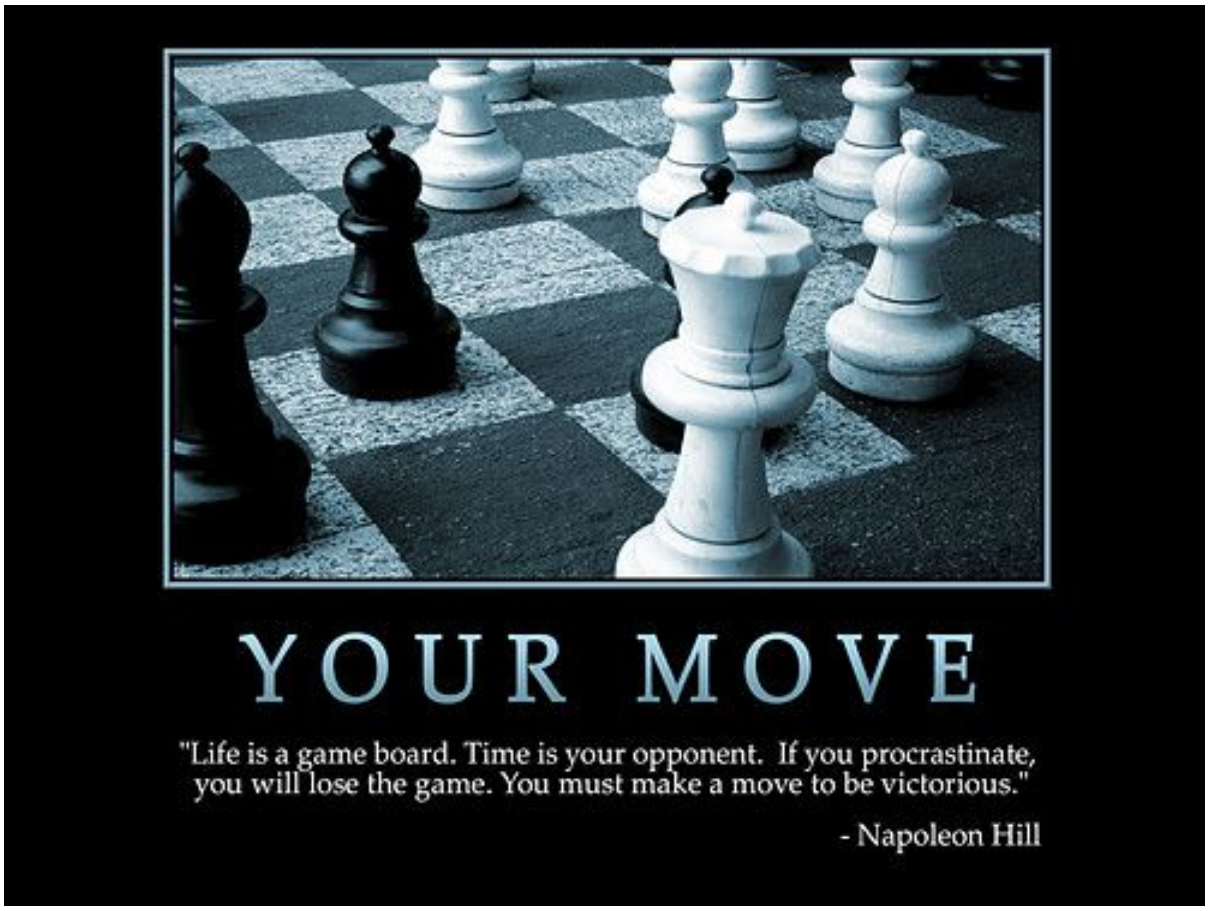
GRIDLang

Move-Driven Game Programming Language

Final Report

Akshay Nagpal (an2756) Dhruv Shekhawat (ds3512)
Parth Panchmatia (psp2137) Sagar Damani (ssd2144)

May 10, 2017



Contents

1 Introduction

GRIDLang is a language to design games in an intuitive and expressive manner. It enables developers to quickly prototype grid-based games and get a programmatic view of it. It simplifies the process of defining rules for a game, creating a grid and manipulating them. There are in-built language components focused on game development that enables developers to express more with less lines of code.

GRIDLang is a move-driven language. The control flow of the program depends on the movement of pieces on a grid structure. This allows for template functions that will be triggered based on pre-specified events, so that the programmer can focus on the game logic without having to worry about controlling the overall flow.

2 Language Tutorial

2.1 Using the Compiler

Inside the GRIDLang directory, type **make**. This creates grid.native, which generates the LLVM IR when given a file in GRIDLang. A file can be run by using **./gridrun.sh -r filename.grid**

2.2 Sample Programs

A simple helloworld.grid program in our language can be written as follows:

```
int setup() {
  print("Hello World");
  return 0;
}
```

- *setup()* is the mandatory function from where the execution of the program begins. It returns an int value.
- *print()* is a built-in function that can be used to print *string*s, *int*s and *bool*s.

