
2311
2312
2313
2314
2315
2316 extern float exp2f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __exp2f (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2317
2318
2319 extern float log2f (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __log2f (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2320
2321
2322
2323
2324
2325
2326
2327
2328 extern float powf (float __x, float __y) __attribute__ ((__nothrow__ ,
     __leaf__)); extern float __powf (float __x, float __y) __attribute__ ((
     __nothrow__ , __leaf__));
2329
2330
2331 extern float sqrtf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern float __sqrtf (float __x) __attribute__ ((__nothrow__ , __leaf__
     ));
2332
2333
2334
2335
2336
2337 extern float hypotf (float __x, float __y) __attribute__ ((__nothrow__ ,
     __leaf__)); extern float __hypotf (float __x, float __y) __attribute__
```

```
   ((__nothrow__ , __leaf__));
2338
2339
2340
2341
2342
2343
2344 extern float cbrtf (float __x) __attribute__ ((__nothrow__ , __leaf__));
   extern float __cbrtf (float __x) __attribute__ ((__nothrow__ , __leaf__
   ));
2345
2346
2347
2348
2349
2350
2351
2352
2353 extern float ceilf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __ceilf (float __x)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2354
2355
2356 extern float fabsf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __fabsf (float __x)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2357
2358
2359 extern float floorf (float __x) __attribute__ ((__nothrow__ , __leaf__))
   __attribute__ ((__const__)); extern float __floorf (float __x)
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2360
2361
2362 extern float fmodf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __fmodf (float __x, float __y) __attribute__
   ((__nothrow__ , __leaf__));
2363
2364
2365
2366
2367 extern int __isinff (float __value) __attribute__ ((__nothrow__ , __leaf__
   )) __attribute__ ((__const__));
2368
2369
2370 extern int __finitef (float __value) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__));
2371
2372
2373
2374
2375
2376 extern int isinff (float __value) __attribute__ ((__nothrow__ , __leaf__))
    __attribute__ ((__const__));
2377
```

```
2378
2379 extern int finitef (float __value) __attribute__ ((__nothrow__ , __leaf__)
        ) __attribute__ ((__const__));
2380
2381
2382 extern float dremf (float __x, float __y) __attribute__ ((__nothrow__ ,
        __leaf__)); extern float __dremf (float __x, float __y) __attribute__
        ((__nothrow__ , __leaf__));
2383
2384
2385
2386 extern float significandf (float __x) __attribute__ ((__nothrow__ ,
        __leaf__)); extern float __significandf (float __x) __attribute__ ((
        __nothrow__ , __leaf__));
2387
2388
2389
2390
2391
2392 extern float copysignf (float __x, float __y) __attribute__ ((__nothrow__
        , __leaf__)) __attribute__ ((__const__)); extern float __copysignf (
        float __x, float __y) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__));
2393
2394
2395
2396
2397
2398
2399 extern float nanf (const char *__tagb) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern float __nanf (const char
         *__tagb) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
        __const__));
2400
2401
2402
2403
2404
2405 extern int __isnanf (float __value) __attribute__ ((__nothrow__ , __leaf__
        )) __attribute__ ((__const__));
2406
2407
2408
2409 extern int isnanf (float __value) __attribute__ ((__nothrow__ , __leaf__))
         __attribute__ ((__const__));
2410
2411
2412 extern float j0f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
        float __j0f (float) __attribute__ ((__nothrow__ , __leaf__));
2413 extern float j1f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
        float __j1f (float) __attribute__ ((__nothrow__ , __leaf__));
2414 extern float jnf (int, float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __jnf (int, float) __attribute__ ((__nothrow__ , __leaf__)
        );
```

```
2415 extern float y0f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
        float __y0f (float) __attribute__ ((__nothrow__ , __leaf__));
2416 extern float y1f (float) __attribute__ ((__nothrow__ , __leaf__)); extern
        float __y1f (float) __attribute__ ((__nothrow__ , __leaf__));
2417 extern float ynf (int, float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __ynf (int, float) __attribute__ ((__nothrow__ , __leaf__)
        );
2418
2419
2420
2421
2422
2423
2424 extern float erff (float) __attribute__ ((__nothrow__ , __leaf__)); extern
         float __erff (float) __attribute__ ((__nothrow__ , __leaf__));
2425 extern float erfcf (float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __erfcf (float) __attribute__ ((__nothrow__ , __leaf__));
2426 extern float lgammaf (float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __lgammaf (float) __attribute__ ((__nothrow__ , __leaf__))
        ;
2427
2428
2429
2430
2431
2432
2433 extern float tgammaf (float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __tgammaf (float) __attribute__ ((__nothrow__ , __leaf__))
        ;
2434
2435
2436
2437
2438
2439 extern float gammaf (float) __attribute__ ((__nothrow__ , __leaf__));
        extern float __gammaf (float) __attribute__ ((__nothrow__ , __leaf__));
2440
2441
2442
2443
2444
2445
2446 extern float lgammaf_r (float, int *__signgamp) __attribute__ ((
        __nothrow__ , __leaf__)); extern float __lgammaf_r (float, int *
        __signgamp) __attribute__ ((__nothrow__ , __leaf__));
2447
2448
2449
2450
2451
2452
2453
2454 extern float rintf (float __x) __attribute__ ((__nothrow__ , __leaf__));
        extern float __rintf (float __x) __attribute__ ((__nothrow__ , __leaf__
```

```
   ));
2455
2456
2457 extern float nextafterf (float __x, float __y) __attribute__ ((__nothrow__
       , __leaf__)) __attribute__ ((__const__)); extern float __nextafterf (
     float __x, float __y) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__const__));
2458
2459 extern float nexttowardf (float __x, long double __y) __attribute__ ((
     __nothrow__ , __leaf__)) __attribute__ ((__const__)); extern float
     __nexttowardf (float __x, long double __y) __attribute__ ((__nothrow__
     , __leaf__)) __attribute__ ((__const__));
2460
2461
2462
2463 extern float remainderf (float __x, float __y) __attribute__ ((__nothrow__
       , __leaf__)); extern float __remainderf (float __x, float __y)
     __attribute__ ((__nothrow__ , __leaf__));
2464
2465
2466
2467 extern float scalbnf (float __x, int __n) __attribute__ ((__nothrow__ ,
     __leaf__)); extern float __scalbnf (float __x, int __n) __attribute__
     ((__nothrow__ , __leaf__));
2468
2469
2470
2471 extern int ilogbf (float __x) __attribute__ ((__nothrow__ , __leaf__));
     extern int __ilogbf (float __x) __attribute__ ((__nothrow__ , __leaf__)
     );
2472
2473
2474
2475
2476 extern float scalblnf (float __x, long int __n) __attribute__ ((
     __nothrow__ , __leaf__)); extern float __scalblnf (float __x, long int
     __n) __attribute__ ((__nothrow__ , __leaf__));
2477
2478
2479
2480 extern float nearbyintf (float __x) __attribute__ ((__nothrow__ , __leaf__
     )); extern float __nearbyintf (float __x) __attribute__ ((__nothrow__ ,
       __leaf__));
2481
2482
2483
2484 extern float roundf (float __x) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__const__)); extern float __roundf (float __x)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2485
2486
2487
2488 extern float truncf (float __x) __attribute__ ((__nothrow__ , __leaf__))
     __attribute__ ((__const__)); extern float __truncf (float __x)
```

```
   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2489
2490
2491
2492
2493 extern float remquof (float __x, float __y, int *__quo) __attribute__ ((
   __nothrow__ , __leaf__)); extern float __remquof (float __x, float __y,
    int *__quo) __attribute__ ((__nothrow__ , __leaf__));
2494
2495
2496
2497
2498
2499
2500 extern long int lrintf (float __x) __attribute__ ((__nothrow__ , __leaf__)
   ); extern long int __lrintf (float __x) __attribute__ ((__nothrow__ ,
   __leaf__));
2501 __extension__
2502 extern long long int llrintf (float __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long long int __llrintf (float __x) __attribute__ ((
   __nothrow__ , __leaf__));
2503
2504
2505
2506 extern long int lroundf (float __x) __attribute__ ((__nothrow__ , __leaf__
   )); extern long int __lroundf (float __x) __attribute__ ((__nothrow__ ,
    __leaf__));
2507 __extension__
2508 extern long long int llroundf (float __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long long int __llroundf (float __x) __attribute__
   ((__nothrow__ , __leaf__));
2509
2510
2511
2512 extern float fdimf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)); extern float __fdimf (float __x, float __y) __attribute__
   ((__nothrow__ , __leaf__));
2513
2514
2515 extern float fmaxf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__)); extern float __fmaxf (float __x
   , float __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
   __const__));
2516
2517
2518 extern float fminf (float __x, float __y) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__const__)); extern float __fminf (float __x
   , float __y) __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((
   __const__));
2519
2520
2521
2522 extern int __fpclassifyf (float __value) __attribute__ ((__nothrow__ ,
   __leaf__))
```

```
2523      __attribute__ ((__const__));
2524
2525
2526 extern int __signbitf (float __value) __attribute__ ((__nothrow__ ,
      __leaf__))
2527      __attribute__ ((__const__));
2528
2529
2530
2531 extern float fmaf (float __x, float __y, float __z) __attribute__ ((
      __nothrow__ , __leaf__)); extern float __fmaf (float __x, float __y,
      float __z) __attribute__ ((__nothrow__ , __leaf__));
2532
2533
2534
2535
2536 # 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2537 extern float scalbf (float __x, float __n) __attribute__ ((__nothrow__ ,
      __leaf__)); extern float __scalbf (float __x, float __n) __attribute__
      ((__nothrow__ , __leaf__));
2538 # 89 "/usr/include/math.h" 2 3 4
2539 # 132 "/usr/include/math.h" 3 4
2540 # 1 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 1 3 4
2541 # 52 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2542
2543
2544 extern long double acosl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __acosl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
2545
2546 extern long double asinl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __asinl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
2547
2548 extern long double atanl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __atanl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
2549
2550 extern long double atan2l (long double __y, long double __x) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __atan2l (long double
      __y, long double __x) __attribute__ ((__nothrow__ , __leaf__));
2551
2552
2553 extern long double cosl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __cosl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
2554
2555 extern long double sinl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __sinl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
2556
2557 extern long double tanl (long double __x) __attribute__ ((__nothrow__ ,
      __leaf__)); extern long double __tanl (long double __x) __attribute__
      ((__nothrow__ , __leaf__));
```

```
2558
2559
2560
2561
2562 extern long double coshl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __coshl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2563
2564 extern long double sinhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __sinhl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2565
2566 extern long double tanhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __tanhl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2567
2568 # 86 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2569
2570
2571 extern long double acoshl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __acoshl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2572
2573 extern long double asinhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __asinhl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2574
2575 extern long double atanhl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __atanhl (long double __x) __attribute__
        ((__nothrow__ , __leaf__));
2576
2577
2578
2579
2580
2581
2582
2583 extern long double expl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __expl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2584
2585
2586 extern long double frexpl (long double __x, int *__exponent) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __frexpl (long double
       __x, int *__exponent) __attribute__ ((__nothrow__ , __leaf__));
2587
2588
2589 extern long double ldexpl (long double __x, int __exponent) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __ldexpl (long double
       __x, int __exponent) __attribute__ ((__nothrow__ , __leaf__));
2590
2591
2592 extern long double logl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __logl (long double __x) __attribute__
```

```
   ((__nothrow__ , __leaf__));
2593
2594
2595 extern long double log10l (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __log10l (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2596
2597
2598 extern long double modfl (long double __x, long double *__iptr)
   __attribute__ ((__nothrow__ , __leaf__)); extern long double __modfl (
   long double __x, long double *__iptr) __attribute__ ((__nothrow__ ,
   __leaf__)) __attribute__ ((__nonnull__ (2)));
2599
2600 # 126 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
2601
2602
2603 extern long double expm1l (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __expm1l (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2604
2605
2606 extern long double log1pl (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __log1pl (long double __x) __attribute__
    ((__nothrow__ , __leaf__));
2607
2608
2609 extern long double logbl (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __logbl (long double __x) __attribute__
   ((__nothrow__ , __leaf__));
2610
2611
2612
2613
2614
2615
2616 extern long double exp2l (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __exp2l (long double __x) __attribute__
   ((__nothrow__ , __leaf__));
2617
2618
2619 extern long double log2l (long double __x) __attribute__ ((__nothrow__ ,
   __leaf__)); extern long double __log2l (long double __x) __attribute__
   ((__nothrow__ , __leaf__));
2620
2621
2622
2623
2624
2625
2626
2627
2628 extern long double powl (long double __x, long double __y) __attribute__
   ((__nothrow__ , __leaf__)); extern long double __powl (long double __x,
    long double __y) __attribute__ ((__nothrow__ , __leaf__));
```

```
2629
2630
2631 extern long double sqrtl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __sqrtl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2632
2633
2634
2635
2636
2637 extern long double hypotl (long double __x, long double __y) __attribute__
        ((__nothrow__ , __leaf__)); extern long double __hypotl (long double
       __x, long double __y) __attribute__ ((__nothrow__ , __leaf__));
2638
2639
2640
2641
2642
2643
2644 extern long double cbrtl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __cbrtl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2645
2646
2647
2648
2649
2650
2651
2652
2653 extern long double ceill (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __ceill (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2654
2655
2656 extern long double fabsl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __fabsl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2657
2658
2659 extern long double floorl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __floorl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2660
2661
2662 extern long double fmodl (long double __x, long double __y) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __fmodl (long double __x
       , long double __y) __attribute__ ((__nothrow__ , __leaf__));
2663
2664
2665
```

```
2666
2667 extern int __isinfl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2668
2669
2670 extern int __finitel (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2671
2672
2673
2674
2675
2676 extern int isinfl (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2677
2678
2679 extern int finitel (long double __value) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__));
2680
2681
2682 extern long double dreml (long double __x, long double __y) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __dreml (long double __x
       , long double __y) __attribute__ ((__nothrow__ , __leaf__));
2683
2684
2685
2686 extern long double significandl (long double __x) __attribute__ ((
       __nothrow__ , __leaf__)); extern long double __significandl (long
       double __x) __attribute__ ((__nothrow__ , __leaf__));
2687
2688
2689
2690
2691
2692 extern long double copysignl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
       extern long double __copysignl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2693
2694
2695
2696
2697
2698
2699 extern long double nanl (const char *__tagb) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__)); extern long double __nanl (
       const char *__tagb) __attribute__ ((__nothrow__ , __leaf__))
       __attribute__ ((__const__));
2700
2701
2702
2703
2704
```

```
2705 extern int __isnanl (long double __value) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__));
2706
2707
2708
2709 extern int isnanl (long double __value) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__const__));
2710
2711
2712 extern long double j0l (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __j0l (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2713 extern long double j1l (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __j1l (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2714 extern long double jnl (int, long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __jnl (int, long double) __attribute__
        ((__nothrow__ , __leaf__));
2715 extern long double y0l (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __y0l (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2716 extern long double y1l (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __y1l (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2717 extern long double ynl (int, long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __ynl (int, long double) __attribute__
        ((__nothrow__ , __leaf__));
2718
2719
2720
2721
2722
2723
2724 extern long double erfl (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __erfl (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2725 extern long double erfcl (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __erfcl (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2726 extern long double lgammal (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __lgammal (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2727
2728
2729
2730
2731
2732
2733 extern long double tgammal (long double) __attribute__ ((__nothrow__ ,
        __leaf__)); extern long double __tgammal (long double) __attribute__ ((
        __nothrow__ , __leaf__));
2734
2735
2736
```

```
2737
2738
2739 extern long double gammal (long double) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __gammal (long double) __attribute__ ((
       __nothrow__ , __leaf__));
2740
2741
2742
2743
2744
2745
2746 extern long double lgammal_r (long double, int *__signgamp) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __lgammal_r (long double
       , int *__signgamp) __attribute__ ((__nothrow__ , __leaf__));
2747
2748
2749
2750
2751
2752
2753
2754 extern long double rintl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long double __rintl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
2755
2756
2757 extern long double nextafterl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
       extern long double __nextafterl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2758
2759 extern long double nexttowardl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
       extern long double __nexttowardl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__const__));
2760
2761
2762
2763 extern long double remainderl (long double __x, long double __y)
       __attribute__ ((__nothrow__ , __leaf__)); extern long double
       __remainderl (long double __x, long double __y) __attribute__ ((
       __nothrow__ , __leaf__));
2764
2765
2766
2767 extern long double scalbnl (long double __x, int __n) __attribute__ ((
       __nothrow__ , __leaf__)); extern long double __scalbnl (long double __x
       , int __n) __attribute__ ((__nothrow__ , __leaf__));
2768
2769
2770
2771 extern int ilogbl (long double __x) __attribute__ ((__nothrow__ , __leaf__
       )); extern int __ilogbl (long double __x) __attribute__ ((__nothrow__ ,
        __leaf__));
```

```
2772
2773
2774
2775
2776 extern long double scalblnl (long double __x, long int __n) __attribute__
       ((__nothrow__ , __leaf__)); extern long double __scalblnl (long double
       __x, long int __n) __attribute__ ((__nothrow__ , __leaf__));
2777
2778
2779
2780 extern long double nearbyintl (long double __x) __attribute__ ((
       __nothrow__ , __leaf__)); extern long double __nearbyintl (long double
       __x) __attribute__ ((__nothrow__ , __leaf__));
2781
2782
2783
2784 extern long double roundl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __roundl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2785
2786
2787
2788 extern long double truncl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)) __attribute__ ((__const__)); extern long double __truncl (
       long double __x) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__const__));
2789
2790
2791
2792
2793 extern long double remquol (long double __x, long double __y, int *__quo)
       __attribute__ ((__nothrow__ , __leaf__)); extern long double __remquol
       (long double __x, long double __y, int *__quo) __attribute__ ((
       __nothrow__ , __leaf__));
2794
2795
2796
2797
2798
2799
2800 extern long int lrintl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long int __lrintl (long double __x) __attribute__ ((
       __nothrow__ , __leaf__));
2801 __extension__
2802 extern long long int llrintl (long double __x) __attribute__ ((__nothrow__
        , __leaf__)); extern long long int __llrintl (long double __x)
       __attribute__ ((__nothrow__ , __leaf__));
2803
2804
2805
2806 extern long int lroundl (long double __x) __attribute__ ((__nothrow__ ,
       __leaf__)); extern long int __lroundl (long double __x) __attribute__
       ((__nothrow__ , __leaf__));
```

```
__extension__
extern long long int llroundl (long double __x) __attribute__ ((
   __nothrow__ , __leaf__)); extern long long int __llroundl (long double
   __x) __attribute__ ((__nothrow__ , __leaf__));



extern long double fdiml (long double __x, long double __y) __attribute__
   ((__nothrow__ , __leaf__)); extern long double __fdiml (long double __x
   , long double __y) __attribute__ ((__nothrow__ , __leaf__));


extern long double fmaxl (long double __x, long double __y) __attribute__
   ((__nothrow__ , __leaf__)) __attribute__ ((__const__)); extern long
   double __fmaxl (long double __x, long double __y) __attribute__ ((
   __nothrow__ , __leaf__)) __attribute__ ((__const__));


extern long double fminl (long double __x, long double __y) __attribute__
   ((__nothrow__ , __leaf__)) __attribute__ ((__const__)); extern long
   double __fminl (long double __x, long double __y) __attribute__ ((
   __nothrow__ , __leaf__)) __attribute__ ((__const__));



extern int __fpclassifyl (long double __value) __attribute__ ((__nothrow__
    , __leaf__))
     __attribute__ ((__const__));


extern int __signbitl (long double __value) __attribute__ ((__nothrow__ ,
   __leaf__))
     __attribute__ ((__const__));



extern long double fmal (long double __x, long double __y, long double __z
   ) __attribute__ ((__nothrow__ , __leaf__)); extern long double __fmal (
   long double __x, long double __y, long double __z) __attribute__ ((
   __nothrow__ , __leaf__));



# 371 "/usr/include/x86_64-linux-gnu/bits/mathcalls.h" 3 4
extern long double scalbl (long double __x, long double __n) __attribute__
    ((__nothrow__ , __leaf__)); extern long double __scalbl (long double
   __x, long double __n) __attribute__ ((__nothrow__ , __leaf__));
# 133 "/usr/include/math.h" 2 3 4
# 148 "/usr/include/math.h" 3 4
extern int signgam;
# 189 "/usr/include/math.h" 3 4
enum
  {
```

```
    FP_NAN =

      0,
    FP_INFINITE =

      1,
    FP_ZERO =

      2,
    FP_SUBNORMAL =

      3,
    FP_NORMAL =

      4
  };
# 301 "/usr/include/math.h" 3 4
typedef enum
{
  _IEEE_ = -1,
  _SVID_,
  _XOPEN_,
  _POSIX_,
  _ISOC_
} _LIB_VERSION_TYPE;




extern _LIB_VERSION_TYPE _LIB_VERSION;
# 326 "/usr/include/math.h" 3 4
struct exception

  {
    int type;
    char *name;
    double arg1;
    double arg2;
    double retval;
  };




extern int matherr (struct exception *__exc);
# 488 "/usr/include/math.h" 3 4

# 7 "headers/all_headers.h" 2
# 1 "/usr/include/setjmp.h" 1 3 4
# 27 "/usr/include/setjmp.h" 3 4


# 1 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 1 3 4
# 26 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 3 4
```

```
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
# 27 "/usr/include/x86_64-linux-gnu/bits/setjmp.h" 2 3 4



typedef long int __jmp_buf[8];
# 30 "/usr/include/setjmp.h" 2 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/sigset.h" 1 3 4
# 31 "/usr/include/setjmp.h" 2 3 4


struct __jmp_buf_tag
  {



    __jmp_buf __jmpbuf;
    int __mask_was_saved;
    __sigset_t __saved_mask;
  };



typedef struct __jmp_buf_tag jmp_buf[1];



extern int setjmp (jmp_buf __env) __attribute__ ((__nothrow__));







extern int __sigsetjmp (struct __jmp_buf_tag __env[1], int __savemask)
   __attribute__ ((__nothrow__));


extern int _setjmp (struct __jmp_buf_tag __env[1]) __attribute__ ((
   __nothrow__));
```

```
2950
2951 extern void longjmp (struct __jmp_buf_tag __env[1], int __val)
2952      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
2953
2954
2955
2956
2957
2958
2959
2960 extern void _longjmp (struct __jmp_buf_tag __env[1], int __val)
2961      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
2962
2963
2964
2965
2966
2967
2968
2969 typedef struct __jmp_buf_tag sigjmp_buf[1];
2970 # 102 "/usr/include/setjmp.h" 3 4
2971 extern void siglongjmp (sigjmp_buf __env, int __val)
2972      __attribute__ ((__nothrow__)) __attribute__ ((__noreturn__));
2973 # 112 "/usr/include/setjmp.h" 3 4
2974
2975 # 8 "headers/all_headers.h" 2
2976 # 1 "/usr/include/string.h" 1 3 4
2977 # 27 "/usr/include/string.h" 3 4
2978
2979
2980
2981
2982
2983 # 1 "/usr/lib/gcc/x86_64-linux-gnu/5/include/stddef.h" 1 3 4
2984 # 33 "/usr/include/string.h" 2 3 4
2985 # 44 "/usr/include/string.h" 3 4
2986
2987
2988 extern void *memcpy (void *__restrict __dest, const void *__restrict __src
     ,
2989        size_t __n) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
     ((__nonnull__ (1, 2)));
2990
2991
2992 extern void *memmove (void *__dest, const void *__src, size_t __n)
2993      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
     (1, 2)));
2994
2995
2996
2997
2998
2999
```

```
3000 extern void *memccpy (void *__restrict __dest, const void *__restrict
        __src,
3001      int __c, size_t __n)
3002   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3003
3004
3005
3006
3007
3008 extern void *memset (void *__s, int __c, size_t __n) __attribute__ ((
        __nothrow__ , __leaf__)) __attribute__ ((__nonnull__ (1)));
3009
3010
3011 extern int memcmp (const void *__s1, const void *__s2, size_t __n)
3012   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3013 # 96 "/usr/include/string.h" 3 4
3014 extern void *memchr (const void *__s, int __c, size_t __n)
3015    __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1)));
3016
3017
3018 # 127 "/usr/include/string.h" 3 4
3019
3020
3021 extern char *strcpy (char *__restrict __dest, const char *__restrict __src
        )
3022   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3023
3024 extern char *strncpy (char *__restrict __dest,
3025      const char *__restrict __src, size_t __n)
3026   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3027
3028
3029 extern char *strcat (char *__restrict __dest, const char *__restrict __src
        )
3030   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3031
3032 extern char *strncat (char *__restrict __dest, const char *__restrict
        __src,
3033      size_t __n) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
        ((__nonnull__ (1, 2)));
3034
3035
3036 extern int strcmp (const char *__s1, const char *__s2)
3037   __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3038
3039 extern int strncmp (const char *__s1, const char *__s2, size_t __n)
```

```
3040      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3041
3042
3043 extern int strcoll (const char *__s1, const char *__s2)
3044      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3045
3046 extern size_t strxfrm (char *__restrict __dest,
3047          const char *__restrict __src, size_t __n)
3048      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (2)));
3049
3050
3051
3052
3053
3054
3055 # 1 "/usr/include/xlocale.h" 1 3 4
3056 # 27 "/usr/include/xlocale.h" 3 4
3057 typedef struct __locale_struct
3058 {
3059
3060   struct __locale_data *__locales[13];
3061
3062
3063   const unsigned short int *__ctype_b;
3064   const int *__ctype_tolower;
3065   const int *__ctype_toupper;
3066
3067
3068   const char *__names[13];
3069 } *__locale_t;
3070
3071
3072 typedef __locale_t locale_t;
3073 # 164 "/usr/include/string.h" 2 3 4
3074
3075
3076 extern int strcoll_l (const char *__s1, const char *__s2, __locale_t __l)
3077      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2, 3)));
3078
3079 extern size_t strxfrm_l (char *__dest, const char *__src, size_t __n,
3080     __locale_t __l) __attribute__ ((__nothrow__ , __leaf__)) __attribute__
    ((__nonnull__ (2, 4)));
3081
3082
3083
3084
3085
3086 extern char *strdup (const char *__s)
3087      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
    __attribute__ ((__nonnull__ (1)));
```

```
3088
3089
3090
3091
3092
3093
3094 extern char *strndup (const char *__string, size_t __n)
3095      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__malloc__))
     __attribute__ ((__nonnull__ (1)));
3096 # 211 "/usr/include/string.h" 3 4
3097
3098 # 236 "/usr/include/string.h" 3 4
3099 extern char *strchr (const char *__s, int __c)
3100      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1)));
3101 # 263 "/usr/include/string.h" 3 4
3102 extern char *strrchr (const char *__s, int __c)
3103      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1)));
3104
3105
3106 # 282 "/usr/include/string.h" 3 4
3107
3108
3109
3110 extern size_t strcspn (const char *__s, const char *__reject)
3111      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3112
3113
3114 extern size_t strspn (const char *__s, const char *__accept)
3115      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3116 # 315 "/usr/include/string.h" 3 4
3117 extern char *strpbrk (const char *__s, const char *__accept)
3118      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3119 # 342 "/usr/include/string.h" 3 4
3120 extern char *strstr (const char *__haystack, const char *__needle)
3121      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
    __attribute__ ((__nonnull__ (1, 2)));
3122
3123
3124
3125
3126 extern char *strtok (char *__restrict __s, const char *__restrict __delim)
3127      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
    (2)));
3128
3129
3130
3131
3132 extern char *__strtok_r (char *__restrict __s,
3133     const char *__restrict __delim,
```