Below is another sample program that demonstrates some additional features. It will print the string z five times.

```
int checkGameEnd() {
    if (count > 0) {
        count = count -1;
        return 0;
    }
    return 1;
}

int setup() {
  return 0;
}

int gameloop() {
    string z;
    z = "Write this 5 times";
    print(z);
    return 0;
}
```

- The function *gameloop()* is implicitly called from *setup()*.
- *gameloop()* runs in a loop until the function *checkGameEnd()* returns 1
- Once *checkGameEnd()* returns 1, the program execution ends.

This final sample demonstrates the non-primitive data types *Player*, *Piece* and *Grid*.

```
import gridBasics.grid

int checkGameEnd() {
  return 1;
}

Piece Token {
  string color;
}

Player {
  Piece Token t;
}

Grid_Init<5,4>;
Player p1,p2;
Player[2] playerOrder;

int setup() {
  playerOrder[0] = p1;
  playerOrder[1] = p2;
  playerOrderSize = 2;
  p1.t.color = "White";
  p2.t.color = "Black";
  p1.t.displayString = "Token1";
  p2.t.displayString = "Token2";
  Grid <1,2> <-- p1.t;
  Grid <3,2> <-- p2.t;
  printGrid();
  return 0;
}
```

- *gridBasics.grid* is the library that must be included for any grid-related functionality.
- *Piece* is a structure that can be placed on the grid. It has the default property *displayString* (the string that represents it on the grid).
- A *Player* is a structure that can hold *Piece*s and other properties.
- *Grid_Init<x,y>* creates a grid with x rows and y columns, on which Pieces can be placed.
- *Grid <x,y> < -- p1.t* adds the *Piece p1.t* to the location ¡x,y¿ on the grid
- *printGrid()* is a library function that prints the grid.

3 Language Reference Manual

3.1 Lexical Elements

3.1.1 Identifiers

Identifiers are strings for naming variables and functions. The identifiers are case-sensitive. They can consist of letters, digits and underscores, but must always start with a letter.

Foo foo Bar BAR

3.1.2 Keywords

Keywords are reserved for use in the language and cannot be used as identifiers. These keywords are case sensitive.

int	string	Player	Piece	Grid
Grid_Init	currentPlayerIndex	colocation	gameLoop	while
return	rule	return	if	abs
for	diceThrow	bool	checkGameEnd	playerOrder
print_endline	print_sameline	print	prompt	getLen
print_int_sameline	rows	cols	triggerRule	playerOrderSize

3.1.3 Literals

Literals are constant string, numeric, array or boolean values, such as “abc”, 25, [1,2,3] or false. Each literal is immutable and has a specific data type corresponding to one of the primitive types. No type casting is allowed. Trying to assign a literal to a variable of mismatching type will cause an error.

```
string x;
x = "abc";
```

String Literals String literals are a sequence of zero or more non-double-quote characters, enclosed in double quotes.

Integer Literals Integer literals are whole numbers represented by a sequence of one or more digits from 0-9. Integers are assumed to be in decimal (base 10) format. Precede an integer with ”-” to denote a negative number. Eg:

31 & -343

Boolean Literals A boolean literal represents a truth value and can have the value true or false (case sensitive).

true false

3.1.4 Delimiters

Delimiters are special tokens that separate other tokens, and tell the compiler how to interpret associated tokens.

Parentheses and Braces Parentheses are used to force evaluation of parts of a program in a specific order. They are also used to enclose arguments for a function.

```
int x,y,z,w;
x= 10; y = 20; z = 30;
w = (x+y) * z /* here, (x+y) will be executed first and then multiplied with z */
```

Commas Commas are used to separate function arguments and variable declarations.

Square Brackets Square brackets are used for array initialization, assignment, and access.

```
arrayName[0] = 3
```

Curly Braces Curly braces are used to enclose function definitions and blocks of code. In general, blocks enclosed within curly braces do not need to be terminated with semicolons.

Periods Periods are used to access attributes of Piece and Player data types.

3.1.5 Whitespace

Whitespace (unless used in a string literal) is used to separate tokens, but has no special meaning otherwise. List of whitespace characters: spaces, tabs and newlines.

3.2 Data Types

3.2.1 Primitive Data Types

int These are 32-bit signed integers that can range from -2,147,483,648 to 2,147,483,647

string All text values will be of this type.

bool A truth value that can be either true or false.

void Used only as a return type in a function definition; specifying void as the return type of a function means that the function does not return anything.

3.2.2 Non-primitive Data Types

Arrays Arrays are ordered, fixed-size lists that can be used to hold both primitive and non-primitive data-types. All elements of an array must be of the same type. An array must be initialized with its size.

Declaring Arrays You can declare an array by indicating the type of the elements that the array will contain, followed by brackets enclosing the number of elements an array will hold, followed by an identifier for the array. For example:

```
int[5] myArray;
/* This declares an array named myArray that can hold 5 integers. */
```

Accessing and setting array elements Array elements can be accessed by providing the desired index of the element in the array you wish to access enclosed within brackets next to the identifier of the array. For example:

```
myArray[1]; /* This returns the element in myArray at index 1. */

myArray[2,3]; /* This returns the element at index 3 of the array that starts
at index 2 of myArray. */
```

Array elements can be set by accessing the element via the desired index in which to place the item and then assigning the desired value to the entry. For example:

```
myArray[1]=4; /* sets the element in myArray at index 1 to 4. */
```

Grid *Grid* is a global 2D array of struct pointers of type *GenericPiece*. On *Grid* initialization, all elements of the *Grid* point to null.

```
Grid_Init<5,5>;
```

Each cell of the *Grid* can point to the head of a linked list containing nodes of type *GenericPiece*. The head of the linked list can be obtained using the getPieceFromGrid() library function. The rest of the elements can be obtained using the next pointer.

NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL

Player Players are represented as structs that can hold multiple pieces as their fields. An analogy would be chess where a player owns pieces of some particular type e.g. knight, bishop or rook. The attribute distinguishing pieces can be color - black or white. Apart from pieces, the programmer can also define other attributes for a player. Moreover, when a piece is added to the *Grid*, the *GenericPiece* created(discussed below) for that piece is linked back to the player it belongs to in the owner field of *GenericPiece*.

An example of defining a *Player* type is shown below:

```
Player {
    Piece Pawn pawn;
    Piece Bishop bishop;
    Piece Knight knight;
    Piece Rook rook;
    Piece King king;
    string color;
}
```

Piece Anything placed on the board is a *Piece*. It may or may not belong to a player depending on whether its declared inside a player struct. Our language imposes a constraint on the programmer to assign a *displayString* for every piece. This helps identify pieces uniquely. The declaration for *displayString* is added to every piece struct in pre-processing. A *Piece* is essentially a struct with a function called *rule*. The rules for an item define its valid movements on the *Grid*.

```
Piece King {
    bool win;
    string displayString;
    bool exists;
        int rule(int src_x, int src_y, int dst_x, int dst_y){
            return 1;
        }
}
```

rule returns 1 if the intended move for that piece is valid, otherwise it returns 0 and its the current player's turn again. *rule* accepts source and destination coordinates as its arguments.

To distinguish between rules of different pieces, the type of the piece is prepended to the name of rule before the function body for every rule is built. For example, rule for piece *Pawn* becomes *Pawnrule*, Piece Bishop becomes *Bishoprule*.

Pieces can be added or removed from a *Grid* cell using the commands-

```
Grid<row,col> <-- p1;   //add
Grid<row,col> --> p1;   //remove
```

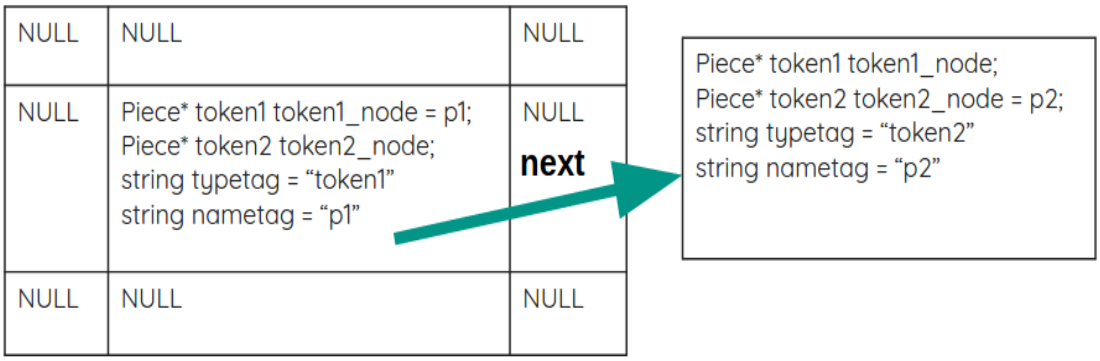
GenericPiece One of the greatest challenges of building a complex board game is to store pieces of different types on the same *Grid* cell. This is tricky as the *Grid* is itself of another type and we cannot assign pieces of different types while promising the guarantees of a strongly typed language. To get around this, we use a "super" piece of type *GenericPiece* which the programmer does not define, but one that gets added during preprocessing. *GenericPiece* contains declarations for pointers of all types of pieces defined by the programmer. It also has a next pointer since we need to construct a linked list for storing pieces.

To add any piece on a *Grid* cell, we first create a 'super' Piece called *newNode* of type *GenericPiece*. The piece that we want to add to Grid is assigned to the *newNode* pointer with the same type. We also assign the *nametag* and *typetag* of *newNode* to the *displayString* and data type of the piece. This helps locate the piece on the *Grid*.

```
Piece GenericPiece {
    Piece King* King_node;
    Piece Pawn* Pawn_node;
    Piece Bishop* Bishop_node;
    Piece Knight* Knight_node;
    Piece Rook* Rook_node;
    int x;
    int y;
    Piece GenericPiece* next;
    string nametag;
    string typetag;
    Player* owner;
}
```

After this is done, the *GenericPiece* holding the piece to be added is appended to the linked list of *GenericPieces* on the cell.

Figure 1: Linked List of GenericPieces on Grid cell



3.3 Expressions and Operators

3.3.1 Expressions

Expressions are made up of one or more operands and zero or more operators. Innermost expressions are evaluated first, as determined by grouping into parentheses, and operator precedence helps determine order

of evaluation. Expressions are otherwise evaluated left to right.