```
3134     char **__restrict __save_ptr)
3135       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
         (2, 3)));
3136
3137 extern char *strtok_r (char *__restrict __s, const char *__restrict
         __delim,
3138           char **__restrict __save_ptr)
3139       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
         (2, 3)));
3140 # 397 "/usr/include/string.h" 3 4
3141
3142
3143 extern size_t strlen (const char *__s)
3144       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
         __attribute__ ((__nonnull__ (1)));
3145
3146
3147
3148
3149
3150 extern size_t strnlen (const char *__string, size_t __maxlen)
3151       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
         __attribute__ ((__nonnull__ (1)));
3152
3153
3154
3155
3156
3157 extern char *strerror (int __errnum) __attribute__ ((__nothrow__ ,
         __leaf__));
3158
3159 # 427 "/usr/include/string.h" 3 4
3160 extern int strerror_r (int __errnum, char *__buf, size_t __buflen) __asm__
       ("" "__xpg_strerror_r") __attribute__ ((__nothrow__ , __leaf__))
3161
3162                         __attribute__ ((__nonnull__ (2)));
3163 # 445 "/usr/include/string.h" 3 4
3164 extern char *strerror_l (int __errnum, __locale_t __l) __attribute__ ((
         __nothrow__ , __leaf__));
3165
3166
3167
3168
3169
3170 extern void __bzero (void *__s, size_t __n) __attribute__ ((__nothrow__ ,
         __leaf__)) __attribute__ ((__nonnull__ (1)));
3171
3172
3173
3174 extern void bcopy (const void *__src, void *__dest, size_t __n)
3175       __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
         (1, 2)));
3176
3177
```

```
3178 extern void bzero (void *__s, size_t __n) __attribute__ ((__nothrow__ ,
        __leaf__)) __attribute__ ((__nonnull__ (1)));
3179
3180
3181 extern int bcmp (const void *__s1, const void *__s2, size_t __n)
3182      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3183 # 489 "/usr/include/string.h" 3 4
3184 extern char *index (const char *__s, int __c)
3185      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1)));
3186 # 517 "/usr/include/string.h" 3 4
3187 extern char *rindex (const char *__s, int __c)
3188      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1)));
3189
3190
3191
3192
3193 extern int ffs (int __i) __attribute__ ((__nothrow__ , __leaf__))
        __attribute__ ((__const__));
3194 # 534 "/usr/include/string.h" 3 4
3195 extern int strcasecmp (const char *__s1, const char *__s2)
3196      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3197
3198
3199 extern int strncasecmp (const char *__s1, const char *__s2, size_t __n)
3200      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__pure__))
        __attribute__ ((__nonnull__ (1, 2)));
3201 # 557 "/usr/include/string.h" 3 4
3202 extern char *strsep (char **__restrict __stringp,
3203        const char *__restrict __delim)
3204      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3205
3206
3207
3208
3209 extern char *strsignal (int __sig) __attribute__ ((__nothrow__ , __leaf__)
        );
3210
3211
3212 extern char *__stpcpy (char *__restrict __dest, const char *__restrict
        __src)
3213      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3214 extern char *stpcpy (char *__restrict __dest, const char *__restrict __src
        )
3215      __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
        (1, 2)));
3216
3217
3218
```

```
extern char *__stpncpy (char *__restrict __dest,
   const char *__restrict __src, size_t __n)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));
extern char *stpncpy (char *__restrict __dest,
        const char *__restrict __src, size_t __n)
     __attribute__ ((__nothrow__ , __leaf__)) __attribute__ ((__nonnull__
   (1, 2)));
# 644 "/usr/include/string.h" 3 4

# 9 "headers/all_headers.h" 2
# 1 "headers/sen_basic_type.h" 1
# 1 "headers/sen_object.h" 1









# 7 "headers/sen_object.h"
struct Sen_object_vtable;
typedef struct Sen_object_vtable Sen_object_vtable;

struct Sen_object_class;
typedef struct Sen_object_class Sen_object_class;

struct Sen_object;
typedef struct Sen_object Sen_object;

struct Sen_object_vtable {
    void (*print) (Sen_object *);
    Sen_object *(*copy) (Sen_object *);
};



struct Sen_object_class {
    Sen_object_vtable *tablep;
};



struct Sen_object {

# 30 "headers/sen_object.h" 3 4
   _Bool
# 30 "headers/sen_object.h"
        bound;
    Sen_object_class *classp;
};

extern Sen_object_class Sen_object_class_;
extern Sen_object_vtable Sen_object_vtable_;
```

```
3271
3272 void print_object (Sen_object *);
3273 # 2 "headers/sen_basic_type.h" 2
3274
3275
3276
3277
3278 struct Sen_basic_type_vtable;
3279 typedef struct Sen_basic_type_vtable Sen_basic_type_vtable;
3280
3281 struct Sen_basic_type_class;
3282 typedef struct Sen_basic_type_class Sen_basic_type_class;
3283
3284 struct Sen_basic_type;
3285 typedef struct Sen_basic_type Sen_basic_type;
3286
3287 typedef enum {BOOL, INT, STR, UNK} Type;
3288
3289 struct Sen_basic_type_vtable {
3290     void (*print) (Sen_object *);
3291     Sen_basic_type *(*construct) (void *);
3292     void *(*get_val) (Sen_basic_type *);
3293     void *(*set_val) (Sen_basic_type *, void *);
3294     Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
3295 };
3296
3297 struct Sen_basic_type_class {
3298     Sen_object_class *superp;
3299     Sen_basic_type_vtable *tablep;
3300     Type type;
3301 };
3302
3303 struct Sen_basic_type {
3304
3305 # 32 "headers/sen_basic_type.h" 3 4
3306    _Bool
3307 # 32 "headers/sen_basic_type.h"
3308         bound;
3309     Sen_basic_type_class *classp;
3310     Sen_object *superp;
3311 };
3312
3313 extern Sen_basic_type_class Sen_basic_type_class_;
3314 extern Sen_basic_type_vtable Sen_basic_type_vtable_;
3315
3316 void * get_val_basic_type (Sen_basic_type *);
3317 void * set_val_basic_type (Sen_basic_type *, void *);
3318 Sen_basic_type * add_basic_type (Sen_basic_type *, Sen_basic_type *);
3319 # 10 "headers/all_headers.h" 2
3320 # 1 "headers/sen_int.h" 1
3321 # 1 "headers/sen_basic_type.h" 1
3322 # 2 "headers/sen_int.h" 2
3323
3324
```

```
3325
3326
3327 struct Sen_int_vtable;
3328 typedef struct Sen_int_vtable Sen_int_vtable;
3329
3330 struct Sen_int_class;
3331 typedef struct Sen_int_class Sen_int_class;
3332
3333 struct Sen_int;
3334 typedef struct Sen_int Sen_int;
3335
3336 struct Sen_int_vtable {
3337     void (*print) (Sen_object *);
3338     void *(*get_val) (Sen_basic_type *);
3339     void *(*set_val) (Sen_basic_type *, void *);
3340     Sen_int *(*construct) (int);
3341     void (*destruct) (Sen_int *);
3342     Sen_int *(*copy) (Sen_int *);
3343     Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
3344 };
3345
3346 struct Sen_int_class {
3347     Sen_basic_type_class *superp;
3348     Sen_int_vtable *tablep;
3349     Type type;
3350 };
3351
3352 struct Sen_int {
3353
3354 # 32 "headers/sen_int.h" 3 4
3355    _Bool
3356 # 32 "headers/sen_int.h"
3357         bound;
3358     Sen_int_class *classp;
3359     Sen_basic_type *superp;
3360     int val;
3361 };
3362
3363 extern Sen_int_class Sen_int_class_;
3364 extern Sen_int_vtable Sen_int_vtable_;
3365
3366 void print_int (Sen_object *);
3367 Sen_int * construct_int (int);
3368 void *get_val_int (Sen_basic_type *);
3369 void *set_val_int (Sen_basic_type *, void *);
3370 # 11 "headers/all_headers.h" 2
3371
3372 # 1 "headers/sen_bool.h" 1
3373 # 1 "headers/sen_basic_type.h" 1
3374 # 2 "headers/sen_bool.h" 2
3375
3376
3377
3378
```

```
struct Sen_bool_vtable;
typedef struct Sen_bool_vtable Sen_bool_vtable;

struct Sen_bool_class;
typedef struct Sen_bool_class Sen_bool_class;

struct Sen_bool;
typedef struct Sen_bool Sen_bool;

struct Sen_bool_vtable {
    void (*print) (Sen_object *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_bool *(*construct) (
# 19 "headers/sen_bool.h" 3 4
                            _Bool
# 19 "headers/sen_bool.h"
                                 );
    void (*destruct) (Sen_bool *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_bool_class {
    Sen_basic_type_class *superp;
    Sen_bool_vtable *tablep;
    Type type;
};

struct Sen_bool {

# 31 "headers/sen_bool.h" 3 4
   _Bool
# 31 "headers/sen_bool.h"
        bound;
    Sen_bool_class *classp;
    Sen_basic_type *superp;

# 34 "headers/sen_bool.h" 3 4
   _Bool
# 34 "headers/sen_bool.h"
        val;
};

extern Sen_bool_class Sen_bool_class_;
extern Sen_bool_vtable Sen_bool_vtable_;

void print_bool (Sen_object *);
Sen_bool * construct_bool (
# 41 "headers/sen_bool.h" 3 4
                           _Bool
# 41 "headers/sen_bool.h"
                                );
void *get_val_bool (Sen_basic_type *);
void *set_val_bool (Sen_basic_type *, void *);
```

```
# 13 "headers/all_headers.h" 2
# 1 "headers/sen_string.h" 1
# 1 "headers/sen_basic_type.h" 1
# 2 "headers/sen_string.h" 2




struct Sen_string_vtable;
typedef struct Sen_string_vtable Sen_string_vtable;

struct Sen_string_class;
typedef struct Sen_string_class Sen_string_class;

struct Sen_string;
typedef struct Sen_string Sen_string;

struct Sen_string_vtable {
    void (*print) (Sen_object *);
    void *(*get_val) (Sen_basic_type *);
    void *(*set_val) (Sen_basic_type *, void *);
    Sen_string *(*construct) (char *);
    void (*destruct) (Sen_string *);
    Sen_basic_type *(*add) (Sen_basic_type *, Sen_basic_type *);
};

struct Sen_string_class {
    Sen_basic_type_class *superp;
    Sen_string_vtable *tablep;
    Type type;
};

struct Sen_string {

# 31 "headers/sen_string.h" 3 4
   _Bool
# 31 "headers/sen_string.h"
        bound;
    Sen_string_class *classp;
    Sen_basic_type *superp;
    char *val;
};

extern Sen_string_class Sen_string_class_;
extern Sen_string_vtable Sen_string_vtable_;

void print_string (Sen_object *);
Sen_string * construct_string (char *);
void *get_val_string (Sen_basic_type *);
void *set_val_string (Sen_basic_type *, void *);
# 14 "headers/all_headers.h" 2
# 1 "headers/sen_array.h" 1

# 1 "headers/sen_basic_type.h" 1
```

```
# 3 "headers/sen_array.h" 2
# 1 "headers/sen_int.h" 1
# 1 "headers/sen_basic_type.h" 1
# 2 "headers/sen_int.h" 2
# 4 "headers/sen_array.h" 2
# 12 "headers/sen_array.h"
struct Sen_array_vtable;
typedef struct Sen_array_vtable Sen_array_vtable;

struct Sen_array_class;
typedef struct Sen_array_class Sen_array_class;

struct Sen_array;
typedef struct Sen_array Sen_array;

struct Sen_array_vtable {
    void (*print) (Sen_object *);
    Sen_array *(*construct) (Sen_object **, int);
    void (*destruct) (Sen_array *);
    Sen_object *(*access) (Sen_array *, Sen_int *);
    Sen_array *(*concat) (Sen_array *, Sen_array *);
};

struct Sen_array_class {
    Sen_object_class *superp;
    Sen_array_vtable *tablep;
};

struct Sen_array {

# 35 "headers/sen_array.h" 3 4
   _Bool
# 35 "headers/sen_array.h"
        bound;
    Sen_array_class *classp;
    Sen_object *superp;
    void **arr;
    int len;
    char print_sep;
};

extern Sen_array_class Sen_array_class_;
extern Sen_array_vtable Sen_array_vtable_;

Sen_array *construct_array (Sen_object **, int);
Sen_object *access_array (Sen_array *, Sen_int *);
Sen_array *concat_array (Sen_array *, Sen_array *);
# 15 "headers/all_headers.h" 2
# 2 "main.c" 2

int main() {
# 29 "main.c"
    __auto_type s = ((Sen_string*) construct_string("tttesting "));
    s->bound=
```

```
# 30 "main.c" 3 4
            1
# 30 "main.c"
                ;
    __auto_type ss = ((Sen_string*) construct_string("hooray!!\n"));
    ss->bound=
# 32 "main.c" 3 4
             1
# 32 "main.c"
                 ;
    Sen_int *arr_[2] = {((Sen_int*) construct_int(100)), ((Sen_int*)
   construct_int(50))};
    printf ("%d\n\n", (int)sizeof(arr_));


    { typeof(((Sen_string *) ss)) __temp__ = ((Sen_string *) ss); __temp__
   ->classp->tablep->print(((Sen_object *) __temp__)); };
    { typeof(((Sen_string *)s->classp->tablep->add((Sen_basic_type *)s, (
   Sen_basic_type *)ss))) __temp__ = ((Sen_string *)s->classp->tablep->add
   ((Sen_basic_type *)s, (Sen_basic_type *)ss)); __temp__->classp->tablep
   ->print(((Sen_object *) __temp__)); };

    ((Sen_string *) s)->classp->tablep->destruct(((Sen_string *) s));
    ((Sen_string *) ss)->classp->tablep->destruct(((Sen_string *) ss));

    Sen_array *arr = (Sen_array_vtable_.construct(arr_, (sizeof((arr_)) /
   sizeof((arr_)[0]))));
    printf("%d\n", (int)sizeof((arr->arr)[0]));

    return 0;
}
```

./cast-test.snt

```
@setup
{

group A() {
    int x;
    int y;
    int z;
    func void __init__() {
        1;
        2;
        3;
    }
};

group B(A()) {
    int x;
    bool a;
    bool b;
};

}
```

```

@turns
{

func void begin () {

}

func void test () {}

}
```

./cast.ml

```
(**
  * C Abstract Syntax Tree (CAST)
  *
  * The CAST is a slightly modifed version of the SAST, where inherited
  * group attributes have been copied, in order, into groups.
  *
  * Also, the second argument to Group is filled in with Some(gdcl).
  *)

open Types
open Sast

let rec get_attributes g = match g.extends with
    None -> g.attributes
  | Some(par) ->
      let pa = get_attributes par in
      pa @ g.attributes

let rec order_attrib = function
    [] -> []
  | g :: rest ->
      let a = get_attributes g in
      let new_g = {g with attributes = a} in
      new_g :: order_attrib rest

let find_group g s =
  if List.exists (fun x -> x.gname = s) g then
    List.find (fun x -> x.gname = s) g
  else
    (* This is an internal dummy gdecl with name "" *)
    { gname = ""; extends = None; par_actuals = None;
      attributes = []; methods = [] }

let tag_groups_vdcl g v = match v.vtype with
    Group(s, _) ->
      let gdcl = find_group g s in
      { v with vtype = Group(s, Some(gdcl)) }
  | _ -> v

let rec tag_groups_field g = function
```

```
    Var(vd) -> Var(tag_groups_vdcl g vd)
  | Attrib(vd1, vd2) ->
      let vd1 = tag_groups_vdcl g vd1 in
      let vd2 = tag_groups_vdcl g vd2 in
      Attrib(vd1, vd2)
  | Fun(fd) -> Fun(tag_groups_func g fd)
  | Method(vd, fd) ->
      let vd = tag_groups_vdcl g vd in
      let fd = tag_groups_func g fd in
      Method(vd, fd)
  | Grp(gd) -> Grp(gd)
  | This -> This

and tag_groups_listlit g = function
    Elems(el, name) -> Elems(List.map (tag_groups_expr g) el, name)
  | EmptyList -> EmptyList

and tag_groups_expr g e =
  let detail, typ = e in
  let typ = match typ with
      Group(s, _) -> Group(s, Some(find_group g s))
    | _ as x -> x
  in
  let detail =
    match detail with
        ListLiteral(ll) ->
          let ll = tag_groups_listlit g ll in
          ListLiteral(ll)
      | Field(fe) -> Field(tag_groups_field g fe)
      | Binop(e1, op, e2) ->
          let e1 = tag_groups_expr g e1 in
          let e2 = tag_groups_expr g e2 in
          Binop(e1, op, e2)
      | Assign(fe, e) ->
          let fe = tag_groups_field g fe in
          let e = tag_groups_expr g e in
          Assign(fe, e)
      | Call(vd_opt, fd, el) ->
          let vd_opt = match vd_opt with
              Some(vd) -> Some(tag_groups_vdcl g vd)
            | None -> None
          in
          let fd = tag_groups_func g fd in
          let el = List.map (tag_groups_expr g) el in
          Call(vd_opt, fd, el)
      | Element(e1, e2) ->
          let e1 = tag_groups_expr g e1 in
          let e2 = tag_groups_expr g e2 in
          Element(e1, e2)
      | Uminus(e) -> Uminus(tag_groups_expr g e)
      | Not(e) -> Not(tag_groups_expr g e)
      | Remove(e) ->
          let e = tag_groups_expr g e in
          Remove(e)
```

```
95        | Place(e) ->
96            let e = tag_groups_expr g e in
97            Place(e)
98        | _ as x -> x
99    in
100   detail, typ
101
102 and tag_groups_stmt g = function
103     Block(scope, sl) -> Block(scope, List.map (tag_groups_stmt g) sl)
104   | Expression(e) -> Expression(tag_groups_expr g e)
105   | Return(e) -> Return(tag_groups_expr g e)
106   | Break -> Break
107   | Continue -> Continue
108   | End -> End
109   | Pass(fdcl, e) ->
110       let fdcl = tag_groups_func g fdcl in
111       let e = tag_groups_expr g e in
112       Pass(fdcl, e)
113   | If(e, s1, e_opt, s2) ->
114       let e = tag_groups_expr g e in
115       let e_opt = match e_opt with
116           None -> None
117         | Some(expr) -> Some(tag_groups_expr g expr)
118       in
119       let s1 = tag_groups_stmt g s1 in
120       let s2 = tag_groups_stmt g s2 in
121       If(e, s1, e_opt, s2)
122   | For(vd, el, s) ->
123       let vd = tag_groups_vdcl g vd in
124       let el = List.map (tag_groups_expr g) el in
125       let s = tag_groups_stmt g s in
126       For(vd, el, s)
127   | While(e, s) ->
128       let e = tag_groups_expr g e in
129       let s = tag_groups_stmt g s in
130       While(e, s)
131
132 and tag_groups_func g = function
133     BasicFunc(f) ->
134       let t = match f.ftype with
135           Group(s, _) -> Group(s, Some(find_group g s))
136         | _ -> f.ftype
137       in
138       let l = List.map (tag_groups_vdcl g) f.locals in
139       let fl = List.map (tag_groups_vdcl g) f.formals in
140       let b = List.map (tag_groups_stmt g) f.body in
141       BasicFunc({ f with ftype = t; locals = l; formals = fl; body = b })
142   | AssertFunc(f) ->
143       let l = List.map (tag_groups_vdcl g) f.alocals in
144       let fl = List.map (tag_groups_vdcl g) f.aformals in
145       let b = List.map (tag_groups_stmt g) f.abody in
146       AssertFunc({ f with alocals =l; aformals = fl; abody = b })
147
148 and tag_groups_grp g gd =
```

```
  let a = List.map (tag_groups_vdcl g) gd.attributes in
  let m = List.map (tag_groups_func g) gd.methods in
  let pa = match gd.par_actuals with
      Some(el) -> Some(List.map (tag_groups_expr g) el)
    | None -> None
  in
  let ex = match gd.extends with
      Some(par) -> Some(tag_groups_grp g par)
    | None -> None
  in
  { gd with attributes = a; methods = m; par_actuals = pa; extends = ex }

let tag_program program =
  let (v, f, g), turns = program in
  let v = List.map (tag_groups_vdcl g) v in
  let f = List.map (tag_groups_func g) f in
  let g = List.map (tag_groups_grp g) g in
  let turns = List.map (tag_groups_func g) turns in
  (v, f, g), turns

let rec get_ll_type = function
    Elems(el, _) ->
      let _, typ = List.hd el in
      typ
  | EmptyList -> List_t(Void)

let fix_ll_lit_expr vars e =
  let detail, typ = e in
  let v_base =
    { vname = ""; vtype = typ; vinit = Some(e); vloop = false }
  in
  match detail with
    IntLiteral(i, name) -> { v_base with vname = name } :: vars
  | StrLiteral(s, name) -> { v_base with vname = name } :: vars
  (* | ListLiteral(ll) ->
      literals.count <- literals.count + 1;
      { v_base with vname = "__elem__" ^ literals.count; vtype = Int } ::
   vars *)
  | BoolLiteral(bl, name) -> { v_base with vname = name } :: vars
  | _ -> vars

let rec fix_ll vars = function
    Elems(el, name) ->
      let vars = List.fold_left fix_ll_lit_expr vars el in
      let _, typ = List.hd el in
      let typ = List_t(typ) in
      let vdcl =
        { vname = name;
          vtype = typ;
          vinit = Some(ListLiteral(Elems(el, name)), typ);
          vloop = false }
      in
      vdcl :: vars
  | EmptyList -> vars
```

```
202
203 and fix_ll_expr vars (expr_detail, expr_typ) = match expr_detail with
204     ListLiteral(ll) -> fix_ll vars ll
205   | Binop(e1, op, e2) ->
206       let vars = fix_ll_expr vars e1 in
207       let vars = fix_ll_expr vars e2 in
208       vars
209   | Assign(fe, e) -> fix_ll_expr vars e
210   | Call(vd_opt, fd, el) -> List.fold_left fix_ll_expr vars el
211   | Element(e1, e2) ->
212       let vars = fix_ll_expr vars e1 in
213       let vars = fix_ll_expr vars e2 in
214       vars
215   | Remove(e) -> fix_ll_expr vars e
216   | Place(e) -> fix_ll_expr vars e
217   | Uminus(e) -> fix_ll_expr vars e
218   | Not(e) -> fix_ll_expr vars e
219   | _ -> vars
220
221 let rec fix_ll_stmt vars = function
222     Block(_, sl) -> List.fold_left fix_ll_stmt vars sl
223   | Expression(e) -> fix_ll_expr vars e
224   | Return(e) -> fix_ll_expr vars e
225   | Pass(fd, e) -> fix_ll_expr vars e
226   | If(e, s1, e_opt, s2) ->
227       let vars = fix_ll_expr vars e in
228       let vars = fix_ll_stmt vars s1 in
229       let vars = match e_opt with
230           None -> vars
231         | Some(expr) -> fix_ll_expr vars expr in
232       let vars = fix_ll_stmt vars s2 in
233       vars
234   | For(vd, el, s) ->
235       let vars = List.fold_left fix_ll_expr vars el in
236       let vars = fix_ll_stmt vars s in
237       vars
238   | While(e, s) ->
239       let vars = fix_ll_expr vars e in
240       let vars = fix_ll_stmt vars s in
241       vars
242   | _ -> vars
243
244 let fix_ll_vdcl vars v = match v.vinit with
245     None -> v :: vars
246   | Some(e) -> (match e with
247       ListLiteral(ll), _ ->
248         let vdcl, vars = match ll with
249             Elems(el, name) ->
250               let vars = List.fold_left fix_ll_lit_expr vars el in
251               [{ v with vname = name }], vars
252           | EmptyList ->
253               [], vars
254         in
255         v :: vdcl @ vars
```

```
256     | _ -> v :: vars)
257
258 let fix_ll_vdcls vars =
259   let new_vars = List.fold_left fix_ll_vdcl [] vars in
260   new_vars (* @ vars *)
261
262 let fix_ll_fdcl = function
263     BasicFunc(f) ->
264       let new_vars = List.fold_left fix_ll_vdcl [] f.locals in
265       let new_vars = List.fold_left fix_ll_stmt new_vars f.body in
266       BasicFunc({ f with locals = List.rev new_vars (* @ f.locals *) })
267   | AssertFunc(f) ->
268       let new_vars = List.fold_left fix_ll_vdcl [] f.alocals in
269       let new_vars = List.fold_left fix_ll_stmt new_vars f.abody in
270       AssertFunc({ f with alocals = List.rev new_vars (* @ f.alocals *) })
271
272 let fix_ll_gdcl g =
273   let new_vars = List.fold_left fix_ll_vdcl [] g.attributes in
274   let g = { g with attributes = List.rev new_vars (* @ g.attributes *) }
     in
275   let m = List.map fix_ll_fdcl g.methods in
276   { g with methods = m }
277
278 let correct_listlit program =
279   let (v, f, g), turns = program in
280   let v = List.rev (fix_ll_vdcls v) in
281   let f = List.map fix_ll_fdcl f in
282   let g = List.map fix_ll_gdcl g in
283   let turns = List.map fix_ll_fdcl turns in
284   (v, f, g), turns
285
286 let build_cast (program : Types.program) =
287   let (v, f, g), turns = program in
288   let g = order_attrib g in
289   let program = tag_program ((v, f, g), turns) in
290   let program = correct_listlit program in
291   program
292
293 (**
294  * ----------------------------------------
295  * Functions to convert the CAST to a string
296  * ----------------------------------------
297  *)
298
299 let rec string_of_t = function
300     Int -> "int"
301   | Bool -> "bool"
302   | Str -> "str"
303   | Void -> "void"
304   | List_t(vt) ->
305       "list[" ^ string_of_t vt ^ "]"
306   | Group(s, _) -> s
307
308 let string_of_field = function
```

```
309     Var(v) -> v.vname
310   | Fun(f) ->
311       (match f with
312          BasicFunc(x) -> x.fname
313        | AssertFunc(x) -> x.aname)
314   | Attrib(v1, v2) -> v1.vname ^ "." ^ v2.vname
315   | Method(v, f) -> v.vname ^ "." ^
316       (match f with
317           BasicFunc(x) -> x.fname
318         | AssertFunc(x) -> x.aname)
319   | Grp(g) -> g.gname
320   | This -> "this"
321
322 let rec string_of_list_lit = function
323     EmptyList -> "[]"
324   | Elems(e, n) ->
325       "[" ^ String.concat ", " (List.map string_of_expression e) ^ "]"
326
327 and string_of_expr_detail = function
328     IntLiteral(l, name) -> string_of_int l
329   | Field(f) -> string_of_field f
330   | Binop(e1, o, e2) ->
331       string_of_expression e1 ^ " " ^
332       (match o with
333         Add -> "+" | Sub -> "-" | Mult -> "*" | Div -> "/"
334       | Equal -> "==" | Neq -> "!="
335       | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">="
336       | Mod -> "%"
337       | And -> "and" | Or -> "or" ) ^ " " ^
338       string_of_expression e2
339   | Assign(f, e) -> string_of_field f ^ " = " ^ string_of_expression e
340   | Call(vopt, f, el) ->
341       let par =
342         (match vopt with
343             None -> ""
344           | Some(v) -> v.vname) in
345       par ^ "." ^ string_of_field(Fun(f)) ^
346       "(" ^ String.concat ", " (List.map string_of_expression el) ^ ")"
347   | Noexpr -> "<Noexpr>"
348   | StrLiteral(s, name) -> Ast.escaped_string s
349   | Uminus(e) -> "-" ^ string_of_expression e
350   | Not(e) -> "not" ^ string_of_expression e
351   | Element(e1, e2) ->
352     string_of_expression e1 ^ "[" ^ string_of_expression e2 ^ "]"
353   | ListLiteral(l) -> string_of_list_lit l
354   | BoolLiteral(b, name) -> (match b with True -> "True" | False -> "False
    ")
355   | VoidLiteral -> "None"
356   | Place(e) -> string_of_expression e
357   | Remove(e) -> string_of_expression e
358
359 and string_of_expression e =
360   let detail, _ = e in
361   string_of_expr_detail detail
```