3.3.2 Operators

The table below presents the language operators (including assignment operators, mathematical operators, logical operators, and comparison operators), descriptions, and associativity rules. Operator precedence is highest at the top and lowest at the bottom of the table.

- '+' (Addition) : Adds values on either side of the operator.
- '-' (Subtraction) : Subtracts right-hand operand from left-hand operand.
- * (Multiplication) : Multiplies values on either side of the operator.
- / (Division) : Divides left-hand operand by right-hand operand.
- % (Modulus) : Divides left-hand operand by right-hand operand and returns remainder.
- Relational Operators: >, <, ==, <=, >=, ! = (standard meanings)
- Logical Operators: &&, ||, ! (standard meanings)
- Assignment Operator: = (standard meaning)

3.4 Statements

3.4.1 The If Statement

The if statement is used to execute a statement if a specified condition is met. If the specified condition is not met, the statement is skipped over. The general form of an if statement is as follows:

```
if(condition) {
    statement1
    statement2
    statement3
}
else {
    statement4
}
```

3.4.2 The while statement

The while statement is used to execute a block of code continuously in a loop until the specified condition is no longer met. If the condition is not met upon initially reaching the while loop, the code is never executed. The general structure of a while loop is as follows:

```
while(condition) {
    Action1
    Action2
    Action3
}
```

3.5 Functions

3.5.1 Function Definitions

Function definitions consist of a return type, a function identifier, a set of parameters and their types, and then a block of code to execute when that function is called with the specified parameters. An example of an addition function definition is as follows:

```
int sum (int a, int b) {
    return a+b;
}
```

3.5.2 Calling Functions

A function can be called its identifier followed by its params in parentheses. For example

```
sum(1,2)
```

3.5.3 Built-in functions

diceThrow() Generate a pseudo-random number between 1 and 6.

print(String s) Prints s on console, followed by nextline character.

print(int x) Prints x on console, followed by nextline character.

print_sameline(string s) Prints s on console.

print_sameline(int x) Prints x on console.

prompt(string msg) Prompts user for input by printing msg.

getLen(string s) Gets length of string s.

abs(int x) Gives the absolute value of int x.

3.5.4 Library Functions

printGrid() Prints the grid on the console. The Grid auto-resizes based on the number of pieces or the size of displayStrings.

addToGrid(int x, int y, Piece GenericPiece* p_n) Appends p_n to the linked list of GenericPiece pointers at Grid cell (x,y) by traversing the list until a null is found.

deleteFromGrid(int x, int y, string tag) Delete the GenericPiece from linked list at Grid cell (x,y) which has its nametag equal to tag.

moveOnGrid(Piece GenericPiece* p_n, int dst_x, int dst_y) Trigger rule for piece type specified by typetag. If rule returns 1 then move p_n to coordinate specified by (dst_x,dst_y), else currentPlayerIndex does not increment.

getPieceAtLocation(int x, int y) Return the head of the linked list at Grid cell (x,y).

getPieceFromGrid(string displayString) Return the GenericPiece from the Grid with nametag equal to displayString.

checkBound(int x, int y) Checks if the coordinate (x,y) is within the Grid bounds. Returns 1 on true, else 0.

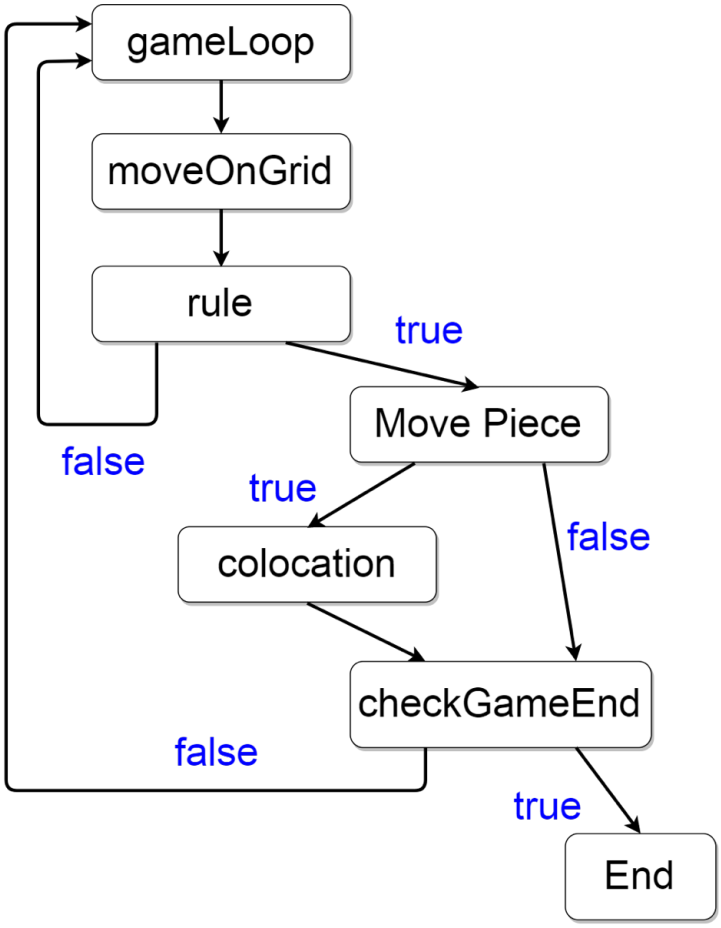
nextPlayer() Increments currentPlayerIndex by 1 to signify the next player's turn.