```
362
363 let string_of_vinit = function
364     None -> "<None>"
365   | Some(e) -> string_of_expression e
366
367 let string_of_vdecl vdecl =
368   "var_decl = { " ^
369   "vtype: " ^ string_of_t vdecl.vtype ^ "; " ^
370   "vname: " ^ vdecl.vname ^ "; " ^
371   "vinit: " ^ string_of_vinit vdecl.vinit ^
372   " }"
373
374 let rec string_of_stmt = function
375     Block(symbols, stmts) ->
376       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
377   | Expression(e) -> string_of_expression e ^ ";\n";
378   | Return(e) -> "return " ^ string_of_expression e ^ ";\n";
379   | If(e, s1, e_opt, s2) ->  "if (" ^ string_of_expression e ^ ")\n" ^
380       string_of_stmt s1 ^
381       (match e_opt with
382         None -> "else\n"
383       | Some(expr) -> string_of_expression e) ^
384       string_of_stmt s2
385   | For(vd, elist, s) ->
386       "for (" ^ string_of_vdecl vd  ^ " in " ^
387             "{\n" ^ String.concat ", " (List.map string_of_expression
    elist) ^ "}\n" ^
388           ") " ^ string_of_stmt s
389   | While(e, s) -> "while (" ^ string_of_expression e ^ ") {\n" ^
    string_of_stmt s ^ "\n}\n"
390   | Pass(e, s) -> "pass (" ^ string_of_field(Fun(e)) ^ ", " ^
    string_of_expression s ^ ")\n"
391   | Break -> "break;\n"
392   | Continue -> "continue;\n"
393   | End -> "end();\n"
394
395 let string_of_basic_fdecl fdecl =
396   "basic_func_decl = {\n" ^
397   "  ftype: " ^ string_of_t fdecl.ftype ^ "; " ^
398   "fname: " ^ fdecl.fname ^ "; " ^
399   "turns_func: " ^ string_of_bool fdecl.turns_func ^ ";\n" ^
400   "  formals: {\n    " ^
401     String.concat ";\n    " (List.map string_of_vdecl fdecl.formals) ^ "\n
      }\n" ^
402   "  locals: {\n    " ^
403     String.concat ";\n    " (List.map string_of_vdecl fdecl.locals) ^ "\n
      }\n" ^
404   "  body: {\n" ^
405     String.concat "" (List.map string_of_stmt fdecl.body) ^
406   "}\n"
407
408 let string_of_assert_decl fdecl =
409   "assert_func_decl = {\n" ^
410   "  aname: " ^ fdecl.aname ^ "; " ^
```

```
  "  a_turns_func: " ^ string_of_bool fdecl.a_turns_func ^ ";\n" ^
  "  aformals: {\n    " ^
    String.concat ";\n    " (List.map string_of_vdecl fdecl.aformals) ^ "\n  }\n" ^
  "  alocals: {\n    " ^
    String.concat ";\n    " (List.map string_of_vdecl fdecl.alocals) ^ "\n  }\n" ^
  "  abody: {\n" ^
    String.concat "" (List.map string_of_stmt fdecl.abody) ^
  "}\n"

let string_of_fdecl = function
    BasicFunc(f) -> string_of_basic_fdecl f
  | AssertFunc(f) -> string_of_assert_decl f

let string_of_gdecl gdecl =
  "group " ^ gdecl.gname ^ "(" ^
      (match gdecl.extends with
           Some(par) -> par.gname ^
              (match gdecl.par_actuals with
                  Some(acts) ->
                    "(" ^ String.concat ", " (List.map string_of_expression acts) ^ ")"
                | None -> "")
         | None -> "") ^ ")\n{\n" ^
  String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") gdecl.attributes) ^
  String.concat "" (List.map string_of_fdecl gdecl.methods) ^
  "};\n"

let string_of_setup s =
  let vars, funcs, groups = s in
  "@setup {\n\n" ^
  String.concat "" (List.map (fun v -> string_of_vdecl v ^ ";\n") vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs) ^
  String.concat "\n" (List.map string_of_gdecl groups) ^
  "\n}\n"

let string_of_turns t =
  "@turns {\n\n" ^
  String.concat "\n" (List.map string_of_fdecl t) ^
  "\n}\n"

let string_of_program (program : program) =
  let s, t = program in
  string_of_setup s ^ string_of_turns t
```

./compile.ml

```
open Types
open Sast

type counter = {
  mutable i : int
```

```
}

let count = { i = 0 }

let tmp_formal_prefix = "__tmp_form__"

let prefix_name n =
  "snt_" ^ n

let senet_header =
  "#include <stdbool.h>" ^ "\n" ^
  "#include <stdio.h>" ^ "\n" ^
  "#include <stdlib.h>" ^ "\n" ^
  "#include <string.h>\n" ^
  "#include \"temp/sen_linked_list.h\"\n" ^
  "#include \"temp/sen_print_base_grps.h\"\n" ^
  "#include \"temp/sen_init_base_grps.h\"\n" ^
  "#include \"temp/sen_read.h\"\n" ^
    "\n" ^
  "struct SENET_NONE {\n" ^
  "  } SENET_NONE;\n" ^
    "\n" ^
  "struct Sen_list snt_SEN_EMPTY_LIST;\n" ^
    "\n" ^
  "char *SENET_STR_CONCAT(char* s1, char* s2) {\n" ^
  "  char *temp = (char * ) malloc(strlen(s1)+ strlen(s2) +1);\n" ^
  "  strcpy(temp, s1);\n" ^
  "  strcat(temp, s2);\n" ^
  "  return temp;\n" ^
  "}\n" ^
    "\n" ^
  "void (*CUR_TURN)();" ^ "\n" ^
  "int snt_PLAYER_ON_MOVE = 0;" ^ "\n" ^
    "\n"

let senet_footer =
  "int main() {\n" ^
  "  CUR_TURN = &snt_begin;\n" ^
  "  snt_PLAYER_ON_MOVE = 0;\n" ^
  "  while (true) {\n" ^
  "    CUR_TURN();\n" ^
  "  }\n" ^
  "  return 0;\n" ^
  "}\n"

let binop_to_c = function
   Add -> "+" | Sub -> "-" | Mult -> "*" | Div -> "/" | Mod -> "%"
 | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">="
 | And -> "&&" | Or -> "||"
 | Equal -> "==" | Neq -> "!="

let id_type_to_c = function
    Int -> "int "
  | Bool -> "bool "
```

```
  | Str -> "char* "
  | Void -> "void "
  | List_t(typ) -> "struct Sen_list "
  | Group(s, _) -> "struct " ^ prefix_name s ^ " "

let rec field_to_c = function
    Var(v) ->
      if v.vloop then
        "*(" ^ prefix_name v.vname ^ " + __cnt__" ^ prefix_name v.vname ^
   ")"
      else
        prefix_name v.vname
  | Fun(f) -> prefix_name
      (match f with
         BasicFunc(x) -> x.fname
       | AssertFunc(x) -> x.aname)
  | Grp(g) -> prefix_name g.gname
  | Attrib(par, child) ->
    let par_name, deref_op =
      if par.vname = "this" then
        "(*this)", "."
      else if par.vloop then
        "(" ^ prefix_name par.vname ^ " + __cnt__" ^ prefix_name par.vname
    ^ ")", "->"
      else
        prefix_name par.vname, "."
    in
    par_name ^ deref_op ^ prefix_name child.vname
  | Method(v, f) -> raise (SemError ("Internal error: Method matched in
   field_to_c(), use Call instead"))
  | This -> "(*this)"

let rec function_call_to_c = function
    BasicFunc(f) -> f.fname
  | AssertFunc(f) -> f.aname

let function_group_method = function
    BasicFunc(f) -> f.group_method
  | AssertFunc(f) -> f.a_group_method

let is_built_in_func = function
    BasicFunc(f) -> f.f_is_built_in
  | AssertFunc(f) -> f.a_is_built_in

let rec printf (detail, typ) =
  let e_c_string = expression_to_c detail in
  match typ with
     Bool -> "printf(\"%s\", " ^ e_c_string ^ " ? \"true\" : \"false\""  ^
    ")"
   | Int -> "printf(\"%d\", " ^ e_c_string  ^ ")"
   | Str -> "printf(\"%s\", " ^ e_c_string  ^ ")"
   | Group(x, _) ->
     "printf(\"%s\", " ^ prefix_name x ^ "_" ^ prefix_name "__repr__" ^
     "((struct " ^ prefix_name x  ^ "*) " ^ "&" ^ e_c_string ^ ")" ^ ")"
```

```
110   | Void -> "printf(" ^"\"None\""  ^ ")"
111   | List_t(l_typ) ->
112     let tmp_var, list_id = match detail with
113         Field(_) -> "", e_c_string
114       | Call(vd_opt, fd, _) ->
115           let tmp_name =
116             count.i <- count.i + 1;
117             "__tp__" ^ string_of_int count.i
118           in
119           id_type_to_c typ ^ tmp_name ^ " = " ^ e_c_string,
120           tmp_name
121       | _ -> "", prefix_name e_c_string
122     in
123     match l_typ with
124       | Group(x, g) ->
125           "printGroupList(&" ^ list_id ^ ", " ^
126           prefix_name x ^ "_" ^ prefix_name "__repr__" ^ ")"
127       | _ ->
128         let func = match l_typ with
129             Int -> "printInt"
130           | Str -> "printStr"
131           | Bool -> "printBool"
132           | Void -> "printEmptyList"
133           | Group(_, _) ->
134               raise (SemError ("Internal error: printList call with Group"
    ))
135           | List_t(_) ->
136               raise (SemError ("Internal error: printList call with List_t
    "))
137         in
138         tmp_var ^ ";\n" ^
139         "printList(&" ^ list_id ^ ", " ^ func ^ ")"
140
141 and formal_to_c is_built_in_func v = match is_built_in_func, v.vtype with
142     false, Group(gname, _) ->
143       id_type_to_c v.vtype ^ "* " ^ tmp_formal_prefix ^ prefix_name v.
    vname
144   | _, List_t(typ) ->
145       id_type_to_c v.vtype ^ "* " ^ tmp_formal_prefix ^ prefix_name v.
    vname
146   | _, _ ->
147       id_type_to_c v.vtype ^ prefix_name v.vname
148
149 and function_group = function
150     BasicFunc(f) -> f.group_method
151   | AssertFunc(f) -> f.a_group_method
152
153 and ll_elem_to_c = function
154     IntLiteral(_, name)
155   | StrLiteral(_, name)
156   | BoolLiteral(_, name) -> prefix_name name
157   | _ as detail -> expression_to_c detail
158
159 and push_ll_to_new_list list_id = function
```

```
    Elems(el, _) ->
      let push_elem_to_new_list (detail, typ) =
        "push(&" ^ prefix_name list_id ^
        ", (void *) &(" ^ ll_elem_to_c detail ^ "))"
      in
      String.concat ";\n" (List.map push_elem_to_new_list el)
  | EmptyList -> ""

and push_to_new_list list_id (det, typ) = match det with
    ListLiteral(ll) -> push_ll_to_new_list list_id ll
  | _ -> raise (SemError "Unsupported expression type to push to a new
   list literal")

and var_decl_to_c v = match v.vtype, v.vinit with
    _, None ->
      id_type_to_c v.vtype ^
      prefix_name v.vname ^ ";"
  | List_t(typ), Some(e) ->
      let detail, _ = e in
      let name = expression_to_c detail in
      id_type_to_c v.vtype ^
      prefix_name v.vname ^
      (if v.vname != "" && String.length v.vname > 5 &&
          String.sub v.vname 0 6 = "__ll__" &&
          String.sub name 0 6 = "__ll__" then
        ";\n" ^
        "new_Sen_list(&" ^ prefix_name name ^
        ", sizeof(" ^ id_type_to_c typ ^ "));\n" ^
        push_to_new_list name e ^ ";"
      else
        " = " ^ prefix_name name ^ ";")
  | _, Some(e) ->
      let detail, typ = e in
      (match typ with
          Group(_, _) ->
            id_type_to_c typ ^ "__tmp__" ^ prefix_name v.vname ^
            " = " ^ expression_to_c detail ^ ";\n" ^
            id_type_to_c v.vtype ^ prefix_name v.vname ^
            " = " ^ "*((" ^ id_type_to_c v.vtype ^ "*) &"
            ^ "__tmp__" ^ prefix_name v.vname ^ ");"
        | _ ->
      id_type_to_c v.vtype ^ prefix_name v.vname ^
      " = " ^ expression_to_c detail ^ ";")

and list_lit_to_c = function
    Elems(el, name) -> name
  | EmptyList -> "SEN_EMPTY_LIST"

and actual_to_c (detail, typ) =
  let e_c_string = expression_to_c detail in
  match typ with
      List_t(l_typ) ->
        (match detail with
            Field(_) -> "&" ^ e_c_string
```

```
          | _ -> "&" ^ prefix_name e_c_string)
    | Group(name, gdcl) -> "&" ^ e_c_string
    | _ ->  e_c_string

and expression_to_c = function
    IntLiteral(i, name) -> string_of_int i
  | StrLiteral(s, name) -> Ast.escaped_string s
  | ListLiteral(ll) -> list_lit_to_c ll
  | BoolLiteral(b, name) ->
      (match b with True -> "true" | False -> "false")
  | VoidLiteral -> "SENET_NONE"
  | Field(fd) -> field_to_c fd
  | Binop(e1, op, e2) ->
    let d1, t1 = e1 and d2, t2 = e2 in
    let d1 = expression_to_c d1 and d2 = expression_to_c d2 in
    (match t1, t2 with
        Int, Int
      | Bool, Bool ->
        "(" ^ d1 ^ " " ^ binop_to_c op ^ " " ^ d2 ^ ")"
      | Str, Str ->
        let eval = match op with
            Equal -> "? 0 : 1"
          | Neq -> "? 1 : 0"
          | _ -> ""
        in
        (match op with
           Equal | Neq -> "(strcmp(" ^ d1 ^ ", " ^ d2 ^ ") " ^ eval ^ ")
"
         | Add -> "SENET_STR_CONCAT(" ^ d1 ^ ", " ^ d2 ^ ")"
         | _ -> raise (SemError "Binop other than +, ==, or != has lhd
and rhs with type Str"))
      | Void, Void ->
        let ans = match op with
            Equal -> True
          | Neq -> False
          | _ ->
            raise (SemError "Binop other than == or != has lhs and rhs
with type Void")
        in
        expression_to_c (BoolLiteral(ans, ""))
      | Group(_, gd1), Group(_, gd2) ->
        let gd1, gd2 = match gd1, gd2 with
            Some(x), Some(y) -> x, y
          | _, _ ->
            raise (SemError "Missing a group decl in expression type")
        in
        let eval = match op with
            Equal -> "? 1 : 0"
          | Neq -> "? 0 : 1"
          | _ ->
            raise (SemError "Binop other than == or != has lhs and rhs
 with type Group(T)")
        in
        let compare_attrib e1 e2 a =
```

```
263             let e1, _ = e1 and e2, _ = e2 in
264             "(" ^ expression_to_c e1 ^ "." ^ prefix_name a.vname ^ " ==
    " ^
265             expression_to_c e2 ^ "." ^ prefix_name a.vname ^ ")"
266           in
267           if gd1.gname == gd2.gname then
268             let attrib = gd1.attributes in
269             "(" ^ "(" ^
270             String.concat " && " (List.map (compare_attrib e1 e2) attrib
    ) ^
271             ") " ^ eval ^ ")"
272           else
273             "false"
274       (* | List_t(t1), List_t(t2) -> *)
275       | _ , _ ->
276         raise (SemError ("Internal error: improper types in binop: " ^
277                          d1 ^ " " ^ binop_to_c op ^ " " ^ d2)))
278   | Assign(fd, e) ->
279     let fd = field_to_c fd
280     and detail, typ = e in
281     fd ^ " = " ^
282     (match detail with
283         ListLiteral(_) -> prefix_name (expression_to_c detail)
284       | _ -> expression_to_c detail)
285   | Call(vopt, fd, el) ->
286     (* let e = List.map (fun (detail, _) -> detail) el in *)
287     let argc = List.length el in
288     let fname = function_call_to_c fd in
289     let gname = function_group_method fd in
290     (* read and stoi take one argument, discard remainder *)
291     if fname = "print" && is_built_in_func fd then
292       String.concat ";\n" (List.map printf el)
293     else if fname = "read" && is_built_in_func fd then
294       let detail, _ = List.hd el in
295       "_snt_read(" ^ expression_to_c detail ^ ")"
296     else if fname = "clear_input" && is_built_in_func fd then
297       "_snt_clear_input()"
298     else if fname = "stoi" && is_built_in_func fd then
299       let detail, _ = List.hd el in
300       "atoi(" ^ expression_to_c detail ^ ")"
301     else if fname = "exit" && is_built_in_func fd then
302       "exit(0)"
303     else if fname = "rand" && is_built_in_func fd then
304       "rand()"
305     else
306     let class_prefix = function_group fd in
307     let class_prefix =
308       if class_prefix <> "" then
309         prefix_name class_prefix ^ "_"
310       else
311         ""
312     in
313     let instance_addr = match vopt with
314         None -> ""
```

```
315         | Some(v) ->
316           let var_name =
317             if v.vloop then
318               "(" ^ prefix_name v.vname ^ " + __cnt__" ^ prefix_name v.
    vname ^ ")"
319             else if v.vname = "this" then
320               "this"
321             else
322               "&" ^ prefix_name v.vname
323           in
324           "(struct " ^ prefix_name gname ^ " *) " ^
325           var_name ^
326           (if argc > 0 then ", " else "")
327        in
328       class_prefix ^
329       field_to_c (Fun(fd)) ^ "(" ^ instance_addr ^
330       String.concat ", " (List.map actual_to_c el) ^ ")"
331   | Element(e1, e2) ->
332       let d1, typ = e1 in
333       let d2, _ = e2 in
334       let c_typ = match typ with
335           List_t(x) -> x
336         | _ as x -> x
337       in
338       "*( (" ^ id_type_to_c c_typ ^ "*) " ^
339       "list_elem(&" ^ expression_to_c d1 ^ ", " ^ expression_to_c d2 ^ ")
    )"
340   | Uminus(e) -> let detail, _ = e in "-(" ^ expression_to_c detail ^ ")"
341   | Not(e) -> let detail, _ = e in "!(" ^ expression_to_c detail ^ ")"
342   | Noexpr -> ""
343   | Remove(e) ->
344       let detail, _ = e in
345       expression_to_c detail
346   | Place(e) ->
347       let detail, _ = e in
348       expression_to_c detail
349
350 let reinitalize_and_push (detail, typ) = match detail with
351     ListLiteral(ll) -> (match ll with
352         EmptyList -> ""
353       | Elems(el, name) ->
354           let e = detail, typ in
355           "new_Sen_list(&" ^ prefix_name name ^
356             ", sizeof(" ^ id_type_to_c typ ^ "));\n" ^
357             push_to_new_list name e ^ ";\n")
358   | _ -> raise (SemError ("non-list passed to reinitalize_and_push"))
359
360 let rec update_ll_expr (detail, typ) = match detail with
361     ListLiteral(ll) -> reinitalize_and_push (detail, typ)
362   (* | Field(fe) -> *)
363   | Binop(e1, op, e2) -> update_ll_expr e1 ^ update_ll_expr e2
364   | Assign(fe, e) -> update_ll_expr e
365   | Call(vd_opt, fd, el) -> String.concat "" (List.map update_ll_expr el)
366   (* | Element(e1, e2) -> *)
```

```
367   (* | Uminus(e) of expression *)
368   | Not(e) -> update_ll_expr e
369   | Remove(e) -> update_ll_expr e
370   | Place(e) -> update_ll_expr e
371   | _ -> ""
372
373 let rec statement_to_c = function
374     Block(scope, slist) ->
375       "  " ^ String.concat "\n  " (List.map statement_to_c slist)
376   | Expression(e) ->
377       let detail, _ = e in
378       update_ll_expr e ^
379       expression_to_c(detail) ^ ";"
380   | Return(e) ->
381       let detail, _ = e in
382       update_ll_expr e ^
383       "return " ^ expression_to_c detail ^ ";"
384   | Break -> "break;"
385   | Continue -> "continue;"
386   | If(e, s1, e_opt, s2) ->
387       let det, _  = e in
388       update_ll_expr e ^
389       "if (" ^ expression_to_c det ^ " ) {\n" ^
390       statement_to_c s1 ^ "\n} " ^
391       (match e_opt with
392           None -> "else {\n"
393         | Some((detail, _)) -> "else if (" ^ expression_to_c detail ^  ")
    {\n") ^
394        statement_to_c s2 ^ "}\n"
395   | For(vd, elist, s) ->
396       let n = List.length elist in
397       let counter = "__cnt__" ^ prefix_name vd.vname in
398       String.concat "" (List.map update_ll_expr elist) ^
399       "int " ^ counter ^ " = 0;\n" ^
400       id_type_to_c vd.vtype ^ prefix_name vd.vname ^ "[] = " ^
401       "{" ^ String.concat ", "
402           (List.map (fun (det, typ) -> expression_to_c det) elist) ^ "};\n
    " ^
403       "for ( ; " ^ counter ^ " < " ^ string_of_int n ^ "; ++" ^ counter ^
    ") {\n" ^
404       statement_to_c s ^ "\n" ^
405       "}\n"
406   | End -> "exit(0);"
407   | Pass(e,s) -> let detaill, _ = s in
408                      "CUR_TURN = &" ^ prefix_name (function_call_to_c e) ^
     ";\n" ^
409                      "snt_PLAYER_ON_MOVE = " ^ expression_to_c detaill ^ "
    ;\n" ^
410                      "return;\n"
411   | While(e, s) ->
412       let detail, _ = e in
413       "while (" ^ expression_to_c detail ^ ") {\n" ^
414       statement_to_c s ^ "\n" ^ "}\n"
415
```

```
let rec assert_stmt_to_c = function
    Block(scope, sl) ->
      "  " ^ String.concat "\n  " (List.map assert_stmt_to_c sl)
  | Expression(e) ->
      let detail, _ = e in
      "if (!(" ^ expression_to_c detail ^ ")) { return false; }\n"
  | If(e, s1, e_opt, s2) ->
      let e, _  = e in
      "if (" ^ expression_to_c e ^ " ) {\n" ^
      assert_stmt_to_c s1 ^ "\n} " ^
      (match e_opt with
          None -> "else {\n"
        | Some(expr) ->
            let det, _ = expr in
            "else if (" ^ expression_to_c det ^  ") {\n") ^
        assert_stmt_to_c s2 ^ "}\n"
  (* | For(vd, el, s) -> *)
  | While(e, s) ->
      let detail, _ = e in
      "while (" ^ expression_to_c detail ^ ") {\n" ^
      assert_stmt_to_c s ^ "\n" ^ "}\n"
  | _ as s -> statement_to_c s

let deref_formal v = match v.vtype with
    Group(gname, _) ->
      id_type_to_c v.vtype ^ prefix_name v.vname ^
      " = *(" ^ tmp_formal_prefix ^ prefix_name v.vname ^ ");"
  | List_t(typ) ->
      id_type_to_c v.vtype ^ prefix_name v.vname ^
      " = *(" ^ tmp_formal_prefix ^ prefix_name v.vname ^ ");"
  | _ -> ""

let basic_func_to_c gprefix f =
    let self_arg =
      if f.group_method <> "" then
        "struct " ^ prefix_name f.group_method ^ " *this"
      else
        ""
    in
    let gprefix =
      if gprefix = "" then "" else gprefix ^ "_"
    in
    (id_type_to_c f.ftype) ^ " " ^ gprefix ^ prefix_name f.fname ^ "(" ^
    self_arg ^
    (if self_arg <> "" && List.length f.formals > 0 then ", " else "") ^
    String.concat ", " (List.map (formal_to_c f.f_is_built_in) f.formals)
   ^ ") {\n" ^
    String.concat "\n" (List.map deref_formal f.formals) ^ "\n" ^
    String.concat "\n" (List.map var_decl_to_c f.locals) ^ "\n" ^
    String.concat "\n" (List.map statement_to_c f.body) ^ "\n" ^
    "}\n"

let assert_func_to_c gprefix f =
  let self_arg =
```

```
      if f.a_group_method <> "" then
        "struct " ^ prefix_name f.a_group_method ^ " *this"
      else
        ""
    in
    let gprefix =
      if gprefix = "" then "" else gprefix ^ "_"
    in
    (id_type_to_c Bool) ^ " " ^ gprefix ^ prefix_name f.aname ^ "(" ^
    self_arg ^
    (if self_arg <> "" && List.length f.aformals > 0 then ", " else "") ^
    String.concat ", " (List.map (formal_to_c f.a_is_built_in) f.aformals)
    ^ ") {\n" ^
    String.concat "\n" (List.map deref_formal f.aformals) ^ "\n" ^
    String.concat "\n" (List.map var_decl_to_c f.alocals) ^ "\n" ^
    String.concat "\n" (List.map assert_stmt_to_c f.abody) ^ "\n" ^
    "}\n"

let func_decl_to_c gprefix = function
    BasicFunc(f) -> basic_func_to_c gprefix f
  | AssertFunc(f) -> assert_func_to_c gprefix f

let group_decl_to_c g =
  let c_name = prefix_name g.gname in
  "struct " ^ c_name ^  "{\n" ^
  "  " ^ String.concat "\n  " (List.map var_decl_to_c g.attributes) ^ "\n"
    ^
  "} " ^ c_name ^ ";\n\n" ^
  String.concat "\n" (List.map (func_decl_to_c c_name) g.methods) ^ "\n"

let setup_to_c s =
  let v, f, g = s in
  String.concat "\n" (List.map var_decl_to_c v) ^ "\n\n" ^
  String.concat "\n" (List.map group_decl_to_c g) ^ "\n" ^
  String.concat "\n" (List.map (func_decl_to_c "") f)

let declare_turn = function
    BasicFunc(f) ->
      (id_type_to_c f.ftype) ^ " " ^
      prefix_name f.fname ^ "(" ^
      String.concat ", " (List.map var_decl_to_c f.formals) ^ ");"
  | AssertFunc(f) -> raise (SemError ("Assert function declared in turns:
   " ^
                                      f.aname))

let turns_to_c t =
  String.concat "\n" (List.map declare_turn t) ^ "\n\n" ^
  String.concat "\n" (List.map (func_decl_to_c "") t)

let senet_to_c (s, t) =
    "// @senet_header\n" ^
    senet_header ^ "\n" ^
    "// @setup\n" ^
    setup_to_c s ^ "\n" ^
```

```
    "// @turns\n" ^
    turns_to_c t ^ "\n" ^
    "// @senet_footer\n" ^
    senet_footer

let translate (program : Types.program) =
    let outfile = open_out "output.c" in
    let ctext = senet_to_c program in
    output_string outfile ctext
```

./examples/chance.snt

```
# This is a game of intense skill and unparalleled intrigue
# Two players race against clock to reach the other side of a treacherous
   40-cell array
# Who will win? Only Lady Luck can tell.