traverse(int src_x, int src_y, int dst_x, int dst_y) Traverses a row, a column or a diagonal in the grid with start coordinates given by (src_x, src_y) and end coordinates (dst_x, dst_y). Returns 1 if there is a Piece along that row, column or diagonal. Returns 0 otherwise. E.g.

```
traverse(1,1,3,3)
/* returns 1 if there is a piece between (1,1) and (3,3)*/
```

3.6 Program Structure and Scope

The entire game code will be in 1 source file. The flow followed by the code, written by any developer, will be as follows:



The aforementioned blocks imply the following:

Layout(int rowCount, int colCount) This block of code is executed first. It is used to set up the game Grid. It takes two parameters: rowCount and colCount, which describe its size. Creation of items, players, playerOrder and the location of these items and players on the Grid will be described here.

gameLoop() This block of code gets executed repeatedly until the game ends. Each execution of this block signifies one turn in the game. The gameLoop reads the order of turns from a built-in list called playerOrder which is defined by the programmer in the layout block. For e.g

```
playerOrder = ['p1', 'p2', 'p3', 'p4']
```

Whenever a player exits the game the programmer deletes this player from the list. As a result, the gameLoop will only run for the remaining players. Inside the gameLoop, the programmer can use a placeholder 'p' which at runtime gets replaced by the player whose turn is executing.

int rule(int, int, int, int) Each Piece has a rule(), and it will be triggered whenever an attempt is made to move a Piece of that type on the grid. It is used to verify valid movement of the Piece. Returning 0 makes the move attempt fail, while returning 1 performs the move on the grid. Eg:

```
int rule (int src_x, int src_y, int dst_x, int dst_y) {
        #Developer code here
}
```

int colocation(int, int, Piece GenericPiece*, Piece GenericPiece*) This block of code gets triggered when two Pieces are on the same cell of the grid. It is executed only after the move made is validated by the rule() block. The significance of this block is to determine if the Piece's resultant position on the board leads to an interaction with an existing Piece on the board in that position.

Eg:

```
int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2) {
  deleteFromGrid(x,y,i2.nametag);
  return 0;
}
```

checkGameEnd This block of code gets executed after every iteration of the gameLoop (and after any colocation events have been handled). The developer would place the logic to test the end of the game in this block. If checkGameEnd returns 0, execution returns to the next iteration of gameLoop(). If it returns 1, the program ends. For example,

```
int checkGameEnd() {
        if (p.score == 100) {
                return 1;
        }
        return 0;
}
```

4 Project Plan

4.1 Process Flow

After planning of each feature, its complete specification was noted, and then a small prototype was developed to analyze its feasibility. Then test cases for the same were written, which were then made to pass during development.

4.2 Programming Style Guide

- Use spaces instead of tabs for indentation.
- Each program uses 2-space indentation.
- Variable names use under_scores and camelCase to increase readability.
- Multi-line comments are included wherever necessary.

4.3 Timeline

Date	Milestone
Feb 8 2017	Language proposal completed
Feb 16 2017	Received feedback on proposal
Feb 22 2017	LRM completed
March 1 2017	Started Scanner and Parser
March 10 2017	Received feedback on LRM
March 20 2017	Hello World completed
April 9 2017	Added Preprocessor
April 9 2017	2D Arrays completed
April 27 2017	Grid, Structs and Lists completed
April 28 2017	Rules completed
May 3 2017	Chess completed
May 5 2017	Semantic checker completed
May 7 2017	Test Suite completed
May 9 2017	Snakes and Ladders completed
May 10 2017	Final report and project submitted

4.4 Roles and Responsibilities

Every member of the team put on different hats as per the requirement and contributed in an equal manner to the project.

Language Guru	Sagar Damani
Manager	Parth Panchmatia
System Architect	Dhruv Shekhawat
Tester	Akshay Nagpal