@setup
{
int left = 0;
int right = 39;
group myline(Line(40)) {
    func bool won(int player) {
        if (this.owns((1-player)*39) == player) {
            return True;
        }
        return False;
    }

    func str __repr__() {
        str ret = "|\n";
        int i = 39;
        group Piece temp;
        while (i>=0) {
              temp = this.cells[i];
              ret = "|" + temp.__repr__() + ret;
              i = i - 1;
        }
        return ret;
    }
};

group Mark(Piece) {
    func group Mark __init__(str s, int player) {
        this.s = s;
        this.owner = player;
        return this;
    }
};


int N_PLAYERS = 2;

group myline ml;
```

```
43
44 }
45
46
47 @turns
48 {
49
50 func void begin() {
51      group Mark m0 = Mark("X", 0);
52      group Mark m1 = Mark("O", 1);
53      ml = myline();
54      m0 >> ml >> [0];
55      m1 >> ml >> [39];
56      pass(prompt, 0);
57 }
58
59 func void prompt() {
60     int a;
61     int c;
62     int next;
63     group Mark m;
64     int i;
65
66     group Mark m0 = Mark("X", 0);
67     group Mark m1 = Mark("O", 1);
68
69     if (PLAYER_ON_MOVE % 2 == 0) {
70         m = Mark("X", PLAYER_ON_MOVE);
71     } else {
72         m = Mark("O", PLAYER_ON_MOVE);
73     }
74
75     print("\n"); print(ml); print("\n");
76
77     # players input moves by guessing even or odd
78     print("PLAYER "); print(PLAYER_ON_MOVE); print(": ");
79     print("Even (0) or odd (1)?\n");
80     a = stoi(read(1));
81     c = rand();
82     clear_input();
83     if (c % 2 == a % 2) {
84       print("\n Well guessed!! You are truly skilled!! \n");
85       if (PLAYER_ON_MOVE == 0) {
86           next = (left + (rand() % 6 ));
87           if (next > 39) {next = 39;}
88       } else {
89           next = (right - (rand() % 6 ));
90           if (next < 0) {next = 0;}
91       }
92     if ((next == 39 and PLAYER_ON_MOVE == 0) or (next == 0 and
    PLAYER_ON_MOVE == 1)) {
93
94       pass(winner, PLAYER_ON_MOVE);
95     } else {
```

```
        if (ml.owns(next) == 1-PLAYER_ON_MOVE) {
           if (PLAYER_ON_MOVE == 1) {
              ml << [left];
              left = left -1;
              m0 >> ml >> [left];
           } else {
              ml << [right];
              right = right + 1;
              m1 >> ml >> [right];
           }
        }
           if (PLAYER_ON_MOVE == 0) {
              ml << [left];
              left = next;
              m >> ml >> [next];
           } else {
              ml << [right];
              right = next;
              m >> ml >> [right];
           }

    }
    } else {
      print("\n Too bad, you guessed wrong, better luck next time!!\n");
    }
    pass(prompt, (PLAYER_ON_MOVE + 1) % N_PLAYERS);

}

func void winner() {
    print("\n"); print(ml); print("\n");
    print("Player "); print(PLAYER_ON_MOVE); print(" wins.\n");
    print("Congratulations!\n");
    end;
}

}
```

./examples/checkers.snt

```
@setup
{
group b (Boards.Rect(3,3)) {



 bool checkWin(int player) {

  int other_player=0;
  if(player==0)
  {
    other_player=1;
  }
  int i=1;
  while(i<64)
```

```
  {
    if(this.owns(i))==other_player)
    {
      return false;
    }
    i=i+1;
  }

  return true;
}

              }
# owns checks the owner of a piece at an index,
                   # and returns -1 if the space is empty
                   }
            return False;
        }

  void init_board()
  { Mark mark_red_1 = mark("Red_1");
    Mark mark_red_2 = mark("Red_2");
          Mark mark_red_3 = mark("Red_3");
    Mark mark_red_4 = mark("Red_4");
          Mark mark_red_5 = mark("Red_5");
    Mark mark_red_6 = mark("Red_6");
          Mark mark_red_7 = mark("Red_7");
    Mark mark_red_8 = mark("Red_8");
          Mark mark_red_9 = mark("Red_10");
          Mark mark_red_11 = mark("Red_11");
    Mark mark_red_12 = mark("Red_12");

    Mark mark_black_1 = mark("Black_1");
    Mark mark_black_2 = mark("Black_2");
          Mark mark_black_3 = mark("Black_3");
    Mark mark_black_4 = mark("Black_4");
          Mark mark_black_5 = mark("Black_5");
    Mark mark_black_6 = mark("Black_6");
          Mark mark_black_7 = mark("Black_7");
    Mark mark_black_8 = mark("Black_8");
          Mark mark_black_9 = mark("Black_9");
    Mark mark_black_10 = mark("Black_10");
          Mark mark_black_11 = mark("Black_11");
    Mark mark_black_12 = mark("Black_12");

    this.place(mark_red_1,2);
    this.place(mark_red_2,4);
    this.place(mark_red_3,6);
    this.place(mark_red_4,8);
          this.place(mark_red_5,9);
    this.place(mark_red_6,11);
    this.place(mark_red_7,13);
    this.place(mark_red_8,15);
          this.place(mark_red_9,18);
    this.place(mark_red_10,20);
```

```
    this.place(mark_red_11,22);
    this.place(mark_red_12,24);

          this.place(mark_black_1,41);
    this.place(mark_black_2,43);
    this.place(mark_black_3,45);
    this.place(mark_black_4,47);
          this.place(mark_black_5,50);
    this.place(mark_black_6,52);
    this.place(mark_black_7,54);
    this.place(mark_black_8,56);
          this.place(mark_black_9,57);
    this.place(mark_black_10,59);
    this.place(mark_black_11,61);
    this.place(mark_black_12,63);

}

      bool validatePiece(int c1, int c2,int player)
  {
  bool valid=false;
  int index=toi([c1,c2]);
  {
  if(this.owns(index)==player)
       {valid=true;
                   }
    }
    return valid;
    }
  (
      bool validateCell(int p1,int p2,cur_p1,cur_p2,int player)
      {int index=toi([c1,c2]);
        int otherplayer=0;

        if(player==0)
        {
          otherplayer=1;
        if(p1<cur_p1+1)
          return false;
        else if(p2<cur_p2+1)
          return false;
        else if(this.owns(index)==player || this.owns(index==otherplayer))
           {return false;}
       }
        else
        {
        if(p1>cur_p1-1)
          return false;
        else if(p2>cur_p2-1)
          return false;
        else if(this.owns(index)==player || this.owns(index==otherplayer))
           {return false;}

        }
```

```
        this.remove(index);
        return true;


    }



           bool won (int player) { # checks if the player won
                   if three_in_a_row(player)
                return True;
                   return False;
           }

        bool draw() {
            if this.full() {
                return True;
            }
            return False;
        }

        assert draw() {
            this.full();
        }
       };



group Mark (Piece) {  # inherits from Piece

bool isKing;


        func group Mark __init__(str s) {
  this.s=s;{
        this.fixed = false;
  isKing = false;
        return this;
}

    }
};


          N_PLAYERS = 2; # number of players, this default of 1

          int gamestep=0;


@turns
{
begin () {
gamestep=gamestep+1;
bool validpiece=false;
```

```
bool validcell=false;
int current_player=gamestep % 2;
# this is basically just "while True" with only 1 phase
       # players input moves by typing coordinates, e.g. "11" or "02"
while(validPiece==false)
{
print( Input coordinates of the piece to you want to use for the turn
   )
print( in i.e. \ 22\    or \ 10\   .\n  ); # prompts the players
       int p1 = stoi(read(1)); # reads one character and converts it to an
    int
       int p2 = stoi(read(1));
validpiece=b.validatePiece(p1,p2,current_player);
}

while(validcell==false)
{
print( Input coordinates of the cell you want your piece to move to  )
print( in i.e. \ 22\    or \ 10\   .\n  ); # prompts the players
       int c1 = stoi(read(1)); # reads one character and converts it to an
    int
       int c2 = stoi(read(1));
       validcell=b.validateCell(c1,c2,p1,p2,current_player);
     }


       if (Mark >> b [a,c]) {
              if (b.won(current_player)) {
                  print("Player " + itos(current_player) + " wins.\n");
                  print("Congratulations!");
           }
    }
           # if the move was legal, went through successfully,
           # and the game is not over, pass the turn to the next player
       }
}
```

./examples/checkers_updated.snt

```
@setup
{

list[group line] victory_conds;

group line(Object) {
    list[int] loci;

    func group line __init__(list[int] l) {
        this.loci = l;
        return this;
    }
};

group Mark(Piece) {
    func group Mark __init__(str s, int player) {
```

```
        this.s = s;
        this.owner = player;
        return this;
    }
};

group ttb(Rect(8, 8))
{

    # NOTE: Need to add a way to call INIT_CELLS since this segfaults
    # func group ttb __init__(list[group line] vc) {
    #     group Piece x = Piece();
    #     bool NO = false;
    #     this.x = 3; this.y = 3;
    #     this.victory_conds = vc;
    #     this.cells = [x, x, x, x, x, x, x, x, x];
    #     this.occupied = [NO, NO, NO, NO, NO, NO, NO, NO, NO];
    #     return this;
    # }

    assert owner_of_cell(int index, int player) {
        this.owns(index) == player;

    }

    func bool three_in_a_row(int player) {
        int i = 0;
        int other_player=0;
        bool valid=False;
        if(player == 0)
       {
          other_player=1;
        }

        while (i < 64) {
             print("DEBUG: three_in_a_row, i = "); print(i); print("\n");
            valid= False;
         if (this.owner_of_cell(i, other_player)) {
                return False;
            }
            i = i + 1;
        }
        return True;
        }

    func bool won(int player) {
        bool win=False;
        if (this.three_in_a_row(player)) {
            return True;
        }
        return False;
    }

        func void draw()
```

```
71      {
72          print ("Drawn");
73      }
74
75      func str __str_of_row(int row) {
76          group Piece p1; group Piece p2; group Piece p3; group Piece p4;
    group Piece p5; group Piece p6; group Piece p7; group Piece p8;
77          str ret;
78          # print("DEUBG: start of __str_of_row with row = "); print(row);
    print("\n");
79          # print("DEBUG: print [0, row] = "); print([0, row]); print("\n");
80          p1 = this.cells[this.toi([0, row])];
81          # print("DEBUG: 1 = this.cells call ok\n");
82          p2 = this.cells[this.toi([1, row])];
83          p3 = this.cells[this.toi([2, row])];
84          p4 = this.cells[this.toi([3, row])];
85          p5 = this.cells[this.toi([4, row])];
86          p6 = this.cells[this.toi([5, row])];
87          p7 = this.cells[this.toi([6, row])];
88          p8 = this.cells[this.toi([7, row])];
89          ret = "[" + p1.__repr__() + ", " + p2.__repr__() + ", " + p3.
    __repr__() + ", " + p4.__repr__() + ", " + p5.__repr__() + ", " + p6.
    __repr__() + ", " + p7.__repr__() + ", " + p8.__repr__() + "]\n";
90          return ret;
91      }
92
93      func str __repr__() {
94          return this.__str_of_row(0) + this.__str_of_row(1) + this.
    __str_of_row(2)+this.__str_of_row(3)+this.__str_of_row(4)+this.
    __str_of_row(5)+this.__str_of_row(6)+this.__str_of_row(7);
95      }
96
97          func bool validatePiece(int index,int player)
98      {
99      bool valid=False;
100   print("index");
101     {
102     if(this.owns(index)==player)
103        {valid=True;
104                    }
105     }
106     return valid;
107     }
108
109     func bool validateCell(int p1, int p2, int cur_p1, int cur_p2, int
    player)
110        {int index=this.toi([p1,p2]);
111         int otherplayer=0;
112         int temp1;
113         int temp2;
114         int tempindex;
115         if(player==0)
116         {
117           otherplayer=1;
```

```
        if(p1-cur_p1 != p2-cur_p2)
        {
            return False;
        }
        if(p1<cur_p1+1)

         {
          return False;
         }

        if(p2<cur_p2+1)

          {return False;
          }
        if(this.owns(index)==player or this.owns(index)==otherplayer)
           {return False;}
        if(p1-cur_p1>1)
        {temp1=cur_p1+1;
         temp2=cur_p2+1;
          while(temp1<p1)
        {   tempindex=this.toi([temp1,temp2]);
            if(this.owns(tempindex)==otherplayer)
            {   temp1=temp1+1;
                temp2=temp2+1;
                continue;
            }
            else
            {
                return False;
            }

        }
        }

       }
        else
        {
         if(cur_p1-p1 != cur_p2-p2)
        {
            return False;
        }
        if(p1>cur_p1-1)
          {return False;}
        if(p2>cur_p2-1)
          {return False;}
        if(this.owns(index) == player or this.owns(index)==otherplayer)
           {return False;}
        if(cur_p1-p1>1)
       {temp1=cur_p1-1;
        temp2=cur_p2-1;
        while(temp1>p1)
        {   tempindex=this.toi([temp1,temp2]);
            if(this.owns(tempindex)==otherplayer)
            {   temp1=temp1-1;
```

```
                temp2=temp2-1;
                continue;
            }
            else
            {
                return False;
            }

        }


       }

        }
        this.remove(index);
        return True;


    }

func void removePieces(int p1,int p2, int cur_p1,int cur_p2)
{int temp1;
    int temp2;
    int tempindex;
if(cur_p1-p1>1)

    {temp1=cur_p1-1;
        temp2=cur_p2-1;
        while(temp1>p1)
        {   tempindex=this.toi([temp1,temp2]);
            this.remove(tempindex);

        }


       }
       if(p1-cur_p1>1)
        {temp1=cur_p1+1;
         temp2=cur_p2+1;
          while(temp1<p1)
        {   tempindex=this.toi([temp1,temp2]);

            this.remove(tempindex);
        }
        }
}

    func void init_board()
    {int player=0;
        int otherplayer=1;

    group Mark mark_red_1 = Mark("R",player);
      group Mark mark_red_2 = Mark("R",player);
          group Mark mark_red_3 = Mark("R",player);
```

```
      group Mark mark_red_4 = Mark("R",player);
         group Mark mark_red_5 = Mark("R",player);
      group Mark mark_red_6 = Mark("R",player);
    group Mark mark_red_7 = Mark("R",player);
      group Mark mark_red_8 = Mark("R",player);
         group Mark mark_red_9 = Mark("R",player);
         group Mark mark_red_10 = Mark("R",player);

          group Mark mark_red_11 = Mark("R",player);
     group Mark mark_red_12 = Mark("R",player);

   group Mark mark_black_1 = Mark("B",otherplayer);
     group Mark mark_black_2 = Mark("B",otherplayer);
    group Mark mark_black_3 = Mark("B",otherplayer);
      group Mark mark_black_4 = Mark("B",otherplayer);
         group Mark mark_black_5 = Mark("B",otherplayer);
     group Mark mark_black_6 = Mark("B",otherplayer);
          group Mark mark_black_7 = Mark("B",otherplayer);
      group Mark mark_black_8 = Mark("B",otherplayer);
         group Mark mark_black_9 = Mark("B",otherplayer);
      group Mark mark_black_10 = Mark("B",otherplayer);
         group Mark mark_black_11 = Mark("B",otherplayer);
     group Mark mark_black_12 = Mark("B",otherplayer);

      this.place(mark_red_1,2);
      this.place(mark_red_2,4);
      this.place(mark_red_3,6);
      this.place(mark_red_4,8);
          this.place(mark_red_5,9);
      this.place(mark_red_6,11);
      this.place(mark_red_7,13);
      this.place(mark_red_8,15);
          this.place(mark_red_9,18);
      this.place(mark_red_10,20);
      this.place(mark_red_11,22);
      this.place(mark_red_12,24);

          this.place(mark_black_1,41);
      this.place(mark_black_2,43);
      this.place(mark_black_3,45);
      this.place(mark_black_4,47);
          this.place(mark_black_5,50);
      this.place(mark_black_6,52);
      this.place(mark_black_7,54);
      this.place(mark_black_8,56);
          this.place(mark_black_9,57);
      this.place(mark_black_10,59);
      this.place(mark_black_11,61);
      this.place(mark_black_12,63);

}


            };
```

```
int N_PLAYERS = 2;

group ttb b;


}


@turns
{

func void begin() {

    # list[group line] victory_conds;

    b = ttb();
    b.init_board();

    # print("DEBUG: v7 ok\n");


    # print("DEBUG: built victory_conds\n");

    # b = ttb();

    # print("DEBUG: built b\n");

    pass(prompt, 0);
}

func void prompt() {
    int a;
    int c;
    int x;
    int y;
    int index;
    bool valid_piece=False;
    bool valid_cell=False;
    group Mark m;
    int i;

    if (PLAYER_ON_MOVE % 2 == 0) {
        m = Mark("R", PLAYER_ON_MOVE);
    } else {
        m = Mark("B", PLAYER_ON_MOVE);
    }

```

```
    # print("DEBUG: start of prompt()\n");

    print("\n"); print(b); print("\n");

    # print("DEBUG: printed b\n");

    # players input moves by typing coordinates, e.g. "11" or "02"

    print("PLAYER "); print(PLAYER_ON_MOVE); print(": ");
    while(not valid_piece)
    {
    print("Input coordinates of the piece to be moved ");
    print("in i.e. \"22\" or \"10\".\n");
    a = stoi(read(1));
    c = stoi(read(1));
    index=b.toi([a,c]);
    valid_piece=b.validatePiece(index,PLAYER_ON_MOVE);
    }

    while(not valid_cell)
    {
    print("Input coordinates of the piece to be moved ");
    print("in i.e. \"22\" or \"10\".\n");
    x= stoi(read(1));
    y = stoi(read(1));
    index=b.toi([x,y]);
    valid_cell=b.validateCell(x,y,a,c,PLAYER_ON_MOVE);
    }

    clear_input();
    # print("index of input: ["); print(a); print(", "); print(c); print("
   ] -> ");
    # print(b.toi([a, c])); print("\n");

    if (m >> b >> [x, y]) {
        b.removePieces(x,y,a,c);
        # PLAYER_ON_MOVE is the index of the player
        if (b.won(PLAYER_ON_MOVE)) {
            pass(winner, PLAYER_ON_MOVE);
        }
    }
    else {
        # A piece is already at [a, c]
        print("Cannot place a mark there, try again.\n\n");
        pass(prompt, PLAYER_ON_MOVE);
    }
    # if the move was legal, went through successfully,
    # and the game is not over, pass the turn to the next player
    pass(prompt, (PLAYER_ON_MOVE + 1) % N_PLAYERS);
}

func void winner() {

    print("\n"); print(b); print("\n");
```

```
    print("Player "); print(PLAYER_ON_MOVE); print(" wins.\n");
    print("Congratulations!\n");
    end;
}

func void nowinner() {
    print("\n"); print(b); print("\n");
    print("Game ends in a draw.\n");
    end;
}

}
```

./examples/tictactoe.snt

```
@setup
{

list[group line] victory_conds;

group line(Object) {
    list[int] loci;

    func group line __init__(list[int] l) {
        this.loci = l;
        return this;
    }
};

group ttb(Rect(3, 3)) {

    # NOTE: Need to add a way to call INIT_CELLS since this segfaults
    # func group ttb __init__(list[group line] vc) {
    #     group Piece x = Piece();
    #     bool NO = false;
    #     this.x = 3; this.y = 3;
    #     this.victory_conds = vc;
    #     this.cells = [x, x, x, x, x, x, x, x, x];
    #     this.occupied = [NO, NO, NO, NO, NO, NO, NO, NO, NO];
    #     return this;
    # }

    assert owner_of_line(group line l, int player) {
        this.owns(l.loci[0]) == player;
        this.owns(l.loci[1]) == player;
        this.owns(l.loci[2]) == player;
    }

    func bool three_in_a_row(int player) {
        int i = 0;
        group line l;
        while (i < 8) {
            l = victory_conds[i];
            # print("DEBUG: three_in_a_row, i = "); print(i); print("\n");

```

```
            if (this.owner_of_line(l, player)) {
                return True;
            }
            i = i + 1;
        }
        return False;
        }

    func bool won(int player) {
        if (this.three_in_a_row(player)) {
            return True;
        }
        return False;
    }

    assert draw() {
        this.full();
    }

    func str __str_of_row(int row) {
        group Piece l; group Piece m; group Piece r;
        str ret;
        # print("DEUBG: start of __str_of_row with row = "); print(row);
   print("\n");
        # print("DEBUG: print [0, row] = "); print([0, row]); print("\n");
        l = this.cells[this.toi([0, row])];
        # print("DEBUG: l = this.cells call ok\n");
        m = this.cells[this.toi([1, row])];
        r = this.cells[this.toi([2, row])];
        ret = "[" + l.__repr__() + ", " + m.__repr__() + ", " + r.__repr__
   () + "]\n";
        return ret;
    }

    func str __repr__() {
        return this.__str_of_row(0) + this.__str_of_row(1) + this.
   __str_of_row(2);
    }
};



group Mark(Piece) {
    func group Mark __init__(str s, int player) {
        this.s = s;
        this.owner = player;
        return this;
    }
};


int N_PLAYERS = 2;

group ttb b;
```

```
}


@turns
{

func void begin() {
    group line v0; group line v1; group line v2; group line v3;
    group line v4; group line v5; group line v6; group line v7;
    int x; int y; int z;
    # list[group line] victory_conds;

    b = ttb();

    x = b.toi([0, 0]); y = b.toi([1, 0]); z = b.toi([2, 0]);
    v0 = line([x, y, z]);
    x = b.toi([0, 1]); y = b.toi([1, 1]); z = b.toi([2, 1]);
    v1 = line([x, y, z]);
    x = b.toi([0, 2]); y = b.toi([1, 2]); z = b.toi([2, 2]);
    v2 = line([x, y, z]);
    x = b.toi([0, 0]); y = b.toi([0, 1]); z = b.toi([0, 2]);
    v3 = line([x, y, z]);
    x = b.toi([1, 0]); y = b.toi([1, 1]); z = b.toi([1, 2]);
    v4 = line([x, y, z]);
    x = b.toi([2, 0]); y = b.toi([2, 1]); z = b.toi([2, 2]);
    v5 = line([x, y, z]);
    x = b.toi([0, 0]); y = b.toi([1, 1]); z = b.toi([2, 2]);
    v6 = line([x, y, z]);
    x = b.toi([0, 2]); y = b.toi([1, 1]); z = b.toi([2, 0]);
    v7 = line([x, y, z]);

    # print("DEBUG: v7 ok\n");

    victory_conds = [v0, v1, v2, v3, v4, v5, v6, v7];

    # print("DEBUG: built victory_conds\n");

    # b = ttb();

    # print("DEBUG: built b\n");

    pass(prompt, 0);
}

func void prompt() {
    int a;
    int c;
    group Mark m;
    int i;

    if (PLAYER_ON_MOVE % 2 == 0) {
        m = Mark("X", PLAYER_ON_MOVE);
    } else {
```

```
146         m = Mark("O", PLAYER_ON_MOVE);
147     }
148 
149     # print("DEBUG: start of prompt()\n");
150 
151     print("\n"); print(b); print("\n");
152 
153     # print("DEBUG: printed b\n");
154 
155     # players input moves by typing coordinates, e.g. "11" or "02"
156     print("PLAYER "); print(PLAYER_ON_MOVE); print(": ");
157     print("Input coordinates of square to place ");
158     print("in i.e. \"22\" or \"10\".\n");
159     a = stoi(read(1));
160     c = stoi(read(1));
161     clear_input();
162     # print("index of input: ["); print(a); print(", "); print(c); print("
    ] -> ");
163     # print(b.toi([a, c])); print("\n");
164 
165     if (m >> b >> [a, c]) {
166 
167         # PLAYER_ON_MOVE is the index of the player
168         if (b.won(PLAYER_ON_MOVE)) {
169             pass(winner, PLAYER_ON_MOVE);
170         }
171         if (b.draw()) {
172             pass(nowinner, PLAYER_ON_MOVE);
173         }
174     } else {
175         # A piece is already at [a, c]
176         print("Cannot place a mark there, try again.\n\n");
177         pass(prompt, PLAYER_ON_MOVE);
178     }
179     # if the move was legal, went through successfully,
180     # and the game is not over, pass the turn to the next player
181     pass(prompt, (PLAYER_ON_MOVE + 1) % N_PLAYERS);
182 }
183 
184 func void winner() {
185     print("\n"); print(b); print("\n");
186     print("Player "); print(PLAYER_ON_MOVE); print(" wins.\n");
187     print("Congratulations!\n");
188     end;
189 }
190 
191 func void nowinner() {
192     print("\n"); print(b); print("\n");
193     print("Game ends in a draw.\n");
194     end;
195 }
196 
197 }
```

./Makefile

```
OBJS = ast.cmo parser.cmo scanner.cmo types.cmo stdlib.cmo sast.cmo cast.cmo \
       compile.cmo senet.cmo
VERBOSE_YACC = -v

# Choose one
# YACC = ocamlyacc
YACC = menhir --explain

senet : $(OBJS)
  ocamlc -o senet $(OBJS)

scanner.ml : scanner.mll
  ocamllex scanner.mll

parser.ml parser.mli : parser.mly
  $(YACC) $(VERBOSE_YACC) parser.mly

%.cmo : %.ml
  ocamlc -c $<

%.cmi : %.mli
  ocamlc -c $<

.PHONY : clean
clean :
  @rm -f senet parser.ml parser.mli scanner.ml \
  *.cmo *.cmi *.out *.diff *.output *.conflicts *.automaton \
  output.c testall.log

# Generated by ocamldep *.ml *.mli *.mly *.mll
# see http://caml.inria.fr/pub/docs/manual-ocaml/depend.html
ast.cmo :
ast.cmx :
cast.cmo : types.cmo sast.cmo ast.cmo
cast.cmx : types.cmx sast.cmx ast.cmx
compile.cmo : types.cmo sast.cmo ast.cmo
compile.cmx : types.cmx sast.cmx ast.cmx
parser.cmo : ast.cmo parser.cmi
parser.cmx : ast.cmx parser.cmi
sast.cmo : types.cmo stdlib.cmo ast.cmo
sast.cmx : types.cmx stdlib.cmx ast.cmx
scanner.cmo : parser.cmi
scanner.cmx : parser.cmx
senet.cmo : scanner.cmo sast.cmo parser.cmi compile.cmo cast.cmo ast.cmo
senet.cmx : scanner.cmx sast.cmx parser.cmx compile.cmx cast.cmx ast.cmx
stdlib.cmo : types.cmo
stdlib.cmx : types.cmx
types.cmo :
types.cmx :
parser.cmi : ast.cmo
```

./parser.mly

```
%{ open Ast
   let fst_of_three (x, _, _) = x
   let snd_of_three (_, x, _) = x
   let trd_of_three (_, _, x) = x  %}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA DOT
%token LBRACKET RBRACKET
%token PLUS MINUS TIMES DIVIDE ASSIGN MOD
%token AND OR NOT
%token EQ NEQ LT LEQ GT GEQ
%token RETURN IF ELSE FOR WHILE INT
%token <int> INTLITERAL
%token <string> STRLITERAL
%token <string> ID
%token EOF
%token IN
%token BREAK CONTINUE PASS END
%token ELIF
%token TRUE FALSE NONE
%token STR BOOL VOID LIST GROUP
%token ASSERT
%token REMOVE PLACE
%token SETUP TURNS FUNC
%token THIS

/* lowest precedance */

/* %nonassoc NOELSE */
/* %nonassoc ELSE */
%right ASSIGN
%left OR
%left AND
%nonassoc NOT
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%nonassoc UMINUS
%nonassoc LBRACKET
/* %left DOT */

/* highest precedance */

%start program
%type <Ast.program> program

%%

program:
  twoparts EOF { $1 }

twoparts:
    SETUP LBRACE decls RBRACE TURNS LBRACE fdecl_list RBRACE
      {(List.rev (fst_of_three $3),
```

```
        List.rev (snd_of_three $3),
        List.rev (trd_of_three $3)),
        List.rev $7}

decls:
   /* nothing */ { [], [], [] }
 | decls vdecl { ($2 :: fst_of_three $1),
                 snd_of_three $1,
                 trd_of_three $1 }
 | decls fdecl { fst_of_three $1,
                 ($2 :: snd_of_three $1),
                 trd_of_three $1 }
 | decls gdecl { fst_of_three $1,
                 snd_of_three $1,
                 ($2 :: trd_of_three $1)}

gdecl:
    GROUP ID LPAREN extend_opt RPAREN LBRACE vdecl_list fdecl_list RBRACE
   SEMI
      { { gname = $2;
          extends = $4;
          par_actuals = None;
          attributes = List.rev $7;
          methods = List.rev $8 } }
  | GROUP ID LPAREN field_expr LPAREN actuals_opt RPAREN RPAREN LBRACE
   vdecl_list fdecl_list RBRACE SEMI
      { { gname = $2;
          extends = Some($4);
          par_actuals = Some($6);
          attributes = List.rev $10;
          methods = List.rev $11 } }

extend_opt:
    /* nothing */ { None }
  | field_expr    { Some($1) }

fdecl_list:
    /* nothing */ { [] }
  | fdecl_list fdecl  { $2 :: $1 }

fdecl:
   FUNC type_id ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list
   RBRACE
     { BasicFunc({ ftype = $2;
     fname = $3;
     formals = $5;
     locals = List.rev $8;
     body = List.rev $9 }) }
  | ASSERT ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
      { AssertFunc({ fname = $2;
     formals = $4;
     locals = List.rev $7;
     body = List.rev $8 }) }

```

```
formals_opt:
    /* nothing */ { [] }
  | formal_list   { List.rev $1 }

formal_list:
    type_id ID                   { [{ vtype = $1; vname = $2; vinit =
   NoInit }] }
  | formal_list COMMA type_id ID { { vtype = $3; vname = $4; vinit =
   NoInit } :: $1 }

vdecl_list:
    /* nothing */    { [] }
  | vdecl_list vdecl { $2 :: $1 }

vdecl:
    type_id ID SEMI
      { { vtype = $1;
          vname = $2;
          vinit = NoInit } }
  | type_id ID ASSIGN expr SEMI
      { { vtype = $1;
          vname = $2;
          vinit = ExprInit($4) } }

type_id:
    INT                           { Int }
  | BOOL                          { Bool }
  | STR                           { Str }
  | VOID                          { Void }
  | LIST LBRACKET type_id RBRACKET { List($3) }
  | GROUP ID                      { Group($2) }

stmt_list:
    /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt_list_req:
    stmt               { [$1] }
  | stmt_list_req stmt { $2 :: $1 }

stmt:
    expr SEMI { Expr($1) }
  | RETURN expr SEMI { Return($2) }
  | LBRACE stmt_list RBRACE { Block(List.rev $2) }
  | IF LPAREN expr RPAREN LBRACE stmt_list_req RBRACE /* %prec NOELSE */
     { If($3, Block(List.rev $6), None, Block([])) }
  | IF LPAREN expr RPAREN LBRACE stmt_list_req RBRACE ELIF expr LBRACE
   stmt_list_req RBRACE
     { If($3, Block(List.rev $6), Some($9), Block(List.rev $11)) }
  | IF LPAREN expr RPAREN LBRACE stmt_list_req RBRACE ELSE LBRACE
   stmt_list_req RBRACE
     { If($3, Block(List.rev $6), None, Block(List.rev $10)) }
  | FOR LPAREN type_id ID IN LBRACE expr_list RBRACE RPAREN LBRACE
   stmt_list_req RBRACE
```

```
155      { For({ vtype = $3;
156              vname = $4;
157              vinit = NoInit },
158            List.rev $7,
159            Block(List.rev $11)) }
160    | WHILE LPAREN expr RPAREN LBRACE stmt_list_req RBRACE
161      { While($3, Block(List.rev $6)) }
162    | BREAK SEMI    { Break}
163    | CONTINUE SEMI { Continue }
164    | END SEMI { End }
165    | PASS LPAREN ID COMMA expr RPAREN SEMI { Pass($3, $5) }
166
167
168  expr:
169      INTLITERAL       { IntLiteral($1) }
170    | STRLITERAL       { StrLiteral($1) }
171    | NONE             { VoidLiteral }
172    | bool_lit         { BoolLiteral($1) }
173    | list_lit         { ListLiteral($1) }
174    | field_expr       { Field($1) }
175    | expr PLUS   expr { Binop($1, Add,   $3) }
176    | expr MINUS  expr { Binop($1, Sub,   $3) }
177    | expr TIMES  expr { Binop($1, Mult,  $3) }
178    | expr DIVIDE expr { Binop($1, Div,   $3) }
179    | expr MOD    expr { Binop($1, Mod,   $3) }
180    | expr EQ     expr { Binop($1, Equal, $3) }
181    | expr NEQ    expr { Binop($1, Neq,   $3) }
182    | expr LT     expr { Binop($1, Less,  $3) }
183    | expr LEQ    expr { Binop($1, Leq,   $3) }
184    | expr GT     expr { Binop($1, Greater,  $3) }
185    | expr GEQ    expr { Binop($1, Geq,   $3) }
186    | expr AND    expr { Binop($1, And,   $3) }
187    | expr OR    expr  { Binop($1, Or,    $3) }
188    | field_expr ASSIGN expr   { Assign($1, $3) }
189    | field_expr LPAREN actuals_opt RPAREN { Call($1, $3) }
190    | expr LBRACKET expr RBRACKET { Element($1, $3) }
191    | LPAREN expr_opt RPAREN { $2 }
192    | MINUS expr %prec UMINUS { Uminus($2) }
193    | NOT expr { Not($2) }
194    | field_expr PLACE field_expr PLACE list_lit   { Place($1, $3, $5) }
195    | field_expr REMOVE list_lit { Remove($1, $3) }
196
197  expr_opt:
198      /* nothing */ { Noexpr }
199    | expr          { $1 }
200
201
202  field_expr:
203      ID                    { Id($1) }
204    | THIS                  { This }
205    | field_expr DOT ID     { FieldCall($1, $3) }
206
207  expr_list:
208      expr                { [$1] }
```

```
  | expr_list COMMA expr { $3 :: $1 }

actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                    { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }

list_lit:
    LBRACKET RBRACKET            { EmptyList }
  | LBRACKET expr_list RBRACKET  { Elems(List.rev $2) }

bool_lit:
    TRUE  { True }
  | FALSE { False }
```

./sast.ml

```
open Types

exception SemError of string

type counter = {
  mutable i : int
}

let count = { i = 0 }

let rec string_of_t = function
    Int -> "int"
  | Bool -> "bool"
  | Str -> "str"
  | Void -> "void"
  | List_t(vt) ->
      "list[" ^ string_of_t vt ^ "]"
  | Group(s, _) -> "group " ^ s

let rec id_type_to_t = function
    Ast.Int -> Int
  | Ast.Bool -> Bool
  | Ast.Str -> Str
  | Ast.Void -> Void
  | Ast.List(id_typ) -> List_t(id_type_to_t id_typ)
  | Ast.Group(s) -> Group(s, None)

let rec find_variable (scope : symbol_table) name =
  try
    List.find (fun v -> v.vname = name) scope.variables
  with Not_found ->
    match scope.parent with
      Some(parent) -> find_variable parent name
    | _ -> raise Not_found

```

```
let rec find_function (scope : symbol_table) name =
  try
    List.find (fun f -> match f with
                    BasicFunc(x) -> x.fname = name
                  | AssertFunc(x) -> x.aname = name) scope.functions
  with Not_found ->
    match scope.parent with
      Some(parent) -> find_function parent name
    | _ -> raise Not_found

let rec find_group (scope : symbol_table) name =
  try
    List.find (fun g -> g.gname = name) scope.groups
  with Not_found ->
    match scope.parent with
      Some(parent) -> find_group parent name
    | _ -> raise Not_found

let verify_args_signature fdcl formals actuals =
  let rec helper check_types flist alist = match flist, alist with
      [], [] -> true
    | f :: frest, a :: arest ->
      let tf = f.vtype
      and _, ta = a in
      if tf = ta || not check_types then
        helper check_types frest arest
      else
        (match tf, ta with
           Group(form_name, _), Group(act_name, _) ->
            (* if form_name = act_name then *)
              helper check_types frest arest
            (* else
              false *)
         | _, _ -> false)
    | _ :: _, [] -> false
    | [], _ :: _ -> false
  in
  let fname, is_built_in =
    (match fdcl with
        BasicFunc(f) -> f.fname, f.f_is_built_in
      | AssertFunc(f) -> f.aname, f.a_is_built_in)
  in
  let check_formal_types =
    if fname = "print" && is_built_in then
      let formal_type =
        (match fdcl with
            BasicFunc(f) -> List.hd f.formals
          | _ -> raise (SemError ("Internal error: built-in print function
    cannot be an assert function")))
      in
      (match formal_type.vtype with
          Group("", _) -> false
        | _ -> true)
    else
```

```
      true
  in
  helper check_formal_types formals actuals

let rec search_func_in_child (parent : group_decl) actuals name =
  let rec helper = function
      [] -> raise Not_found
    | f :: rest ->
        let fname, formals =
          (match f with
              BasicFunc(x) -> x.fname, x.formals
            | AssertFunc(x) -> x.aname, x.aformals)
        in
        if name = fname &&
           List.length formals = List.length actuals then
          if verify_args_signature f formals actuals then
            f
          else
            helper rest
        else
          helper rest
  in
  helper parent.methods

let search_func_in_scope scope actuals name =
  let rec helper = function
      [] -> raise (SemError ("Function name " ^ name ^ " exists in group
   scope " ^
                             "but actuals signature not matched"))
    | f :: rest ->
        let fname, formals, built_in =
          (match f with
              BasicFunc(x) -> x.fname, x.formals, x.f_is_built_in
            | AssertFunc(x) -> x.aname, x.aformals, x.a_is_built_in)
        in
        if name = "print" && name = fname && built_in then
          f
        else if name = fname &&
          List.length formals = List.length actuals then
          if verify_args_signature f formals actuals then
            f
          else
            helper rest
        else
          helper rest
  in
  helper scope.functions

let rec find_child_in_group_def env (parent : group_decl) actuals name =
  if List.exists (fun v -> v.vname = name) parent.attributes then
    let vdcl = List.find (fun v -> v.vname = name) parent.attributes in
    Var(vdcl), vdcl.vtype
  else
  if List.exists (fun f -> match f with
```

```
                      BasicFunc(x) -> x.fname = name
                    | AssertFunc(x) -> x.aname = name) parent.methods then
    try
      let fdcl = search_func_in_child parent actuals name in
      let f_typ = (match fdcl with
                      BasicFunc(x) -> x.ftype
                    | AssertFunc(x) -> Bool) in
      Fun(fdcl), f_typ
    with Not_found ->
      (match parent.extends with
          None -> raise Not_found
        | Some(g) -> find_child_in_group_def env g actuals name)
  else
    (match parent.extends with
        None -> raise Not_found
      | Some(g) -> find_child_in_group_def env g actuals name)

let rec find_child env (par_instance : var_decl) actuals name =
  let class_name =
    (match par_instance.vtype with
       Group(s, _) -> s
     | _ -> raise (SemError ("DOT operator does not work with non-group
   variable: " ^ par_instance.vname)))
  in
  let parent =
    try
      find_group env.scope class_name
    with Not_found ->
      raise (SemError ("Group definition not found: " ^ class_name))
  in
  let child, child_typ =
    try
      find_child_in_group_def env parent actuals name
    with Not_found ->
        raise (SemError ("Child of " ^ parent.gname ^ " not found: " ^
   name ^
                         ";\nActual types: " ^ String.concat ", " (List.
   map (fun (_, typ) -> string_of_t typ) actuals)))
  in
  (match child with
      Var(v) -> Attrib(par_instance, v), child_typ
    | Fun(f) -> Method(par_instance, f), child_typ
    | _ -> raise (SemError ("Child is not a variable or function")))

let find_this_child env actuals name =
  let info =
    (match env.partial_group_info with
       None -> (raise (SemError "'this' field call outside of group
   definition"))
     | Some(info) -> info) in
  let this_dummy =
    { vname = "this"; vtype = Group(info.group_name, None);
      vinit = None; vloop = false } in
  let scope = info.symbols in
```

```
  if List.exists (fun v -> v.vname = name) scope.variables then
    let vdcl = List.find (fun v -> v.vname = name) scope.variables in
    Attrib(this_dummy, vdcl), vdcl.vtype
  else
  if List.exists (fun f -> match f with
                     BasicFunc(x) -> x.fname = name
                   | AssertFunc(x) -> x.aname = name) scope.functions then
    let fdcl = search_func_in_scope scope actuals name in
    let f_typ = (match fdcl with
                    BasicFunc(x) -> x.ftype
                  | AssertFunc(x) -> Bool) in
      Method(this_dummy, fdcl), f_typ
  else
    (match info.par with
        None -> raise Not_found
      | Some(p) ->
          let child, child_typ =
            try
              find_child_in_group_def env p actuals name
            with Not_found ->
              raise (SemError ("Child of 'this' not found: " ^ name))
          in
          (match child with
              Var(v) -> Attrib(this_dummy, v), child_typ
            | Fun(f) -> Method(this_dummy, f), child_typ
            | _ -> raise (SemError ("Child is not a variable or function")
   )))

let rec search_func_in_parent scope actuals name =
  let rec helper = function
      [] ->
        (match scope.parent with
            Some(parent) -> search_func_in_parent parent actuals name
          | None -> raise Not_found)
    | f :: rest ->
        let formals = (match f with
                            BasicFunc(x) -> x.formals
                          | AssertFunc(x) -> x.aformals)
        in
        if List.length formals = List.length actuals then
          if verify_args_signature f formals actuals then
            f
          else
            helper rest
        else
          helper rest
  in
  helper scope.functions

let rec search_field_local_first scope actuals name =
  let fe_is_v =
    List.exists (fun x -> x.vname = name) scope.variables
  and fe_is_f =
    List.exists (fun x -> match x with
```

```
                    BasicFunc(b) -> b.fname = name
                  | AssertFunc(a) -> a.aname = name )
                scope.functions
  and fe_is_g =
    List.exists (fun x -> x.gname = name) scope.groups
  in
  if fe_is_v then
    let vdecl =
      List.find (fun v -> v.vname = name) scope.variables
    in
    Var(vdecl)
  else if fe_is_f then
    let rec helper = function
        [] ->
            (match scope.parent with
                Some(parent) -> search_func_in_parent parent actuals name
              | None -> raise (SemError ("Function name " ^ name ^ "
   exists in scope " ^
                                "but actuals signature not matched")))
      | f :: rest ->
        let n, formals, built_in =
          (match f with
             BasicFunc(x) -> x.fname, x.formals, x.f_is_built_in
           | AssertFunc(x) -> x.aname, x.aformals, x.a_is_built_in)
        in
        if n = name && n = "print" && built_in then
          f
        else if n = name &&
         List.length formals = List.length actuals &&
         verify_args_signature f formals actuals then
          f
        else
          helper rest
    in
    let fdecl = helper scope.functions in
    Fun(fdecl)
  else if fe_is_g then
    let gdecl =
      List.find (fun g -> g.gname = name) scope.groups
    in
    Grp(gdecl)
  else
    match scope.parent with
        Some(parent) -> search_field_local_first parent actuals name
      | _ -> raise Not_found

let rec check_field env actuals = function
    Ast.Id(name) ->
      let dcl =
        try
          search_field_local_first env.scope actuals name
        with Not_found ->
          raise (SemError("Undeclared identifier: " ^ name))
      in
```

```
298       let typ = match dcl with
299           Var(v) -> v.vtype
300         | Fun(x) -> (match x with
301             BasicFunc(f) -> f.ftype
302           | AssertFunc(a) -> Bool)
303         | Grp(g) -> Group(g.gname, None)
304         | This -> raise (SemError("Internal error: 'this' keyword match
    with Ast.Id"))
305         | _ -> raise (SemError "Internal error: Ast.Id matched with Attrib
     or Method")
306       in
307       dcl, typ
308   | Ast.This ->
309       let gname =
310         (match env.partial_group_info with
311             None -> raise (SemError("'this' keyword used outside of group
    declaration"))
312           | Some(info) -> info.group_name)
313       in
314       This, Group(gname, None)
315   | Ast.FieldCall(fe, name) ->
316     let parent, _ = check_field env [] fe in
317       (match parent with
318           Var(par) ->
319             (try
320               find_child env par actuals name
321             with Not_found ->
322               raise (SemError("Undeclared child identifier: " ^ name)))
323         | This ->
324             (try
325               find_this_child env actuals name
326             with Not_found ->
327               raise (SemError("Undeclared 'this' child identifier: " ^
    name)))
328         | _ ->
329             raise (SemError("Parent is either a function or variable, not
    a group")))
330
331 let create_ll_name scope =
332   count.i <- count.i + 1;
333   "__ll__" ^ string_of_int count.i
334
335 let create_elem_name scope =
336   count.i <- count.i + 1;
337   "__elem__" ^ string_of_int count.i
338
339 let ast_op_to_sast_op = function
340     Ast.Add -> Add
341   | Ast.Sub -> Sub
342   | Ast.Mult -> Mult
343   | Ast.Div -> Div
344   | Ast.Equal -> Equal
345   | Ast.Neq -> Neq
346   | Ast.Less -> Less
```

```
  | Ast.Leq -> Leq
  | Ast.Greater -> Greater
  | Ast.Geq -> Geq
  | Ast.Mod -> Mod
  | Ast.And -> And
  | Ast.Or -> Or

let require_bool e msg = match e with
    _, Bool -> ()
  | _, _ -> raise (SemError msg)

let require_int e msg = match e with
    _, Int -> ()
  | _, _ -> raise (SemError msg)

let require_same e1 e2 msg =
  let _, t1 = e1
  and _, t2 = e2 in
  if t1 = t2 then
    ()
  else raise (SemError msg)

let require_integer_list l msg = match l with
    _, List_t(typ) ->
      (match typ with
         Int -> ()
       | _ -> raise (SemError msg))
  | _, _ -> raise (SemError msg)

let rec require_parent_helper pname gdcl msg =
  if gdcl.gname = pname then
    ()
  else
    (match gdcl.extends with
       Some(gp) -> require_parent_helper pname gp msg
     | _ -> raise (SemError msg))

let rec require_parent env pname fe msg = match fe with
    Var(v) ->
      (match v.vtype with
          Group(s, _) ->
            let gdcl =
              try
                find_group env.scope s
              with Not_found ->
                raise (SemError ("require_parent did not find group: " ^
   pname))
            in
            require_parent_helper pname gdcl msg
        | _ -> raise (SemError ("Variable is not a group: " ^ v.vname)))
  | _ -> raise (SemError ("Field expr. is not a variable"))

let rec verify_args_helper f_typ a_typ check_types = match f_typ, a_typ
   with
```

```
399     [], [] -> ()
400   | f_hd :: f_rest, a_hd :: a_rest ->
401       if f_hd = a_hd then
402         verify_args_helper f_rest a_rest check_types
403       else if not check_types then
404         ()
405       else
406         (match f_hd, a_hd with
407             Group(_, _), Group(_, _) -> ()
408           | _, _ ->
409             raise (SemError ("Formal type " ^ string_of_t f_hd ^ " " ^
410                              "does not match actual type " ^ string_of_t
    a_hd)))
411   | _, [] ->
412       raise (SemError ("Formal and actual argument lengths do not match"))
413   | [], _ ->
414       raise (SemError ("Formal and actual argument lengths do not match"))
415
416 let verify_args fdcl formals actuals =
417   let f_typ = List.map (fun v -> v.vtype) formals in
418   let a_typ = List.map (fun (_, typ) -> typ) actuals in
419   let fname, is_built_in =
420     (match fdcl with
421         BasicFunc(f) -> f.fname, f.f_is_built_in
422       | AssertFunc(f) -> f.aname, f.a_is_built_in)
423   in
424   if fname = "print" && is_built_in then
425     match fdcl with
426         BasicFunc(f) -> ()
427       | _ -> raise (SemError ("Internal error: built-in print function
    cannot be an assert function"))
428   else
429     verify_args_helper f_typ a_typ true
430
431 let rec verify_elems_list_type env typ = function
432     [] -> typ
433   | (a, b) :: rest ->
434       if b = typ then
435         verify_elems_list_type env typ rest
436       else
437         raise (SemError ("List elements are not all of same type: " ^
438                          string_of_t typ))
439
440 let rec check_single_elem env (detail, typ) = match detail with
441     IntLiteral(i, _) ->
442       let name = create_elem_name env.scope in
443       IntLiteral(i, name), typ
444   | StrLiteral(s, _) ->
445       let name = create_elem_name env.scope in
446       StrLiteral(s, name), typ
447   | ListLiteral(ll) ->
448       raise (SemError ("A list literal cannot be a list element."))
449   | BoolLiteral(bl, _) ->
450       let name = create_elem_name env.scope in
```

```
      (match bl with
          True -> BoolLiteral(True, name), typ
        | False -> BoolLiteral(False, name), typ)
  | Field(fe) ->
      (match fe with
          Var(_) -> detail, typ
        | Attrib(_, _) -> detail, typ
        | Method (_, _) -> detail, typ
        | _ ->
          raise (SemError ("A list element is not a variable, " ^
                           "attribute, or method")))
  | Binop(e1, op, e2) ->
      raise (SemError ("A list element cannot be binary expression."))
  | Assign(fe, e) ->
      raise (SemError ("A list element cannot be assign expression."))
  | Call(vd_opt, fd, el) ->
      raise (SemError ("A list element cannot be call expression."))
  | Uminus(e) ->
      raise (SemError ("A list element cannot be unary minus expression.")
   )
  | Not(e) ->
      raise (SemError ("A list element cannot be not expression."))
  | Noexpr ->
      raise (SemError ("A list element cannot be an empty expression."))
  | Remove(_) ->
      raise (SemError ("A list element cannot be a remove expression."))
  | Place(_) ->
      raise (SemError ("A list element cannot be a place expression."))
  | _ -> detail, typ

and check_listlit env = function
    Ast.Elems(elems_list) ->
      let el = List.map (check_expr env) elems_list in
      let el = List.map (check_single_elem env) el in
      let e, typ = List.hd el in
      let typ = verify_elems_list_type env typ el in
      let name = create_ll_name env.scope in
      Elems(el,name), List_t(typ)
  | Ast.EmptyList ->
      EmptyList, List_t(Void)

and check_expr env = function
    Ast.IntLiteral(i) -> IntLiteral(i, ""), Int
  | Ast.StrLiteral(s) -> StrLiteral(s, ""), Str
  | Ast.ListLiteral(ll) ->
      let l, typ = check_listlit env ll in
      ListLiteral(l), typ
  | Ast.BoolLiteral(b) ->
      (match b with
          Ast.True -> BoolLiteral(True, ""), Bool
        | Ast.False -> BoolLiteral(False, ""), Bool)
  | Ast.VoidLiteral -> VoidLiteral, Void
  | Ast.Field(fe) ->
      let f, typ = check_field env [] fe in
```

```
504       Field(f), typ
505   | Ast.Binop(e1, op, e2) ->
506       let e1 = check_expr env e1
507       and e2 = check_expr env e2 in
508       if op = Ast.Add || op = Ast.Sub || op = Ast.Mult ||
509          op = Ast.Div || op = Ast.Mod then
510           let _, t1 = e1 and _, t2 = e2 in
511           let op = ast_op_to_sast_op op in
512           match t1, t2 with
513               Str, Str ->
514                 Binop(e1, op, e2), Str
515             | Int, Int -> Binop(e1, op, e2), Int
516             | _, _ ->
517               raise (SemError ("Additiona operation requires two integer "
    ^
518                                "or two string operands."))
519       else if op = Ast.Less || op = Ast.Leq ||
520               op = Ast.Greater || op = Ast.Geq then
521            ((require_int e1 "Left operand must be integer";
522              require_int e2 "Right operand must be integer";
523              let op = ast_op_to_sast_op op in
524              Binop(e1, op, e2), Bool))
525       else if op = Ast.Equal || op = Ast.Neq then
526         (require_same e1 e2 "Left and right operands in comparison must be
    equal";
527          let op = ast_op_to_sast_op op in
528          Binop(e1, op, e2), Bool)
529       else (* op = Ast.And || op = Ast.Or *)
530         (require_bool e1 "Left opreand must be boolean";
531          require_bool e2 "Right operand must be boolean";
532          let op = ast_op_to_sast_op op in
533          Binop(e1, op, e2), Bool)
534   | Ast.Assign(fd, e) ->
535       let field, tf = check_field env [] fd
536       and e = check_expr env e in
537       let _, te = e in
538       if tf = te then
539         Assign(field, e), tf
540       else
541         raise (SemError ("Types differ in assignment expression; expected:
    " ^
542                          string_of_t tf))
543   | Ast.Call(fd, el) ->
544       let actuals = List.map (check_expr env) el in
545       let fd, typ = check_field env actuals fd in
546       (match fd with
547            Fun(f) ->
548              (match f with
549                   BasicFunc(bf) ->
550                     verify_args f bf.formals actuals
551                 | AssertFunc(af) ->
552                     verify_args f af.aformals actuals);
553              Call(None, f, actuals), typ
554          | This -> raise (SemError ("Not callable: 'this'"))
```