4.5 Development Environment

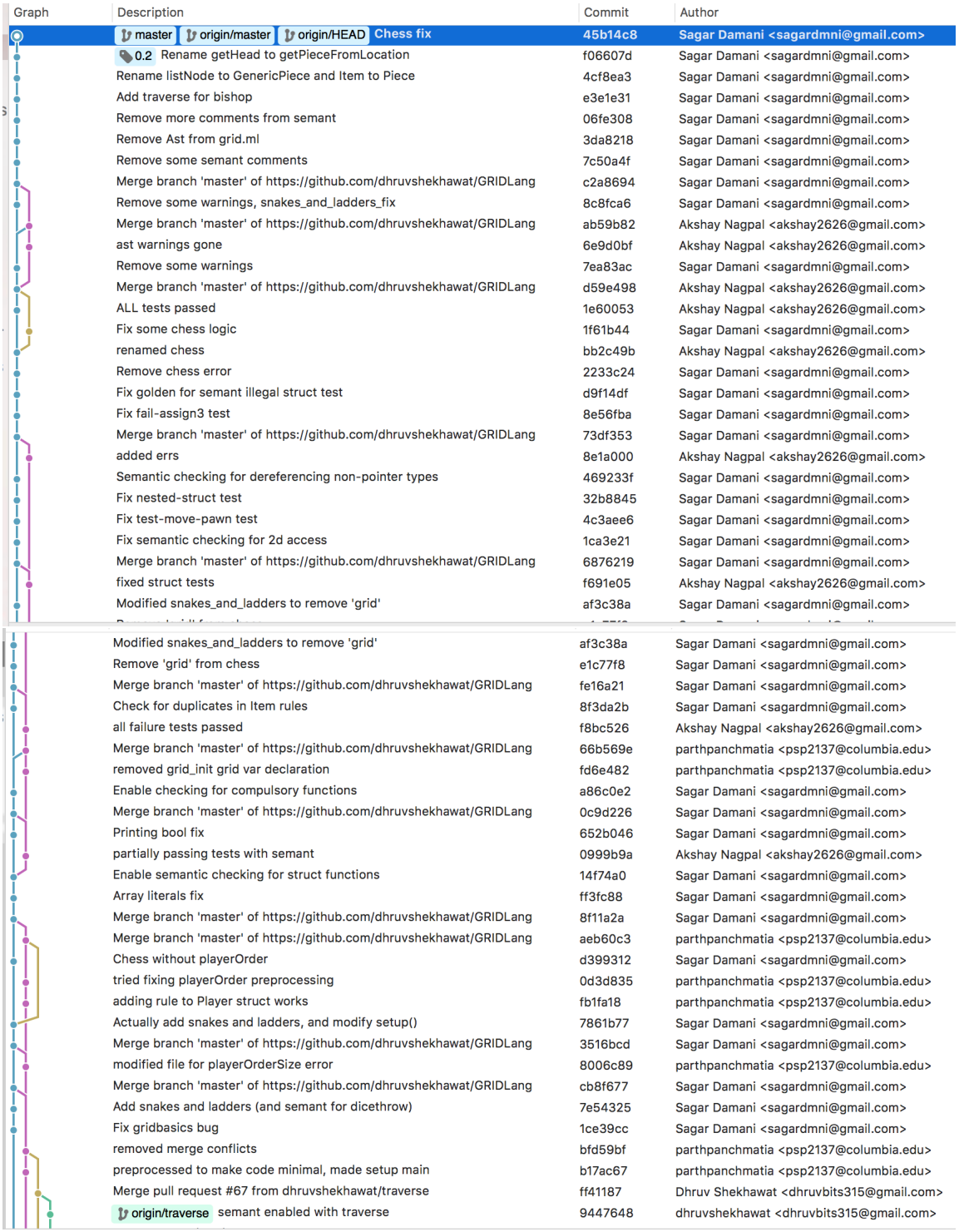
4.5.1 Environment and Programming Language

All the team members used the Virtualbox image provided by Professor Edwards which had Ubuntu 15.10, OCaml 4.01.0 and LLVM 3.6.2 .Ocamlyacc and Ocamllex extensions were used for compiling the scanner and parser front end. C was used to add some built in functions to our language.