```
555         | Var(v) -> raise (SemError ("Not callable: " ^ v.vname))
556         | Attrib(v1, v2) -> raise (SemError ("Not callable: " ^ v1.vname
    ^ "." ^ v2.vname))
557         | Grp(g) ->
558            let par =
559              { vname = g.gname ; vtype = Group(g.gname, None);
560                vinit = None; vloop = false }
561            in
562            let init =
563              List.find (fun f -> match f with
564                           BasicFunc(x) -> x.fname = "__init__"
565                         | AssertFunc(_) -> false) g.methods
566            in
567            let formals = match init with
568                BasicFunc(x) -> x.formals
569              | AssertFunc(_) -> raise (SemError ("Internal error:
    __init__ is an assert function"))
570            in
571            verify_args init formals actuals;
572            Call(Some(par), init, actuals), Group(g.gname, None)
573         | Method(par, child) ->
574              (match par.vtype with
575                  Group(s, _) -> ()
576                | _ -> raise (SemError ("Method call with parent that is
    not a group")));
577              (match child with
578                  BasicFunc(bf) ->
579                    verify_args child bf.formals actuals
580                | AssertFunc(af) ->
581                    verify_args child af.aformals actuals);
582              Call(Some(par), child, actuals), typ)
583   | Ast.Element(e1, e2) ->
584       let e1 = check_expr env e1
585       and e2 = check_expr env e2 in
586       let _, t1 = e1 in
587       require_int e2 "Integer expected for element number";
588       (match t1 with
589           List_t(typ) -> Element(e1, e2), typ
590         | _ -> raise (SemError ("Expression not subscriptable")))
591   | Ast.Uminus(e) ->
592       let e = check_expr env e in
593       require_int e "Operand must be integer";
594       Uminus(e), Int
595   | Ast.Not(e) ->
596       let e = check_expr env e in
597       require_bool e "Operand must be boolean";
598       Not(e), Bool
599   | Ast.Noexpr ->
600       Noexpr, Void
601   | Ast.Remove(fd1, ll) ->
602       let board_name = match fd1 with
603           Ast.Id(s) -> s
604         | _ -> raise (SemError "Not implemented")
605       in
```

```
      let toi_call = Ast.Call(Ast.FieldCall(Ast.Id(board_name), "toi"),
                              [Ast.ListLiteral(ll)])
      in
      let rmv_call = Ast.Call(Ast.FieldCall(Ast.Id(board_name), "remove"),
                              [toi_call])
      in
      let rmv_call = check_expr env rmv_call in
      let fd1, _ = check_field env [] fd1 in
      let checked_ll, ll_typ = check_listlit env ll in
      require_parent env "Board" fd1 "Board (sub)group expected";
      require_integer_list (checked_ll, ll_typ) "List of integers expected
    ";
      Remove(rmv_call), Bool
  | Ast.Place(fd1, fd2, ll) ->
      let board_name = match fd2 with
          Ast.Id(s) -> s
        | _ -> raise (SemError "Not implemented")
      in
      let toi_call = Ast.Call(Ast.FieldCall(Ast.Id(board_name), "toi"),
                              [Ast.ListLiteral(ll)])
      in
      let plc_call = Ast.Call(Ast.FieldCall(Ast.Id(board_name), "place"),
                              [Ast.Field(fd1); toi_call])
      in
      let plc_call = check_expr env plc_call in
      let fd1, _ = check_field env [] fd1
      and fd2, _ = check_field env [] fd2
      and ll, ll_typ = check_listlit env ll in
      require_parent env "Piece" fd1 "Piece (sub)group expected";
      require_parent env "Board" fd2 "Board (sub)group expected";
      require_integer_list (ll, ll_typ) "List of integers expected";
      Place(plc_call), Bool

let rec verify_expr_list_type env typ = function
    [] -> typ
  | (a, b) :: rest ->
      if b = typ then
        verify_expr_list_type env typ rest
      else
        raise (SemError ("List of expressions are not all of same type: "
    ^
                         string_of_t typ))

(* let verify_group_cast env to_name from_name =
  let rec helper gdcl =
    if gdcl.gname = to_name then
      ()
    else
      (match gdcl.extends with
          Some(g) -> helper g
        | None -> raise (SemError ("Invalid group cast: " ^ to_name ^
                         " is not an ancestor of " ^ from_name)))
  in
  let from_grp = find_group env.scope from_name in
```

```
  helper from_grp *)

let check_init env v_name v_typ = function
    Ast.ExprInit(e) ->
      let e = check_expr env e in
      let _, typ = e in
      if v_typ = typ then
        Some(e)
      else
        (match v_typ, typ with
          Group(to_, _), Group(from, _) ->
            (* verify_group_cast env to_ from; *)
            Some(e)
        | _, _ ->
            raise (SemError ("Variable initiation type for identifer " ^ v_name ^
                             " does not match, expected: " ^ string_of_t v_typ)))
  | Ast.NoInit -> None

let rec find_turn_name (scope : symbol_table) name =
  try
    List.find (fun t -> t = name) scope.turns
  with Not_found ->
    match scope.parent with
      Some(parent) -> find_turn_name parent name
    | _ -> raise Not_found

let rec check_stmt env = function
    Ast.Block(sl) ->
      let scope' =
        { parent = Some(env.scope);
          variables = [];
          functions = [];
          groups = [];
          turns = [];
          ll_count = 0;
          elem_count = 0 } in
      let env' =
        { env with scope = scope'; } in
      let sl = List.map (fun s -> check_stmt env' s) sl in
      Block(scope', sl)
  | Ast.Expr(e) -> Expression(check_expr env e)
  | Ast.Pass(s, e) ->
      let turn_name =
        try
          find_turn_name env.scope s
        with Not_found ->
          raise (SemError ("Turn name not found: " ^ s))
      in
      let dummy_turn_func =
        { ftype = Void;
          fname = turn_name;
          formals = [];
```

```
          locals = [];
          body = [];
          turns_func = true;
          group_method = "";
          f_is_built_in = false }
      in
      let e = check_expr env e in
      require_int e "Must pass to an integer player.";
      Pass(BasicFunc(dummy_turn_func), e)
  | Ast.Return(e) ->
      let e = check_expr env e in
      let _, typ = e in
      if env.return_type <> typ then
        raise (SemError ("Return types do not match; expected: " ^
                         string_of_t env.return_type))
      else
        Return(e)
  | Ast.Break ->
      if env.in_loop = false then
        raise (SemError ("Break outside of loop"))
      else
        Break
  | Ast.Continue ->
      if env.in_loop = false then
        raise (SemError ("Continue outside of loop"))
      else
        Continue
  | Ast.End -> End
  | Ast.If(e, s1, e_opt, s2) ->
      let e = check_expr env e in
      let e_opt = match e_opt with
          None -> None
        | Some(expr) -> Some(check_expr env expr)
      in
      require_bool e "Predicate of if must be boolean";
      If(e, check_stmt env s1, e_opt, check_stmt env s2)
  | Ast.For(vd, el, s) ->
      let name, t_vd = vd.Ast.vname, id_type_to_t vd.Ast.vtype in
      let decl =
      { vname = name;
        vtype = t_vd;
        vinit = check_init env name t_vd vd.Ast.vinit;
        vloop = true }
      and el = List.map (check_expr env) el in
      let _, t_el = List.hd el in
      let t_el = verify_expr_list_type env t_el el in
      if t_vd <> t_el then
        raise (SemError ("For loop elements and loop variable must be " ^
                         "the same type. Variable type: " ^ string_of_t
 t_vd ^
                         "; List type: " ^ string_of_t t_el))
      else
        let scope' =
          { parent = Some(env.scope);
```

```
          variables = [];
          functions = [];
          groups = [];
          turns = [];
          ll_count = 0;
          elem_count = 0 } in
        let env' = { env with scope = scope'; in_loop = true } in
        scope'.variables <- decl :: scope'.variables;
        For(decl, el, check_stmt env' s)
  | Ast.While(e, s) ->
      let e = check_expr env e in
      let scope' =
        { parent = Some(env.scope);
          variables = [];
          functions = [];
          groups = [];
          turns = [];
          ll_count = 0;
          elem_count = 0 } in
      let env' = { env with scope = scope'; in_loop = true } in
      require_bool e "While loop predicate must be Boolean";
      While(e, check_stmt env' s)

let rec check_for_begin(turns_section) = match turns_section with
    [] -> false
  | Ast.BasicFunc(f) :: rest ->
      if f.Ast.fname = "begin" then
        true
      else
        check_for_begin(rest)
  | Ast.AssertFunc(f) :: rest -> check_for_begin(rest)

let require_no_init = function
    Ast.NoInit -> ()
  | Ast.ExprInit(e) ->
      raise (SemError "Function formal arguments cannot have default
   values.")

let require_non_void v = match v.vtype with
    Void -> raise (SemError (v.vname ^ " declared with void type"))
  | _ -> ()

let verify_not_redeclaring scope name =
  let is_var_or_fun_in_scope =
    List.exists (fun x -> x.vname = name) scope.variables ||
    List.exists (fun x -> match x with
                    BasicFunc(f) -> f.fname = name
                  | AssertFunc(f) -> f.aname = name) scope.functions
  in
  let gdcl_opt =
    try
      Some(find_group scope name)
    with Not_found ->
      None
```

```
  in
  match gdcl_opt with
    Some(_) ->
      raise (SemError ("Cannot redeclare an identifier that is also a group " ^
                       "definition name: " ^ name))
  | None ->
    if is_var_or_fun_in_scope then
      raise (SemError ("Cannot redeclare an indentifer matching with a " ^
                       "variable or a function in current scope: " ^ name))
    else
      ()

let check_vdcl_helper env v init_ok =
  let name = v.Ast.vname in
  let t_vd = id_type_to_t v.Ast.vtype in
  let init = check_init env name t_vd v.Ast.vinit in
  let init =
    (match init with
        None -> init
      | _ ->
        if not init_ok then
          raise (SemError ("Initiation not allowed here for variable: " ^ name))
        else
          init)
  in
  let decl =
    { vname = name;
      vtype = t_vd;
      vinit = init;
      vloop = false }
  in
  verify_not_redeclaring env.scope v.Ast.vname;
  require_non_void decl;
  decl

let check_formal env v =
  let decl = check_vdcl_helper env v false in
  let decl = {decl with vinit = None} in
  require_no_init v.Ast.vinit;
  env.scope.variables <- decl :: env.scope.variables;
  decl

let check_vdcl env v =
  let decl = check_vdcl_helper env v true in
  env.scope.variables <- decl :: env.scope.variables;
  decl

let rec verify_implicit_return_basic_fun name ftyp body=
  let last_stmt =
    try
      List.hd (List.rev body)
```

```
    with Failure("hd") ->
      End (* empty body, so use non-Return dummy stmt *)
  in
  match last_stmt with
    Return(e) ->
      () (* Last statement is not an implicit return *)
  | Block(scope, sl) ->
      verify_implicit_return_basic_fun name ftyp sl
  | If(e, s1, e_opt, s2) ->
      verify_implicit_return_basic_fun name ftyp [s1];
      verify_implicit_return_basic_fun name ftyp [s2]
  | For(vd, el, s) ->
      verify_implicit_return_basic_fun name ftyp [s]
  | While(e, s) ->
      verify_implicit_return_basic_fun name ftyp [s]
  | _ ->
    (match ftyp with
        Void -> ()
      | _ -> raise (SemError ("function " ^ name ^ " implicit return " ^
               "invalid type, expected: " ^  string_of_t ftyp)))

let rec verify_return_pass_end_turn_fun name body =
  let error_on_return_stmt = function
      Return(_) ->
        raise (SemError ("Return statement in turn function: " ^ name))
    | _ -> ()
  in
  let last_stmt =
    try
      List.hd (List.rev body)
    with Failure("hd") ->
      raise (SemError ("Turn function " ^ name ^
             " does not end with 'end' or 'pass' statement."))
  in
  List.iter error_on_return_stmt body;
  match last_stmt with
      Pass(fd, e) -> ()
    | End -> ()
    | Block(scope, sl) ->
        verify_return_pass_end_turn_fun name sl
    | If(e, s1, e_opt, s2) ->
        verify_return_pass_end_turn_fun name [s1];
        verify_return_pass_end_turn_fun name [s2]
    | For(vd, el, s) ->
        verify_return_pass_end_turn_fun name [s]
    | While(e, s) ->
        verify_return_pass_end_turn_fun name [s]
    | _ -> raise (SemError ("Turn function " ^ name ^
                  " does not end with 'end' or 'pass' statement."))

let verify_implicit_return = function
    AssertFunc(f) ->
        () (* Implicit and explicit returns always Bool *)
  | BasicFunc(f) ->
```

```
        if f.turns_func then
          verify_return_pass_end_turn_fun f.fname f.body
        else
          verify_implicit_return_basic_fun f.fname f.ftype f.body

let check_basic_func env in_turn_section (f : Ast.basic_func_decl) =
    let scope' =
      { parent = Some(env.scope);
        variables = [];
        functions = [];
        groups = [];
        turns = [];
        ll_count = 0;
        elem_count = 0 } in
    let env' =
      { env with scope = scope';
        return_type = id_type_to_t f.Ast.ftype; } in
    let fl = List.map (fun v -> check_formal env' v) f.Ast.formals in
    let ll = List.map (fun dcl -> check_vdcl env' dcl) f.Ast.locals in
    let sl = List.map (fun s -> check_stmt env' s) f.Ast.body in
    let gname = (match env.partial_group_info with None -> "" | Some(g) ->
    g.group_name) in
    let fdecl =
      BasicFunc({ ftype = id_type_to_t f.Ast.ftype;
                  fname = f.Ast.fname;
                  formals = fl;
                  locals = ll;
                  body = sl;
                  turns_func = in_turn_section;
                  group_method = gname;
                  f_is_built_in = false })
    in
    verify_implicit_return fdecl;
    env.scope.functions <- fdecl :: env.scope.functions;
    fdecl

let rec verify_assert_func_stmt stmt =
  let msg_bool = "Assert function expression statment or return statement"
    ^
                 " must be of type boolean."
  in
  let msg_stmt = "Assert function statment cannot be " in
  match stmt with
      Block(scope, sl) -> List.iter verify_assert_func_stmt sl
    | Expression(e) -> require_bool e msg_bool
    | Return(e) -> require_bool e msg_bool
    | Break -> ()
    | Continue -> ()
    | End -> raise (SemError (msg_stmt ^ "end."))
    | Pass(_, _) -> raise (SemError (msg_stmt ^ "pass."))
    | If(e, s1, e_opt, s2) ->
        verify_assert_func_stmt s1;
        verify_assert_func_stmt s2
    | For(vd, el, s) -> verify_assert_func_stmt s
```

```
    | While(e, s) -> verify_assert_func_stmt s

let check_assert_func env in_turn_section (f : Ast.assert_decl) =
  let scope' =
    { parent = Some(env.scope);
      variables = [];
      functions = [];
      groups = [];
      turns = [];
      ll_count = 0;
      elem_count = 0 } in
  let env' =
    { env with scope = scope';
      return_type = Bool; } in
  let fl = List.map (fun v -> check_formal env' v) f.Ast.formals in
  let ll = List.map (fun dcl -> check_vdcl env' dcl) f.Ast.locals in
  let sl = List.map (fun s -> check_stmt env' s) f.Ast.body in
  let ret_stmt = Return(BoolLiteral(True, ""), Bool) in
  let sl = ret_stmt :: List.rev sl in
  let sl = List. rev sl in
  let gname = (match env.partial_group_info with None -> "" | Some(g) -> g
   .group_name) in
  let fdecl =
    AssertFunc({ aname = f.Ast.fname;
                 aformals = fl;
                 alocals = ll;
                 abody = sl ;
                 a_turns_func = in_turn_section;
                 a_group_method = gname;
                 a_is_built_in = false })
  in
  List.iter verify_assert_func_stmt sl;
  env.scope.functions <- fdecl :: env.scope.functions;
  fdecl

let check_function env in_turn_section = function
    Ast.BasicFunc(f) ->
      verify_not_redeclaring env.scope f.Ast.fname;
      check_basic_func env in_turn_section f
  | Ast.AssertFunc(f) ->
      verify_not_redeclaring env.scope f.Ast.fname;
      check_assert_func env in_turn_section f

let find_init_func methods =
    List.find (fun x -> match x with
                    BasicFunc(f) -> f.fname = "__init__"
                  | AssertFunc(f) -> false) methods

let verify_extends parent par_actuals init_opt =
  match parent with
      Some(p) ->
        (match par_actuals with
            Some(el) ->
              (match init_opt with
```

```
                  Some(init) ->
                    raise (SemError ("Defined child constructor while also using parent's constructor"))
                | None ->
                    let par_init = try
                      find_init_func p.methods
                    with Not_found ->
                      raise (SemError("Constructor function not found in parent"))
                    in
                    let fdcl = par_init in
                    let par_init = match par_init with
                        BasicFunc(f) -> f
                      | AssertFunc(f) ->
                          raise (SemError("Parent constructor cannot be an assert function"))
                    in
                    if List.length par_init.formals = List.length el then
                      verify_args fdcl par_init.formals el
                    else
                      raise (SemError("Number of constrctur variables for parent's constructor does not match")))
          | None ->
              match init_opt with
                  Some(init) -> ()
                | None -> () )
    | None ->
        (match init_opt with
            Some(init) -> ()
          | None -> raise (SemError("No constructor function")))

let rec verify_single_attrib par_attr v = match par_attr with
    [] -> Some(v)
  | p :: rest ->
      if p.vname = v.vname then
        if p.vtype = v.vtype then
          None
        else
          raise (SemError("Cannot change type of inherited attribute"))
      else
        verify_single_attrib rest v

let rec verify_attributes par_attr = function
    [] -> []
  | v :: rest ->
      (match verify_single_attrib par_attr v with
          None -> verify_attributes par_attr rest
        | Some(var) -> var :: verify_attributes par_attr rest)

let check_attrib env v =
  let decl = check_vdcl_helper env v false in
  let name = decl.vname in
  let scope =
    (match env.partial_group_info with
```

```
1076        None -> raise (SemError "Internal error: check_attrib called
    outside of group definition.")
1077      | Some(info) -> info.symbols) in
1078  verify_not_redeclaring scope name;
1079  scope.variables <- decl :: scope.variables;
1080  decl
1081
1082let check_method env new_fun =
1083  let fdcl = check_function env false new_fun in
1084  let name = match fdcl with
1085      BasicFunc(f) -> f.fname
1086    | AssertFunc(f) -> f.aname
1087  in
1088  let scope =
1089    (match env.partial_group_info with
1090        None -> raise (SemError "Internal error: check_attrib called
    outside of group definition.")
1091      | Some(info) -> info.symbols) in
1092  verify_not_redeclaring scope name;
1093  scope.functions <- fdcl :: scope.functions;
1094  fdcl
1095
1096let add_parent_init parent par_actuals name methods = function
1097    Some(init_fun) ->
1098      (match init_fun with
1099          BasicFunc(f) ->
1100            if f.ftype = Group(name, None) then
1101              methods
1102            else
1103              raise (SemError ("Group " ^ name ^ " __init__ function has
    type not equal to itself"))
1104        | AssertFunc(f) ->
1105            raise (SemError ("Group " ^ name ^ " __init__ function not a
    basic function")))
1106  | None ->
1107    (match parent with
1108        None -> raise (SemError "No parent but using parent __init__")
1109      | Some(par) ->
1110        let par_init =
1111          try
1112            find_init_func par.methods
1113          with Not_found ->
1114            raise (SemError ("Parent __init__ function not found for child
    : " ^
1115                             name))
1116        in
1117        let child_init =
1118          (match par_actuals with
1119            Some(el) ->
1120            (match par_init with
1121                AssertFunc(f) -> raise (SemError "Assert Function used as
    parent __init__")
1122              | BasicFunc(f) ->
1123                let dcls =
```

```
                  List.map2 (fun vdcl act -> {vdcl with vinit = Some(act)}) f.formals el
                in
                { ftype = Group(name, None);
                  fname = "__init__";
                  formals = [];
                  locals = dcls;
                  body = f.body;
                  turns_func = false;
                  group_method = name;
                  f_is_built_in = false })
          | None ->
            (match par_init with
                AssertFunc(f) -> raise (SemError "Assert Function used as parent __init__")
              | BasicFunc(f) ->
                { f with ftype = Group(name, None); group_method = name })
      in
      BasicFunc(child_init) :: methods)

let verify_repr gname = function
    BasicFunc(f) ->
      if f.ftype <> Str then
        raise (SemError ("Group " ^ gname ^ " __repr__() does not have Str return type"))
      else if List.length f.formals > 0 then
        raise (SemError ("Group " ^ gname ^ " __repr__() has more than 0 formals"))
      else
        ()
  | AssertFunc(f) ->
      raise (SemError ("Group " ^ gname ^ " __repr__() cannot be an assert function"))

let rec find_repr gdcl =
  try
    List.find
      (fun f ->
        (match f with
           BasicFunc(x) -> x.fname = "__repr__"
         | AssertFunc(x) -> x.aname = "__repr__")) gdcl.methods
  with Not_found ->
    match gdcl.extends with
        Some(par) -> find_repr par
      | None -> raise Not_found

let rec instance_of name gdcl =
  if gdcl.gname = name then
    true
  else
    match gdcl.extends with
      None -> false
    | Some(par) -> instance_of name par
```

```
1172
1173 let check_for_repr env gdcl =
1174   try
1175     (* search locally first *)
1176     let repr = find_repr { gdcl with extends = None } in
1177     verify_repr gdcl.gname repr;
1178     gdcl
1179   with Not_found ->
1180   try
1181     (* not in group, search any parents *)
1182     let repr = find_repr gdcl in
1183     let built_in = match repr with
1184         BasicFunc(f) -> f.f_is_built_in
1185       | AssertFunc(f) -> raise (SemError ("Group " ^ gdcl.gname ^ "
    __repr__() cannot be an assert function"))
1186     in
1187     let not_a_piece = not (instance_of "Piece" gdcl) in
1188     let repr =
1189       if built_in && not_a_piece then
1190         let s = "<Group " ^ gdcl.gname ^ " instance>" in
1191         let body = Return(StrLiteral(s, ""), Str) in
1192         BasicFunc({ ftype = Str;
1193                     fname = "__repr__";
1194                     formals = [];
1195                     locals = [];
1196                     body = [body];
1197                     turns_func = false;
1198                     group_method = gdcl.gname;
1199                     f_is_built_in = true})
1200       else
1201         repr
1202     in
1203     verify_repr gdcl.gname repr;
1204     { gdcl with methods = repr :: gdcl.methods }
1205   with Not_found ->
1206     let s = "<Group " ^ gdcl.gname ^ " instance>" in
1207     let body = Return(StrLiteral(s, ""), Str) in
1208     let repr =
1209       { ftype = Str;
1210         fname = "__repr__";
1211         formals = [];
1212         locals = [];
1213         body = [body];
1214         turns_func = false;
1215         group_method = gdcl.gname;
1216         f_is_built_in = true }
1217     in
1218     { gdcl with methods = BasicFunc(repr) :: gdcl.methods }
1219
1220 let rec check_group env g =
1221   let parent = match g.Ast.extends with
1222       Some(fe) ->
1223         let par, _ = check_field env [] fe in
1224         (match par with
```

```
            Grp(p) -> Some(p)
          | _ -> raise (SemError ("Parent is not a group")))
    | None ->
        if g.Ast.gname = "Object" then
          None
        else
          raise (SemError "Must inherit from Object or another group")
  in
  let par_actuals = match g.Ast.par_actuals with
      Some(el) -> Some(List.map (check_expr env) el)
    | None -> None
  in
  let scope' =
      { parent = Some(env.scope);
        variables = [];
        functions = [];
        groups = [];
        turns = [];
        ll_count = 0;
        elem_count = 0 } in
  let partial_scope = {scope' with parent = None} in
  let info =
    { group_name = g.Ast.gname;
      symbols = partial_scope;
      par = parent } in
  let env' =
    { env with scope = scope';
      return_type = Void;
      partial_group_info = Some(info) } in
  let attribs = List.map (check_attrib env') g.Ast.attributes in
  let attribs =
    (match parent with
       Some(par) -> verify_attributes par.attributes attribs
     | None -> attribs) in
  let methods = List.map (check_method env') g.Ast.methods in
  let init_opt = try
    Some(find_init_func methods)
  with Not_found ->
    None
  in
  let methods = add_parent_init parent par_actuals g.Ast.gname methods
   init_opt in
  verify_extends parent par_actuals init_opt;
  let gdecl =
    { gname = g.Ast.gname;
      extends = parent;
      par_actuals = par_actuals;
      attributes = attribs;
      methods = methods }
  in
  let gdecl = check_for_repr env gdecl in
  verify_not_redeclaring env.scope gdecl.gname;
  env.scope.groups <- gdecl :: env.scope.groups;
  gdecl
```

```
1278
1279 let require_no_init_list_groups v = match v.vtype, v.vinit with
1280     Group(_, _), Some(_) ->
1281       raise(SemError ("Cannot initialize groups in @setup"))
1282   | List_t(_), Some(_) ->
1283       raise(SemError ("Cannot initialize lists in @setup"))
1284   | _, _ -> ()
1285
1286 let check_setup env setup_section =
1287   let vars, funcs, groups = setup_section in
1288   let v = List.map (check_vdcl env) vars in
1289   let g = List.map (check_group env) groups in
1290   let f = List.map (check_function env false) funcs in
1291   ignore(List.map require_no_init_list_groups v);
1292   v, f, g
1293
1294 let rec gather_turn_names env = function
1295     [] -> ()
1296   | Ast.BasicFunc(t) :: rest ->
1297       env.scope.turns <- t.Ast.fname :: env.scope.turns;
1298       gather_turn_names env rest
1299   | Ast.AssertFunc(t) :: rest ->
1300       raise (SemError ("Assert function cannot be a turns function: " ^ t.
    Ast.fname ))
1301
1302 let check_turns env turns_section =
1303   if check_for_begin(turns_section) = false then
1304     raise (SemError "No begin() function in @turns section")
1305   else
1306     gather_turn_names env turns_section;
1307     List.map (check_function env true) turns_section
1308
1309 let fix_pass_stmts_helper turns = function
1310     Pass(s, e) ->
1311       let name = (match s with
1312                     BasicFunc(x) -> x.fname
1313                   | AssertFunc(x) -> x.aname)
1314       in
1315       let fdcl =
1316         try
1317           List.find
1318             (fun f -> match f with
1319                BasicFunc(x) -> x.fname = name
1320              | AssertFunc(x) -> x.aname = name)
1321           turns
1322         with Not_found ->
1323           raise (SemError ("Turns function declaration not found in pass
    statement: " ^ name ))
1324       in
1325       Pass(fdcl, e)
1326   | s -> s
1327
1328 let fix_pass_stmts turns = function
1329     BasicFunc(f) ->
```

```
    let new_f =
      {f with body = List.map (fix_pass_stmts_helper turns) f.body}
    in
    BasicFunc(new_f)
  | AssertFunc(f) ->
    let new_f =
      {f with abody = List.map (fix_pass_stmts_helper turns) f.abody}
    in
    AssertFunc(new_f)

let check_program (program : Ast.program) =
  let symbols =
    { parent = None ;
      variables = Stdlib.vars;
      functions = Stdlib.funcs;
      groups = Stdlib.grps;
      turns = [];
      ll_count = 0;
      elem_count = 0 } in
  let env =
    { scope = symbols;
      return_type = Void;
      in_loop = false;
      partial_group_info = None } in
  let setup_section, turns_section = program in
  let setup_section = check_setup env setup_section in
  let turns_section = check_turns env turns_section in
  let turns_section = List.map (fix_pass_stmts turns_section)
   turns_section in
  setup_section, turns_section
```

./scanner.mll

```
{ open Parser
  exception LexError of string * Lexing.lexbuf }

rule token = parse
  '\n'            {Lexing.new_line lexbuf; token lexbuf} (* http://caml.
   inria.fr/pub/docs/manual-ocaml/libref/Lexing.html; http://courses.
   softlab.ntua.gr/compilers/ocamlyacc-tutorial.pdf *)
| [' ' '\t' '\r'] { token lexbuf } (* Whitespace *)
| "#"       { comment lexbuf }           (* Comment until EOL *)
| '"'       { str (Buffer.create 20) lexbuf }  (* String until next
   unescaped quote *)
| '.'       { DOT }
| '('       { LPAREN }
| ')'       { RPAREN }
| '['       { LBRACKET }
| ']'       { RBRACKET }
| '{'       { LBRACE }
| '}'       { RBRACE }
| ';'       { SEMI }
| ','       { COMMA }
| '+'       { PLUS }
| '-'       { MINUS }
```

```
| '*'       { TIMES }
| '/'       { DIVIDE }
| '%'       { MOD }
| '='       { ASSIGN }
| "=="      { EQ }
| "!="      { NEQ }
| '<'       { LT }
| "<="      { LEQ }
| ">"       { GT }
| ">="      { GEQ }
| "if"      { IF }
| "else"    { ELSE }
| "elif"    { ELIF }
| "for"     { FOR }
| "while"   { WHILE }
| "break"   { BREAK }
| "continue" { CONTINUE }
| "end"     { END }
| "return" { RETURN }
| "True"    { TRUE }
| "False"   { FALSE }
| "None"    { NONE }
| "in"      { IN }
| "int"     { INT }
| "str"     { STR }
| "bool"    { BOOL }
| "void"    { VOID }
| "list"    { LIST }
| "group"   { GROUP }
| "and"     { AND }
| "or"      { OR }
| "not"     { NOT }
| "assert" { ASSERT }
| "<<"      { REMOVE }
| ">>"      { PLACE }
| "func"    { FUNC }
| "pass"    { PASS }
| "@setup" { SETUP }
| "@turns" { TURNS }
| "this"    { THIS }
| ['0'-'9']+ as lxm { INTLITERAL(int_of_string lxm) }
| ['a'-'z' 'A'-'Z' '_']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| eof { EOF }
| _ as char
    { let msg = "Lexical error: Illegal character: " ^ Char.escaped char
    in
      raise (LexError(msg, lexbuf)) }

and comment = parse
  '\n' { Lexing.new_line lexbuf; token lexbuf }
| _    { comment lexbuf }

and str buf = parse
  (**
```

```
   * String parsing. Modified version of:
   * https://realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html
   *)
  '"'       { STRLITERAL (Buffer.contents buf) }
| '\\' '\\' { Buffer.add_char buf '\\'; str buf lexbuf }
| '\\' 'n'  { Buffer.add_char buf '\n'; str buf lexbuf }
| '\\' '"'  { Buffer.add_char buf '\"'; str buf lexbuf }
| [^ '"' '\\']+ as lxm
  { Buffer.add_string buf (lxm);
    str buf lexbuf
  }
| _  as char
    { let msg = "Lexical error: Illegal string character: " ^ Char.escaped char in
      raise (LexError (msg, lexbuf)) }
| eof
    { let msg = "Lexical error: String is not terminated" in
      raise (LexError (msg, lexbuf)) }
```

./senet.ml

```
open Lexing

type action = Ast | Semantic | Groupeval | Compile

(**
  * Printing of error line based on:
  * https://realworldocaml.org/v1/en/html/parsing-with-ocamllex-and-menhir.html
  * http://courses.softlab.ntua.gr/compilers/ocamlyacc-tutorial.pdf
  * http://stackoverflow.com/questions/14046392/verbose-error-with-ocamlyacc?lq=1
  *)

let string_of_error msg lb =
  let pos = lexeme_start_p lb in
  let line = pos.pos_lnum in
  let first_char = lexeme_start lb - pos.pos_bol + 1 in
  let last_char = lexeme_end lb - pos.pos_bol + 1 in

  "line "^ string_of_int line ^ ", " ^
  "characters " ^ string_of_int first_char ^ "-" ^
                  string_of_int last_char ^":\n" ^
  msg


let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [ ("-a", Ast) ;
            ("-s", Semantic) ;
            ("-g", Groupeval) ;
            ("-c", Compile) ]
  else Compile in
  let lexbuf = Lexing.from_channel stdin in
```

```
32
33  try
34
35    let program = Parser.program Scanner.token lexbuf in
36
37      match action with
38        Ast -> let listing = Ast.string_of_program program
39               in print_string listing
40      | Semantic -> let checked_program = Sast.check_program program
41               in ignore(checked_program);
42                  print_string ("Success!\n")
43      | Groupeval -> let checked_program = Sast.check_program program in
44                     let checked_program = Cast.build_cast checked_program
    in
45                     print_string (Cast.string_of_program checked_program)
46      | Compile -> let checked_program = Sast.check_program program in
47                   Compile.translate (Cast.build_cast checked_program)
48
49  with Scanner.LexError(msg,lb) ->
50          print_string (string_of_error msg lb); print_newline ()
51     | Parser.Error ->
52          let msg = "Syntax error: " ^ Lexing.lexeme lexbuf in
53          print_string (string_of_error msg lexbuf); print_newline ()
54     | Sast.SemError(msg) ->
55          let msg = "Semantic error: " ^ msg in
56          print_string msg; print_newline ()
```

./stdlib.ml

```
1  open Types
2
3  let funcs =
4    let s =
5      { vname = "print_arg";
6        vtype = Str;
7        vinit = None;
8        vloop = false }
9    in
10   let b = { s with vtype = Bool } in
11   let i = { s with vtype = Int } in
12   let g = { s with vtype = Group("", None) } in
13   let print_str =
14     { ftype = Void;
15       fname = "print";
16       formals = [s];
17       locals = [];
18       body = [];
19       turns_func = false;
20       group_method = "";
21       f_is_built_in = true }
22   in
23   let print_bool = { print_str with formals = [b] } in
24   let print_int = { print_str with formals = [i] } in
25   let print_group = { print_str with formals = [g] } in
```

```
  let read = { print_str with fname = "read"; ftype = Str; formals = [i] }
    in
  let clear = { print_str with fname = "clear_input"; ftype = Void;
   formals = [] } in
  let stoi = { print_str with fname = "stoi"; ftype = Int; formals = [s] }
    in
  let exit = { print_str with fname = "exit"; ftype = Void; formals = [] }
    in
  let rand = { print_str with fname = "rand"; ftype = Int; formals = [] }
   in
  [BasicFunc(print_str); BasicFunc(print_bool); BasicFunc(print_int);
   BasicFunc(print_group); BasicFunc(read); BasicFunc(stoi); BasicFunc(
   exit);
   BasicFunc(clear); BasicFunc(rand)]

let vars =
  let init = IntLiteral(0, "") in
  let v = { vname = "PLAYER_ON_MOVE" ; vinit = Some(init, Int) ;
            vtype = Int; vloop = false }
  in
  [v]

let base_init name =
  let stmt = Return(Field(This), Group(name, None)) in
  { ftype = Group(name, None);
    fname = "__init__";
    formals = [];
    locals = [];
    body = [stmt];
    turns_func = false;
    group_method = name;
    f_is_built_in = true }

let base_repr name =
  let s = "<Group " ^ name ^ " instance>" in
  let body = Return(StrLiteral(s, ""), Str) in
  BasicFunc({ ftype = Str;
              fname = "__repr__";
              formals = [];
              locals = [];
              body = [body];
              turns_func = false;
              group_method = name;
              f_is_built_in = true})

let obj =
  let name = "Object" in
  { gname = name;
    extends = None;
    par_actuals = None;
    attributes = [];
    methods = [BasicFunc(base_init name)] }

let board =
```

```
74    let v =
75      { vname = "cells"; vtype = List_t(Group("Piece", None));
76        vinit = None; vloop = false }
77    in
78    let occupied = { v with vname ="occupied"; vtype = List_t(Bool) } in
79    let attr = [v; occupied] in
80    let v = { v with vname = "x"; vtype = Int } in
81    let remove =
82      { ftype = Bool; fname = "remove"; formals = [v]; locals = []; body =
       [];
83        turns_func = false; group_method = "Board"; f_is_built_in = true }
84    in
85    let owns = { remove with ftype = Int; fname = "owns" } in
86    let v = { v with vname = "l"; vtype = List_t(Int) } in
87    let toi = { owns with fname = "toi"; formals = [v] } in
88    let tol = { owns with ftype = List_t(Int); fname = "tol" } in
89    let full = { owns with ftype = Bool; fname = "full"; formals = [] } in
90    let x = { v with vname = "x"; vtype = Int } in
91    let p = { v with vname = "p"; vtype = Group("Piece", None) } in
92    let place =
93      { ftype = Bool; fname = "place"; formals = [p; x]; locals = []; body =
       [];
94        turns_func = false; group_method = "Board"; f_is_built_in = true }
95    in
96    let meth =
97      [BasicFunc(remove); BasicFunc(owns); BasicFunc(owns);
98       BasicFunc(toi); BasicFunc(tol); BasicFunc(base_init "Board");
99       base_repr "Board"; BasicFunc(full); BasicFunc(place)]
100   in
101   { obj with gname = "Board"; extends = Some(obj);
102     attributes = attr; methods = meth }
103
104 let piece =
105   let v = { vname = ""; vtype = Void; vinit = None; vloop = false } in
106   let owner = { v with vname = "owner"; vtype = Int } in
107   let fixed = { v with vname = "fixed"; vtype = Bool } in
108   let s = { v with vname = "s"; vtype = Str } in
109   let this_dummy =
110     { vname = "this"; vtype = Group("Loop", None);
111       vinit = None; vloop = false }
112   in
113   let stmts =
114     [Expression(Assign(Attrib(this_dummy, s), (Field(Var(s)), Int)), Int);
115      Return(Field(This), Group("Piece", None))]
116   in
117   let init = base_init "Piece" in
118   let init = { init with body = stmts; formals = [s] } in
119   let stmts =
120     [Return(Field(Attrib(this_dummy, s)), Str)]
121   in
122   let repr = base_repr "Piece" in
123   let repr =
124     match repr with
125       BasicFunc(f) -> { f with body = stmts }
```

```
    | _ -> raise (Failure ("__repr__ for piece matched to non-basicfunc"))
  in
  { obj with gname = "Piece"; extends = Some(obj);
    attributes = [owner; fixed; s];
    methods = [BasicFunc(base_init "Piece"); BasicFunc(init); BasicFunc(
   repr)] }

let boards_lib =
  let x = { vname = "x"; vtype = Int; vinit = None; vloop = false } in
  let y = { x with vname = "y" } in
  let this_dummy =
    { vname = "this"; vtype = Group("Rect", None);
      vinit = None; vloop = false }
  in
  let init_cells =
    { ftype = Void;
      fname = "INIT_CELLS";
      formals = [this_dummy; x];
      locals = [];
      body = [];
      turns_func = false;
      group_method = "Board";
      f_is_built_in = true }
  in
  let stmts =
    [Expression(Assign(Attrib(this_dummy, x), (Field(Var(x)), Int)), Int);
     Expression(Assign(Attrib(this_dummy, y), (Field(Var(y)), Int)), Int);
     Expression(Call(Some(this_dummy), BasicFunc(init_cells),
                     [Binop((Field(Var(x)), Int), Mult,
                            (Field(Var(y)), Int)), Int]), Void);
     Return(Field(This), Group("Rect", None))]
  in
  let init = base_init "Rect" in
  let init = { init with formals = [x; y]; body = stmts } in
  let coord = { x with vname = "coord"; vtype = List_t(Int) } in
  let toi =
    {init with ftype = Int; fname = "toi"; formals = [coord]; body = [] }
  in
  let tol =
    {init with ftype = List_t(Int); fname = "tol"; formals = [x]; body =
   [] }
  in
  let rect =
    { board with gname = "Rect"; extends = Some(board); attributes = [x; y
   ];
      methods = [BasicFunc(init); base_repr "Rect"; BasicFunc(toi);
                 BasicFunc(tol)] }
  in
  let this_dummy =
    { vname = "this"; vtype = Group("Loop", None);
      vinit = None; vloop = false }
  in
  let stmts =
    [Expression(Assign(Attrib(this_dummy, x), (Field(Var(x)), Int)), Int);
```

```
    Expression(Call(Some(this_dummy), BasicFunc(init_cells),
                    [(Field(Var(x)), Int)]), Void);
    Return(Field(This), Group("Loop", None))]
in
let init = base_init "Loop" in
let init = { init with formals = [x]; body = stmts } in
let coord = { x with vname = "coord"; vtype = List_t(Int) } in
let toi =
  {init with ftype = Int; fname = "toi"; formals = [coord]; body = [] }
in
let tol =
  {init with ftype = List_t(Int); fname = "tol"; formals = [x]; body =
 [] }
in
let loop =
  { rect with gname = "Loop"; attributes = [x];
    methods = [BasicFunc(init); base_repr "Loop"; BasicFunc(toi);
 BasicFunc(tol)] }
in
let this_dummy =
  { vname = "this"; vtype = Group("Line", None);
    vinit = None; vloop = false }
in
let init = base_init "Line" in
let stmts =
  [Expression(Assign(Attrib(this_dummy, x), (Field(Var(x)), Int)), Int);
   Expression(Call(Some(this_dummy), BasicFunc(init_cells),
                   [(Field(Var(x)), Int)]), Void);
   Return(Field(This), Group("Line", None))]
in
let init = { init with formals = [x]; body = stmts } in
let coord = { x with vname = "coord"; vtype = List_t(Int) } in
let toi =
  {init with ftype = Int; fname = "toi"; formals = [coord]; body = [] }
in
let tol =
  {init with ftype = List_t(Int); fname = "tol"; formals = [x]; body =
 [] }
in
let line =
  { loop with gname = "Line";
    methods = [BasicFunc(init); base_repr "Loop"; BasicFunc(toi);
 BasicFunc(tol)] }
in
let this_dummy =
  { vname = "this"; vtype = Group("Hex", None);
    vinit = None; vloop = false }
in
let stmts =
  [Expression(Assign(Attrib(this_dummy, x), (Field(Var(x)), Int)), Int);
   Expression(Call(Some(this_dummy), BasicFunc(init_cells),
                   [(Field(Var(x)), Int)]), Void);
   Return(Field(This), Group("Hex", None))]
in
```

```
  let init = base_init "Hex" in
  let init = { init with formals = [x]; body = stmts } in
  let coord = { x with vname = "coord"; vtype = List_t(Int) } in
  let toi =
    {init with ftype = Int; fname = "toi"; formals = [coord]; body = [] }
  in
  let tol =
    {init with ftype = List_t(Int); fname = "tol"; formals = [x]; body =
    [] }
  in
  let hex =
    { loop with gname = "Hex";
      methods = [BasicFunc(init); base_repr "Loop"; BasicFunc(toi);
    BasicFunc(tol)] }
  in
  [rect; loop; line; hex]

let grps =
  [obj; board; piece] @ boards_lib
```

./temp/sen_init_base_grps.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "sen_linked_list.h"

#ifndef SEN_INIT_BASE_GRPS
#define SEN_INIT_BASE_GRPS

bool OCCUPIED = true;
bool NOT_OCCUPIED = false;

struct snt_Board {
    Sen_list cells;
    Sen_list occupied;
};

struct snt_Piece {
    int snt_owner;
    int snt_fixed;
    char *snt_s;
} p;

struct snt_Rect{
    Sen_list cells;
    Sen_list occupied;
    int x;
    int y;
};

struct snt_Line{
    Sen_list cells;
    Sen_list occupied;
    int x;
```

```
};
struct snt_Loop{
    Sen_list cells;
    Sen_list occupied;
    int x;
};
struct snt_Hex{
    Sen_list cells;
    Sen_list occupied;
    int x;
};



void snt_Board_snt_INIT_CELLS(struct snt_Board *b, int n) {
    int i = 0;

    struct snt_Piece *dummy_piece = malloc(sizeof(struct snt_Piece));
    dummy_piece->snt_owner = -1;
    dummy_piece->snt_fixed = 0;
    dummy_piece->snt_s = " ";

    new_Sen_list(&(b->cells), sizeof(struct snt_Piece));
    new_Sen_list(&(b->occupied), sizeof(bool));
    while (i < n) {
        push(&(b->cells), dummy_piece);
        push(&(b->occupied), &NOT_OCCUPIED);
        ++i;
    }
}

int snt_Board_snt_owns(struct snt_Board *b, int i) {
    struct snt_Piece *p = (struct snt_Piece *) list_elem(&(b->cells), i);
    return p->snt_owner;
}

bool snt_Board_snt_full(struct snt_Board *b) {
    int i = 0;
    while (i < b->occupied.len) {
        bool elem = *((bool *) list_elem(&(b->occupied), i));
        if (elem == false) {
            return false;
        }
        ++i;
    }
    return true;
}

bool snt_Board_snt_remove(struct snt_Board *b, int x) {
    bool *elem = (bool *) list_elem(&(b->occupied), x);
    if (*elem) {
        *elem = false;
        struct snt_Piece *dummy_piece = malloc(sizeof(struct snt_Piece));
        dummy_piece->snt_owner = -1;
```

```
        dummy_piece->snt_fixed = 0;
        dummy_piece->snt_s = " ";
        replace(&(b->cells), dummy_piece, x);
        return true;
    } else {
        return false;
    }
}

bool snt_Board_snt_place(struct snt_Board *b, struct snt_Piece *p, int x)
    {
    bool *occ_elem = (bool *) list_elem(&(b->occupied), x);
    if (*occ_elem) {
        return false;
    }
    *occ_elem = true;
    replace(&(b->cells), p, x);
    return true;
}

int snt_Board_snt_toi(struct snt_Board *b, Sen_list *list) {
    return 0; // must be overwritten in child classes
}

int snt_Rect_snt_toi(struct snt_Rect *b, Sen_list *list) {
    // printf("DEBUG: snt_Rect_snt_toi\n");
    // printf("DEBUG: converting: "); printList(list, printInt); printf("\
   n");

    int x = *((int *) list_elem(list, 0));
    int y = *((int *) list_elem(list, 1));

    // printf("DEBUG: x: %d\n", x);
    // printf("DEBUG: y: %d\n", y);
    // printf("DEBUG: b->y: %d\n", b->y);
    // printf("DEBUG: output: %d\n", (b->y) * y + x);

    return (b->y) * y + x;
}

int snt_Line_snt_toi(struct snt_Line *b, Sen_list *list) {
    return *((int *) list_elem(list, 0));
}

int snt_Loop_snt_toi(struct snt_Loop *b, Sen_list *list) {
    int x = *((int *) list_elem(list, 0));
    while (x < 0) {
        x += (b->x);
    }
    return x % (b->x);
}

Sen_list snt_Board_snt_tol(struct snt_Board *b, int i) {
    Sen_list list;
```

```
    new_Sen_list(&list, sizeof(int));
    return list; // must be overwritten in child classes
}

Sen_list snt_Rect_snt_tol(struct snt_Rect *b, int i) {
    int max_x = b->x;
    Sen_list list;
    new_Sen_list(&list, sizeof(int));

    int y = i / 3;
    int x = i - (max_x * y);

    push(&list, &x);
    push(&list, &y);

    return list;
}

Sen_list snt_Line_snt_tol(struct snt_Line *b, int i) {
    Sen_list list;
    new_Sen_list(&list, sizeof(int));
    int x = i;
    push(&list, &x);
    return list;
}

Sen_list snt_Loop_snt_tol(struct snt_Loop *b, int i) {
    Sen_list list;
    new_Sen_list(&list, sizeof(int));
    int x = i;
    push(&list, &x);
    return list;
}

#endif
```

./temp/sen_linked_list.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#ifndef SEN_LINKED_LIST
#define SEN_LINKED_LIST

// Heavily based on
// http://pseudomuto.com/development/2013/05/02/implementing-a-generic-
   linked-list-in-c/

typedef struct Sen_node {
  void *data;
  struct Sen_node *next;
  struct Sen_node *prev;
} Sen_node;

```

```
typedef struct Sen_list {
  int len;
  int data_size;
  Sen_node *head;
  Sen_node *tail;
} Sen_list;


void new_Sen_list(Sen_list *list, int data_size) {
  list->len = 0;
  list->data_size = data_size;
  list->head = list->tail = NULL;
}

/* Function to add a Sen_node at the beginning of Linked List.
   This function expects a pointer to the data to be added
   and size of the data type */
void push(Sen_list *head_ref, void *new_data) {
  // Allocate memory for Sen_node
  Sen_node* new_Sen_node = malloc(sizeof(Sen_node));

  new_Sen_node->data  = malloc(head_ref->data_size);
//  memcpy(new_Sen_node->data, new_data, head_ref->data_size);


  new_Sen_node->next = head_ref->head;
  new_Sen_node->prev = NULL;
  head_ref->head = new_Sen_node;

  if (head_ref->len == 0) {
    head_ref->tail = head_ref->head;
  } else {
    new_Sen_node->next->prev = new_Sen_node;
  }

  // Copy contents of new_data to newly allocated memory.
  // Assumption: char takes 1 byte.
  int i;
  for (i = 0; i < head_ref->data_size; i++) {
    *(char *)(new_Sen_node->data + i) = *(char *)(new_data + i);
  }

  // Change head pointer as new Sen_node is added at the beginning

  head_ref->len++;
}

// Function to print an integer
void printInt(void *n) {
   printf("%d", *(int *)n);
}

void printStr(void *n) {
   printf("%s", *(char **)n);
```

```
}

void printBool(void *b) {
  printf("%s", *(bool *)b ? "true" : "false");
}

void printEmptyList(void *el) {
  printf("[]");
}

void printGroupList(Sen_list *list, char *(*fptr)(void *)) {
  int i = 0;
  Sen_node *n = list->tail;
  printf("[");
    while (i < list->len) {
    printf("%s", (*fptr)(n->data));
    n = n->prev;
    ++i;
    if (i < list->len) {
      printf(", ");
    }
  }
  printf("]");
}

/* Function to print Sen_nodes in a given linked list. fpitr is used
   to access the function to be used for printing current Sen_node data.
   Note that different data types need different specifier in printf() */
void printList(Sen_list *list, void (*fptr)(void *)) {
  int i = 0;
  Sen_node *n = list->tail;
  printf("[");
  while (i < list->len) {
    (*fptr)(n->data);
    n = n->prev;
    ++i;
    if (i < list->len) {
      printf(", ");
    }
  }
  printf("]");
}

void *list_elem(Sen_list *list, int i) {
  int x = 0;
  Sen_node *n = list->tail;
  while (x < list->len && x != i) {
    n = n-> prev;
    ++x;
  }
  return n->data;
}

void replace(Sen_list *list, void *new_data, int i) {
```

```
  Sen_node *new_node = malloc(sizeof(Sen_node));
  new_node->data = malloc(list->data_size);
  memcpy(new_node->data, new_data, list->data_size);

  int x = 0;
  Sen_node *tmp;
  Sen_node *n = list->tail;
  while (x < list->len) {
    if (x == i - 1) {
      tmp = n->prev;
      n->prev = new_node;
      n = tmp;
    } else if (x == i) {
      new_node->prev = n->prev;
      new_node->next = n->next;
      n = n->prev;
    } else if (x == i + 1) {
      tmp = n->prev;
      n->next = new_node;
      n = tmp;
    } else {
      n = n->prev;
    }
    ++x;
  }

  if (list->len > 0) {
    if (i == 0) {
      list->tail = new_node;
    }
    if (i == list->len - 1) {
      list->head = new_node;
    }
  }
}

#endif
```

./temp/sen_print_base_grps.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "sen_init_base_grps.h"

#ifndef SEN_PRINT_BASE_GRPS
#define SEN_PRINT_BASE_GRPS

char*  snt_Piece_snt___repr__(struct snt_Piece *this) {
  return this->snt_s;
}

char*  snt_Board_snt___repr__(struct snt_Board *this) {
  return "<Group Board instance>";
}
```

```

char*  snt_Line_snt___repr__(struct snt_Line *this) {
  return "<Group Line instance>";
}

char*  snt_Loop_snt___repr__(struct snt_Loop *this) {
  return "<Group Loop instance>";
}

char*  snt_Hex_snt___repr__(struct snt_Hex *this) {
  return "<Group Hex instance>";
}

char*  snt_Rect_snt___repr__(struct snt_Rect *this) {
  return "<Group Rect instance>";
}

#endif
```

./temp/sen_read.h

```
#include <stdio.h>
#include <stdlib.h>

#ifndef SEN_READ
#define SEN_READ

char SENET_BUFF[128];

char *_snt_read(int x) {
  // http://stackoverflow.com/questions/4404368/using-scanf-to-read-in-certain-amount-of-characters-in-c
  char *s = malloc(sizeof(char) * 128);
  sprintf(SENET_BUFF, "%%%dc", x);
  scanf(SENET_BUFF, s);
  return s;
}

void _snt_clear_input() {
    int c;
    while ( (c = getchar()) != '\n' && c != EOF ) ;
}

#endif
```

./testall.sh

```
#!/bin/sh

SENET="./senet"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
```

```
rm -f output.c
rm -f a.out
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.snt files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
  echo "FAILED"
  error=1
    fi
    echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to
   difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
  SignalError "$1 differs"
  echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
  SignalError "$1 failed on $*"
  return 1
    }
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                              s/.snt//'`
    reffile=`echo $1 | sed 's/.snt$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
```

```
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.c.out output.c a.out" &&
    Run "$SENET" "-c" "<" $1 &&
    Run "gcc output.c" &&
    Run "./a.out >" ${basename}.c.out &&
    Compare ${basename}.c.out ${reffile}.out ${basename}.c.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
  if [ $keep -eq 0 ] ; then
      rm -f $generatedfiles
  fi
  echo "OK"
  echo "###### SUCCESS" 1>&2
    else
  echo "###### FAILED" 1>&2
  globalerror=$error
    fi
}

while getopts kdpsh c; do
    case $c in
  k) # Keep intermediate files
      keep=1
      ;;
  h) # Help
      Usage
      ;;
    esac
done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/fail-*.snt tests/test-*.snt"
fi

for file in $files
do
    case $file in
  *test-*)
      Check $file 2>> $globallog
      ;;
  *fail-*)
      CheckFail $file 2>> $globallog
      ;;
  *)
```

```
      echo "unknown file type $file"
      globalerror=1
      ;;
    esac
done

exit $globalerror
```

./tests/test-add.out

```
5
```

./tests/test-add.snt

```
@setup { }

@turns {

func void begin() {
    print(2 + 3);
    end;
}

}
```

./tests/test-and.out

```
true
false
false
false
```

./tests/test-and.snt

```
@setup { }

@turns {

func void begin() {
    print(True and True); print("\n");
    print(False and False); print("\n");
    print(True and False); print("\n");
    print(False and True); print("\n");
    end;
}

}
```

./tests/test-assert-func.out

```
true
true
false
false
```

./tests/test-assert-func.snt

```
@setup {

```

```
assert test() {
    True;
}

assert test2() {
    2 + 2 > 3;
    4 - 2 < 0;
    2 == 2;
}

assert test3(int x) {
    if (x > 3) { True; } else { False; }
}

}

@turns {

func void begin() {
    print(True); print("\n");
    print(test()); print("\n");
    print(test2()); print("\n");
    print(test3(2)); print("\n");
    end;
}

}
```

./tests/test-assign.out

```
2
```

./tests/test-assign.snt

```
@setup { }

@turns {

func void begin() {
    int x = 3;
    x = 2;
    print(x); print("\n");
    end;
}

}
```

./tests/test-basic-func-call.out

```
5
6
```

./tests/test-basic-func-call.snt

```
@setup {

func int test(int x) {
```

```
    return x + 1;
}

}

@turns {

func void begin() {
    int a = 5;
    print(a); print("\n");
    a = test(a);
    print(a); print("\n");
    end;
}

}
```

./tests/test-board-cells.out

```
[ ,  ,  ,  ,  ,  ,  ,  ,  ]

false
-1
false
```