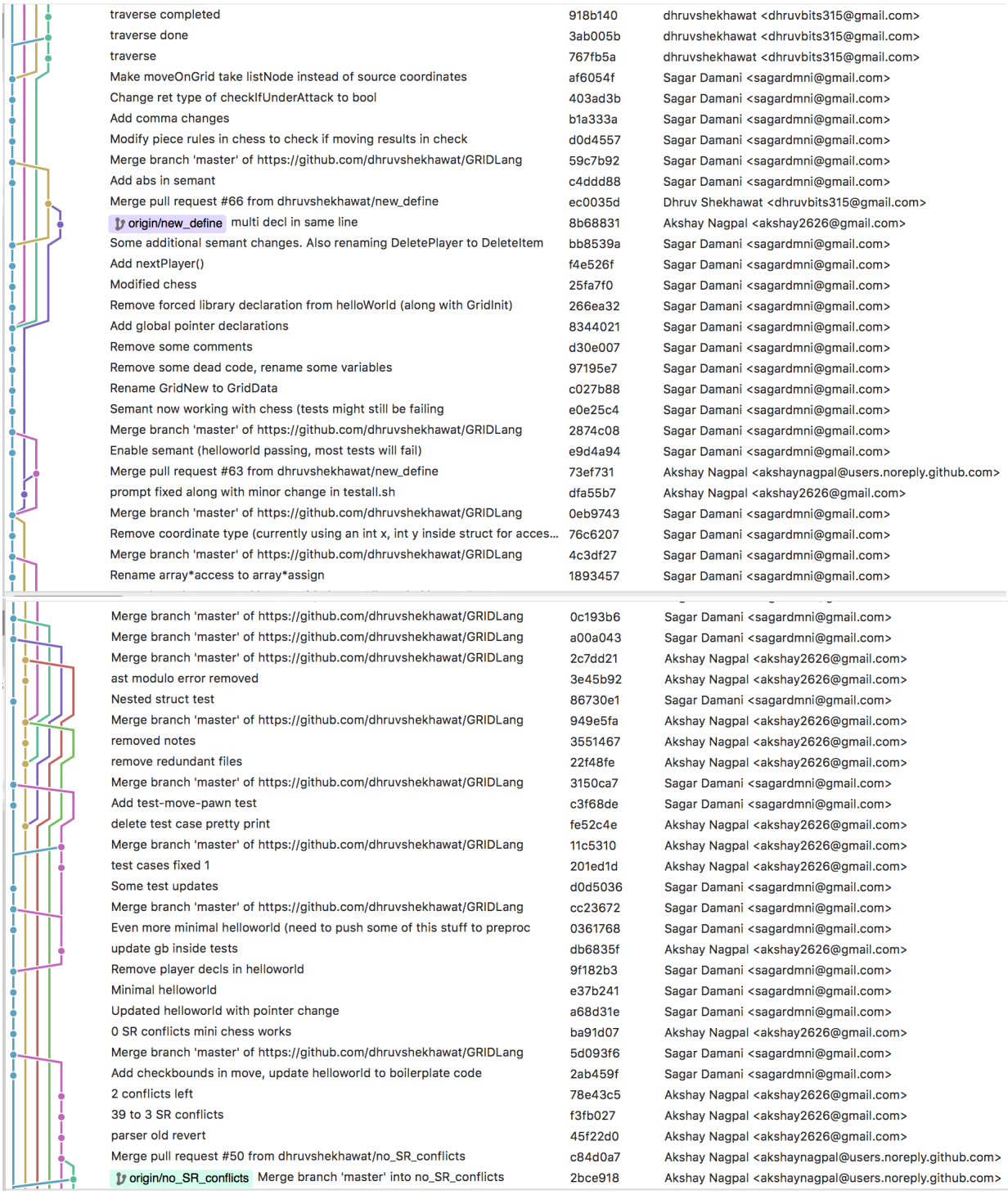
4.5.2 Tools

We used Vim and Sublime Text Editor to code all the files and git was used to have version control for our project. GCC was used to compile the C file used.