./tests/test-board-cells.snt

```
@setup {

group table(Rect(3, 3)) { };

}

@turns {

func void begin() {
    group table a = table();

    print(a.cells); print("\n");
    print(a.cells[2]); print("\n");
    print(a.occupied[5]); print("\n");
    print(a.owns(2)); print("\n");
    print(a.full()); print("\n");
    a.remove(6);

    end;
}

}
```

./tests/test-board-place.out

```
M
did not remove
```

./tests/test-board-place.snt

```
@setup {

group table(Rect(3, 3)) { };
group mark(Piece) {
    func group mark __init__(str s) { this.s = s; return this; }
};

}

@turns {

func void begin() {
    group table t = table();
    group mark m = mark("M");
    group mark x = mark("X");

    t.place(m, 3);
    t.place(x, 3);  # doesn't place X since it is occupied
    print(t.cells[3]); print("\n");

    if (t.remove(2)) {
        print("remove error\n");
    } else {
        print("did not remove\n");
    }

    end;
}

}
```

./tests/test-board-toi.out

```
0
1
2
3
4
5
6
7
8
4
0
1
```

./tests/test-board-toi.snt

```
@setup {

group table(Rect(3, 3)) { };

group myloop(Loop(5)) { };

}

```

```
@turns {

func void begin() {
    group table a = table();
    group myloop b = myloop();
    int x = -1;
    list[int] l = [x];

    print(a.toi([0, 0])); print("\n");
    print(a.toi([1, 0])); print("\n");
    print(a.toi([2, 0])); print("\n");
    print(a.toi([0, 1])); print("\n");
    print(a.toi([1, 1])); print("\n");
    print(a.toi([2, 1])); print("\n");
    print(a.toi([0, 2])); print("\n");
    print(a.toi([1, 2])); print("\n");
    print(a.toi([2, 2])); print("\n");


    print(b.toi([x])); print("\n");
    print(b.toi([0])); print("\n");
    print(b.toi([1])); print("\n");

    end;
}

}
```

./tests/test-board-tol.out

```
[0, 0]
[1, 0]
[2, 0]
[0, 1]
[1, 1]
[2, 1]
[0, 2]
[1, 2]
[2, 2]
```

./tests/test-board-tol.snt

```
@setup {

group table(Rect(3, 3)) { };

}

@turns {

func void begin() {
    group table a = table();

    print(a.tol(0)); print("\n");
    print(a.tol(1)); print("\n");
    print(a.tol(2)); print("\n");
```

```
    print(a.tol(3)); print("\n");
    print(a.tol(4)); print("\n");
    print(a.tol(5)); print("\n");
    print(a.tol(6)); print("\n");
    print(a.tol(7)); print("\n");
    print(a.tol(8)); print("\n");

    end;
}

}
```

./tests/test-board.out

```
3 3
5
10
20
```

./tests/test-board.snt

```
@setup {

group rect(Rect(3, 3)) { };
group line(Line(5)) { };
group loop(Loop(10)) { };
group hex(Hex(20)) { };

group rect r;
group line li;
group loop lo;
group hex h;

}

@turns {

func void begin() {
    r = rect();
    li = line();
    lo = loop();
    h = hex();

    print(r.x); print(" "); print(r.y); print("\n");
    print(li.x); print("\n");
    print(lo.x); print("\n");
    print(h.x); print("\n");
    end;
}

}
```

./tests/test-bool-literal.out

```
true
false
```

./tests/test-bool-literal.snt

```
@setup { }

@turns {

func void begin() {
    print(True);
    print("\n");
    print(False);
    end;
}

}
```

./tests/test-break.out

```
5
```

./tests/test-break.snt

```
@setup { }

@turns {

func void begin() {
    int a = 5;
    while (a > 0) {
        print(a); print("\n");
        a = a - 1;
        break;
    }
    end;
}

}
```

./tests/test-continue.out

```
4
2
```

./tests/test-continue.snt

```
@setup { }

@turns {

func void begin() {
    int a = 5;
    while (a > 0) {
        if (a % 2 == 1) {
            a = a - 1;
            continue;
        }
        print(a); print("\n");
        a = a - 1;
    }
```

```
    end;
}

}
```

./tests/test-div.out

```
5
4
```

./tests/test-div.snt

```
@setup { }

@turns {

func void begin() {
    print(15 / 3); print("\n");
    print(14 / 3); print("\n");
    end;
}

}
```

./tests/test-element.out

```
1
400
<Group A instance>
400
```

./tests/test-element.snt

```
@setup {

group A(Object) {
    int x;
    func group A __init__(int x) {
        this.x = x;
        return this;
    }
};

}

@turns {

func void begin() {
    list[int] x = [0, 1, 2, 3, 4, 5];

    print(x[1]); print("\n");

    pass(test_groups, 0);
}

func void test_groups() {
    group A obj1 = A(100);
```

```
    group A obj2 = A(200);
    group A obj3 = A(300);
    group A obj4 = A(400);
    list[group A] x = [obj1, obj2, obj3, obj4];
    group A y;

    print(obj4.x); print("\n");
    print(x[3]); print("\n");

    y = x[3];
    print(y.x); print("\n");

    end;
}

}
```

./tests/test-equal.out

```
false
true
true
false
true
```

./tests/test-equal.snt

```
@setup {

group base(Object) {
    int x;
    int y;
    func group base __init__(int x, int y) {
        this.x = x;
        this.y = y;
        return this;
    }
};

group A(base) {
    int x;
    int y;
};

group B(base) {
    int x;
    int y;
};

group C(base) {
    int a;
    int b;
};

group A a;
group A aa;
```

```
group B b;
group C c;

}

@turns {

func void begin() {
    print(2 == 3); print("\n");
    print(2 == 2); print("\n");
    print(2 == 2 == True); print("\n");
    print("abc" == "def"); print("\n");

    a.__init__(100, 200);
    aa.__init__(100, 200);
    print(a == aa); print("\n");
    end;
}

}
```

./tests/test-exit.out

```
test
```

./tests/test-exit.snt

```
@setup { }

@turns {

func void begin() {
    print("test\n");
    exit();
    print("did not exit\n");
    end;
}

}
```

./tests/test-for.out

```
1
2
3
abc
def
ghi
jkl
2 3
4 7
6 11
1
[1, 2, 3]
4
[4, 5]
6
```

```
[6, 7]
```

./tests/test-for.snt

```
@setup {

group A(Object) {
    int x;
    int y;
    func group A __init__(int x, int y) {
        this.x = x;
        this.y = y;
        return this;
    }

    func int test() {
        return this.x + this.y;
    }
};

}

@turns {

func void begin() {
    for (int x in {1, 2, 3}) {
        print(x); print("\n");
    }
    pass(test_str, 0);
}

func void test_str() {
    for (str s in {"abc", "def", "ghi", "jkl"}) {
        print(s); print("\n");
    }
    pass(test_grp, 0);
}

func void test_grp() {
    group A a1 = A(1, 2);
    group A a2 = A(3, 4);
    group A a3 = A(5, 6);
    for (group A x in {a1, a2, a3}) {
        print(x.y); print(" ");
        print(x.test()); print("\n");
    }
    pass(test_list, 0);
}

func void test_list() {
    list[int] a = [1, 2, 3];
    list[int] b = [4, 5];
    list[int] c = [6, 7];
    int y;
    for (list[int] x in {a, b, c}) {
```

```
        y = x[0];
        print(y); print("\n");
        print(x); print("\n");
    }
    end;
}

}
```

./tests/test-geq.out

```
false
true
true
```

./tests/test-geq.snt

```
@setup { }

@turns {

func void begin() {
    print(2 >= 3); print("\n");
    print(2 >= 2); print("\n");
    print(2 >= 1); print("\n");
    end;
}

}
```

./tests/test-greater.out

```
false
false
true
```

./tests/test-greater.snt

```
@setup { }

@turns {

func void begin() {
    print(2 > 3); print("\n");
    print(2 > 2); print("\n");
    print(2 > 1); print("\n");
    end;
}

}
```

./tests/test-group-cast.out

```
2
3
```

./tests/test-group-cast.snt

```
@setup {

group A(Object) {
    int x;
    int y;
    func group A __init__(int x, int y) {
        this.x = x; this.y = y;
        return this;
    }

    func int test() {
        return this.x + this.y;
    }
};

group B(A) {
    int z;
    func group B __init__(int x, int y, int z) {
        this.x = x; this.y = y; this.z = z;
        return this;
    }

    func int test() {
        return this.x + this.y + this.z;
    }

};

}

@turns {

func void begin() {
    group A a = B(1, 2, 3);
    # group B b = A(1, 2);

    print(a.y); print("\n");
    print(a.test()); print("\n");
    # print(b.test()); print("\n");
    end;
}

}
```

./tests/test-group.out

```
2
3
```

./tests/test-group.snt

```
@setup {

group A(Object) {
    int x;

```

```
    func group A __init__() {
        this.x = 2;
        return this;
    }
};

group A obj;

}

@turns {

func void begin() {
    # group A obj;
    obj.__init__();
    print(obj.x); print("\n");
    obj.x = obj.x + 1;
    print(obj.x); print("\n");
    end;
}

}
```

./tests/test-hello.out

```
Hello World
```

./tests/test-hello.snt

```
@setup { }
@turns { func void begin() { print("Hello World\n"); end;} }
```

./tests/test-if.out

```
if ok
if else ok
if elif ok
```

./tests/test-if.snt

```
@setup {

}

@turns {

func void begin() {
    if (5 > 0) {
        print("if ok"); print("\n");
    }
    if (2 < 0) {
        print("if error"); print("\n");
    } else {
        print("if else ok"); print("\n");
    }
    if (2 < 0) {
        print("if error"); print("\n");
```

```
    } elif (3 > 0) {
        print("if elif ok"); print("\n");
    }

    end;
}

}
```

./tests/test-inherit-func-override.out

```
1
2
3
20
30
6
10
8
10
10
10
true
```

./tests/test-inherit-func-override.snt

```
@setup {

group A(Object) {
    int x;

    func group A __init__(int x) {
        this.x = x;
        return this;
    }

    func int test() {
        return this.x + 5;
    }

};


group B(A) {
    int y;

    func int test() {
        this.y = 10;
        return this.y;
    }
};

group C(A) {

};

```

```
group D(B) {
    func group D __init__(int x, int y) {
        this.x = x;
        this.y = y;
        return this;
    }
};

group E(D) {
    func bool test(int x) {
        this.x = x;
        if (this.x > 0) {
            return (True);
        } else {
            return (False);
        }
    }
};

group A obj;
group B obj2;
group C obj3;
group D obj4;
group E obj5;

}

@turns {

func void begin() {
    int y;

    obj.__init__(1);
    obj2.__init__(2);
    obj3.__init__(3);
    obj4.__init__(20, 30);
    obj5.__init__(0, 1);

    print(obj.x); print("\n");
    print(obj2.x); print("\n");
    print(obj3.x); print("\n");
    print(obj4.x); print("\n");
    print(obj4.y); print("\n");

    print(obj.test()); print("\n");
    print(obj2.test()); print("\n");
    print(obj3.test()); print("\n");
    print(obj4.test()); print("\n");

    print(obj4.y); print("\n");

    print(obj5.test()); print("\n");
    print(obj5.test(5)); print("\n");

```

```
85
86      end;
87  }
88
89  }
```

./tests/test-inherit-func.out

```
1
2  10
3  21
4  2
5  3
6  15
7  22
```

./tests/test-inherit-func.snt

```
1  @setup {
2
3  group A(Object) {
4      int x;
5
6      func group A __init__() {
7          this.x = 1;
8          return this;
9      }
10
11     func int testA(int x) {
12         return x + 5;
13     }
14
15     func int testB() {
16         return this.x + 20;
17     }
18 };
19
20
21 group B(A) {
22     int y;
23     int x;
24
25     func group B __init__(int a, int b) {
26         this.x = a;
27         this.y = b;
28         return this;
29     }
30 };
31
32
33 group A obj;
34 group B obj2;
35
36 }
37
38 @turns {
```

```

func void begin() {
    int y;

    obj.__init__();
    obj2.__init__(2, 3);

    print(obj.x); print("\n");
    print(obj.testA(5)); print("\n");
    print(obj.testB()); print("\n");

    print(obj2.x); print("\n");
    print(obj2.y); print("\n");
    print(obj2.testA(10)); print("\n");
    print(obj2.testB()); print("\n");

    end;
}

}
```

./tests/test-inherit-three-deg.out

```
1
2
3
6
7
8
```

./tests/test-inherit-three-deg.snt

```
@setup {

group A(Object) {
    int x;

    func group A __init__(int x) {
        this.x = x;
        return this;
    }

    func int test() {
        return this.x + 5;
    }

};


group B(A) {
    int y;
};

group C(B) {
    int z;
};
```

```

group A obj;
group B obj2;
group C obj3;

}

@turns {

func void begin() {
    int y;

    obj.__init__(1);
    obj2.__init__(2);
    obj3.__init__(3);

    print(obj.x); print("\n");
    print(obj2.x); print("\n");
    print(obj3.x); print("\n");

    print(obj.test()); print("\n");
    print(obj2.test()); print("\n");
    print(obj3.test()); print("\n");

    end;
}

}
```

./tests/test-inherit.out

```
2
3
1
1
2
1
2
5
6
```

./tests/test-inherit.snt

```
@setup {

group A(Object) {
    int x;

    func group A __init__() {
        this.x = 1;
        return this;
    }
};


group B(A) {
```

```
    int y;

    func group B __init__(int a, int b) {
        this.x = a;
        this.y = b;
        return this;
    }
};

group C(A()) {
    int a;
    int b;
};

group D(B(1, 2)) {
    int a;
};

group B obj;
group C obj2;
group D obj3;
group D obj4;
group B obj5;

}

@turns {

func void begin() {
    obj.__init__(2, 3);
    obj2.__init__();
    obj3.__init__();
    obj4.__init__();
    obj5.__init__(5, 6);
    print(obj.x); print("\n");
    print(obj.y); print("\n");
    print(obj2.x); print("\n");
    print(obj3.x); print("\n");
    print(obj3.y); print("\n");
    print(obj4.x); print("\n");
    print(obj4.y); print("\n");
    print(obj5.x); print("\n");
    print(obj5.y); print("\n");
    end;
}

}
```

./tests/test-init.out

```
5
hello
world
world
!!!
```

./tests/test-init.snt

```
@setup {

group A(Object) {
    str x;

    func group A __init__(str x) {
        this.x = x;
        return this;
    }

};

}

@turns {

func void begin() {
    int a = 2 + 3;
    int b = a = 5;
    group A g1 = A("hello");
    group A g2;
    group A g3 = g2 = A("world");

    print(a); print("\n");
    print(g1.x + "\n");
    print(g2.x + "\n");
    print(g3.x + "\n");
    g2.x = "!!!";
    print(g2.x + "\n");


    end;
}

}
```

./tests/test-int-literal.out

```
3
```

./tests/test-int-literal.snt

```
@setup { }

@turns {

func void begin() {
    print(3);
    end;
}

}
```

./tests/test-leq.out

```
true
true
false
```

./tests/test-leq.snt

```
@setup { }

@turns {

func void begin() {
    print(2 <= 3); print("\n");
    print(2 <= 2); print("\n");
    print(2 <= 1); print("\n");
    end;
}

}
```

./tests/test-less.out

```
true
false
false
```

./tests/test-less.snt

```
@setup { }

@turns {

func void begin() {
    print(2 < 3); print("\n");
    print(2 < 2); print("\n");
    print(2 < 1); print("\n");
    end;
}

}
```

./tests/test-list-literal-emptylist.out

```
[]
[]
```

./tests/test-list-literal-emptylist.snt

```
@setup {

}

@turns {

func void begin() {
    list[void] x = [];

    print([]); print("\n");
    print(x); print("\n");
```

```
12
13      end;
14  }
15
16  }
```

./tests/test-list-literal.out

```
1  []
2  [1, 2, 3]
3  [3, 4, 5]
4  [abc, def]
5  [hello world!, 123]
6  [true, false, true]
7  [1, 2, 3]
8  [x, y, hello world!]
9  [obj1, obj2, obj3]
10 [obj1, obj2, obj3]
```

./tests/test-list-literal.snt

```
1  @setup {
2
3  group A(Object) {
4      str s;
5      func group A __init__(str s) {
6          this.s = s;
7          return this;
8      }
9      func str __repr__() {
10         return this.s;
11     }
12 };
13
14 }
15
16 @turns {
17
18 func void begin() {
19     int x = 1; int y = 2; int z = 3;
20     str s = "hello world!";
21     list[int] a = [1, 2, 3];
22     list[str] b = ["x", "y", s];
23
24     group A obj1 = A("obj1");
25     group A obj2 = A("obj2");
26     group A obj3 = A("obj3");
27     list[group A] g = [obj1, obj2, obj3];
28
29     print([]); print("\n");
30     print([x, y, z]); print("\n");
31     print([3, 4, 5]); print("\n");
32     print(["abc", "def"]); print("\n");
33     print([s, "123"]); print("\n");
34     print([True, False, True]); print("\n");
35     print(a); print("\n");
```

```
    print(b); print("\n");

    print([obj1, obj2, obj3]); print("\n");
    print(g); print("\n");


    end;
}

}
```

./tests/test-min-not.out

```
5
false
-3
```

./tests/test-min-not.snt

```
@setup { }

@turns {

func void begin() {
    print(5); print("\n");
    print(not True); print("\n");
    print(-3); print("\n");
    end;
}

}
```

./tests/test-mod.out

```
1
2
0
```

./tests/test-mod.snt

```
@setup { }

@turns {

func void begin() {
    print(5 % 2); print("\n");
    print(5 % 3); print("\n");
    print(5 % 5); print("\n");
    end;
}

}
```

./tests/test-mult-turn.out

```
true
false
```

./tests/test-mult-turn.snt

```
@setup { }

@turns {

func void begin() {
    print(True); print("\n");
    pass(ending, 2);
}

func void ending() {
     print(False); print("\n");
     end;
}

}
```

./tests/test-mult.out

```
-15
```

./tests/test-mult.snt

```
@setup { }

@turns {

func void begin() {
    print(5 * -3);
    end;
}

}
```

./tests/test-neq.out

```
true
false
false
true
true
false
false
```

./tests/test-neq.snt

```
@setup {

group A(Object) {
    int x;
    int y;

    func group A __init__(int x, int y) {
        this.x = x;
        this.y = y;
        return this;
    }
};
```

```

group B(A(2, 3)) {

};

group A a;
group A aa;
group B b;
group B bb;

}

@turns {

func void begin() {
    print(2 != 3); print("\n");
    print(2 != 2); print("\n");
    print("abc" != "abc"); print("\n");
    print("abcd" != "abc"); print("\n");

    a.__init__(100, 200);
    aa.__init__(200, 300);
    b.__init__();
    bb.__init__();

    print(a != aa); print("\n");
    print(b != b); print("\n");
    print(b != bb); print("\n");
    end;
}

}
```

./tests/test-none.out

```
None
true
```

./tests/test-none.snt

```
@setup {

group A(Object) {

    func group A __init__() { return this; }

};

group A test;

}

@turns {

func void begin() {

```

```
    print(None); print("\n");
    print(None == None); print("\n");

    end;
}

}
```

./tests/test-or.out

```
true
false
true
true
```

./tests/test-or.snt

```
@setup { }

@turns {

func void begin() {
    print(True or True); print("\n");
    print(False or False); print("\n");
    print(True or False); print("\n");
    print(False or True); print("\n");
    end;
}

}
```

./tests/test-pass-group.out

```
X
false
```

./tests/test-pass-group.snt

```
@setup {

group Mark(Piece) {
    func group Mark __init__(str s) {
        this.s = s;
        return this;
    }
};

func str test(group Mark m) {
    return m.s;
}

assert check(group Mark m) {
    m.s == "M";
}

}

```

```
@turns {

func void begin() {
    group Mark x = Mark("X");
    print(test(x)); print("\n");
    print(check(x)); print("\n");
    end;
}

}
```

./tests/test-pass-list.out

```
3
true
```

./tests/test-pass-list.snt

```
@setup {

list[int] mylist;

func int test(list[int] x) {
    return x[2];
}

assert check(list[int] x) {
    x[0] == 1;
}

}

@turns {

func void begin() {
    mylist = [1, 2, 3];
    print(test(mylist)); print("\n");
    print(check(mylist)); print("\n");
    end;
}

}
```

./tests/test-piece.out

```
x
o
```

./tests/test-piece.snt

```
@setup {

group mark(Piece) {
    str symbol;
    func group mark __init__(str s) {
        this.fixed = True;
        this.symbol = s;
```

```
        return this;
    }

    func str __repr__() {
        return this.symbol;
    }
};

group mark x;
group mark o;

}

@turns {

func void begin() {
    x = mark("x");
    o = mark("o");

    print(x); print("\n");
    print(o); print("\n");

    end;
}

}
```

./tests/test-place.out

```
true
false
```

./tests/test-place.snt

```
@setup {

group B(Rect(2, 2)) { };
group Mark(Piece) {
    func group Mark __init__(str s) {
        this.s = s;
        return this;
    }
};

group B brd;




}

@turns {

func void begin() {
    group Mark m = Mark("M");
    brd = B();

```

```
    print(m >> brd >> [1, 1]); print("\n");
    print(m >> brd >> [1, 1]); print("\n");

    end;
}

}
```

./tests/test-print.out

```
1
abc
true false
<Group A instance>
<Group A instance>
Hello from class B!
Hello from class B!
<Group D instance>
```

./tests/test-print.snt

```
@setup {

group A(Object) {
    func group A __init__() { return this; }
};

group B(A) {
    func str __repr__() {
        return "Hello from class B!";
    }
};

group C(B) {

};

group D(A) {

};

group A no_repr;
group B yes_repr;
group C inherited_repr;
group D inherited_repr_built_in;

}

@turns {

func void begin() {

    print(1, "\n");
    print("abc\n");
    print(True, " ", False, "\n");
    print(no_repr, "\n");
```

```
    print(no_repr.__repr__(), "\n");
    print(yes_repr, "\n");
    print(inherited_repr, "\n");
    print(inherited_repr_built_in, "\n");

    end;
}

}
```

./tests/test-rand.out

```
success calling rand
```

./tests/test-rand.snt

```
@setup { }

@turns {

func void begin() {
    int x = rand();
    print("success calling rand\n");
    end;
}

}
```

./tests/test-remove.out

```
false
true
```

./tests/test-remove.snt

```
@setup {

group B(Rect(2, 2)) { };
group Mark(Piece) {
    func group Mark __init__(str s) {
        this.s = s;
        return this;
    }
};

group B brd;




}

@turns {

func void begin() {
    group Mark m = Mark("M");
    brd = B();

```

```
    brd.place(m, brd.toi([0, 1]));

    # print(brd.toi([1, 1])); print("\n");
    print(brd << [1, 1]); print("\n");
    print(brd << [0, 1]); print("\n");

    end;
}

}
```

./tests/test-scope.out

```
6
25
```

./tests/test-scope.snt

```
@setup {
    int x = 4;

    group A(Object) {
        int x;

        func group A __init__(int x) {
            this.x = x;
            return this;
        }

        func void y() {

        }
    };

}

@turns {

func void begin() {
    int x = 6;
    group A a = A(25);

    print(x); print("\n");
    print(a.x); print("\n");

    end;
}

}
```

./tests/test-stoi.out

```
12345
```

./tests/test-stoi.snt

```
@setup { }
```

```

@turns {

func void begin() {
    int x = stoi("12345");

    print(x); print("\n");

    end;
}

}
```

./tests/test-str-lit.out

```
abcdef
```

./tests/test-str-lit.snt

```
@setup { }

@turns {

func void begin() {
    str s = "abcdef";
    print(s); print("\n");
    end;
}

}
```

./tests/test-strcat.out

```
abcdef
```

./tests/test-strcat.snt

```
@setup { }

@turns {

func void begin() {
    str a = "abc";
    str b = "def";

    print(a + b); print("\n");

    end;
}

}
```

./tests/test-sub.out

```
-1
```

./tests/test-sub.snt

```
@setup { }

@turns {

func void begin() {
    print(2 - 3);
    end;
}

}
```

./tests/test-turn-end.out

```
passing
```

./tests/test-turn-end.snt

```
@setup { int x = 0; }

@turns {

func void begin() {
    if (PLAYER_ON_MOVE == 1) {
        end;
    } else {
        print("passing\n");
        pass(ending, 1);
    }
}

func void ending() {
    pass (begin, 1);
}

}
```

./tests/test-while.out

```
5
4
3
2
1
```

./tests/test-while.snt

```
@setup { }

@turns {

func void begin() {
    int a = 5;
    while (a > 0) {
        print(a); print("\n");
        a = a - 1;
    }
    end;
```

```
}

}
```

./types.ml

```
type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater |
    Geq
          | Mod | And | Or

type bool_lit =
    True
  | False

type t =
    Int
  | Bool
  | Str
  | Void
  | List_t of t
  | Group of string * group_decl option  (* 2nd value filled in by Cast *)

and list_lit =
    Elems of expression list * string
  | EmptyList

and var_decl = {
    vname : string;
    vtype : t;
    vinit : expression option;
    vloop : bool
}

and expr_detail =
    IntLiteral of int * string
  | StrLiteral of string * string
  | ListLiteral of list_lit
  | BoolLiteral of bool_lit * string
  | VoidLiteral
  | Field of field_expr
  | Binop of expression * op * expression
  | Assign of field_expr * expression
  | Call of var_decl option * func_decl * expression list
  | Element of expression * expression
  | Uminus of expression
  | Not of expression
  | Noexpr
  | Remove of expression
  | Place of expression

and expression =
    expr_detail * t

and field_expr =
    Var of var_decl
```

```
  | Attrib of var_decl * var_decl
  | Fun of func_decl
  | Method of var_decl * func_decl
  | Grp of group_decl
  | This

and statement =
    Block of symbol_table * statement list
  | Expression of expression
  | Return of expression
  | Break
  | Continue
  | End
  | Pass of func_decl * expression
  | If of expression * statement * expression option * statement
  | For of var_decl * expression list * statement
  | While of expression * statement

and basic_func_decl = {
    ftype : t;
    fname : string;
    formals : var_decl list;
    locals : var_decl list;
    body : statement list;
    turns_func : bool;
    group_method : string;
    f_is_built_in : bool
  }

and assert_decl = {
    aname : string;
    aformals : var_decl list;
    alocals : var_decl list;
    abody : statement list;
    a_turns_func : bool;
    a_group_method : string;
    a_is_built_in : bool
  }

and func_decl =
    BasicFunc of basic_func_decl
  | AssertFunc of assert_decl

and group_decl = {
    gname : string;
    extends : group_decl option;
    par_actuals : expression list option;
    attributes : var_decl list;
    methods : func_decl list;
  }

and setup = var_decl list * func_decl list * group_decl list

and turns = func_decl list
```

```
103
104 and program = setup * turns
105
106
107 and symbol_table = {
108   parent : symbol_table option;
109   mutable variables : var_decl list;
110   mutable functions : func_decl list;
111   mutable groups : group_decl list;
112   mutable turns : string list;
113   mutable ll_count : int;
114   mutable elem_count : int
115 }
116
117 type partial_group_table = {
118   group_name : string;
119   par : group_decl option;
120   symbols : symbol_table;
121 }
122
123 type translation_environment = {
124   scope : symbol_table;
125   return_type : t;
126   in_loop : bool;
127   partial_group_info : partial_group_table option
128 }
```