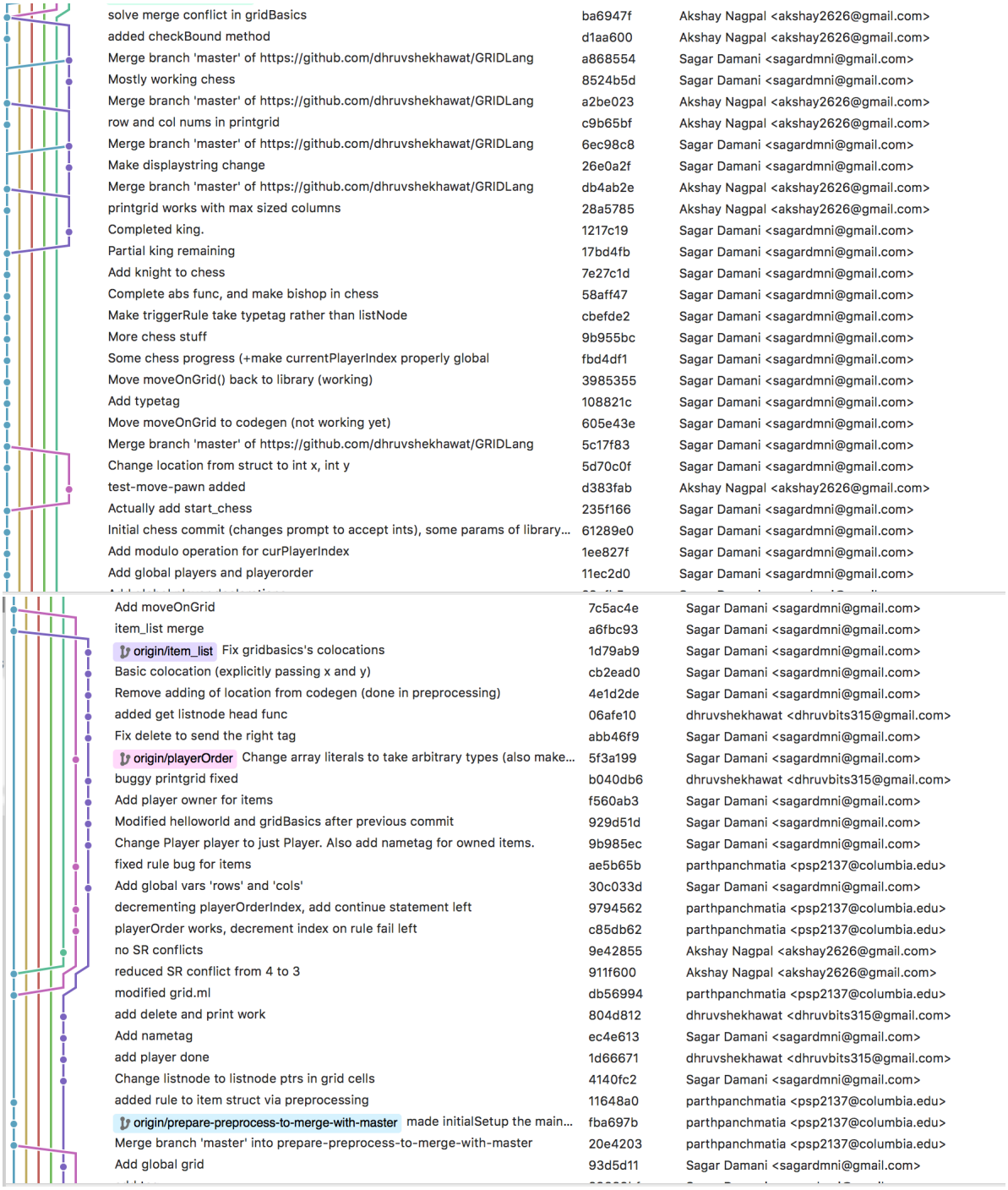
4.6 Project Log



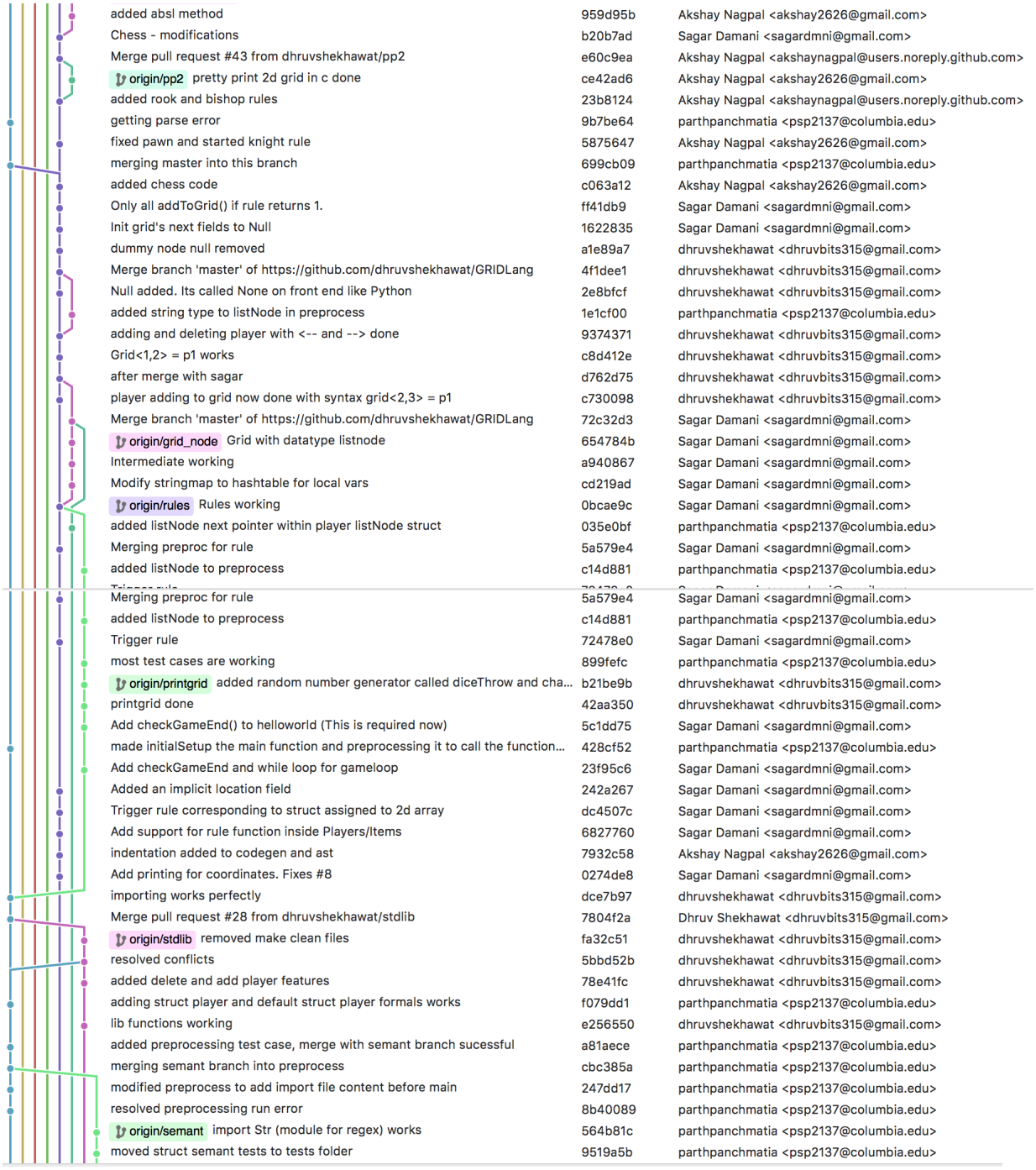
Graph	Description	Commit	Author
	master origin/master origin/HEAD **Chess fix**	**45b14c8**	**Sagar Damani <sagardmni@gmail.com>**
	0.2 Rename getHead to getPieceFromLocation	f06607d	Sagar Damani <sagardmni@gmail.com>
	Rename listNode to GenericPiece and Item to Piece	4cf8ea3	Sagar Damani <sagardmni@gmail.com>
	Add traverse for bishop	e3e1e31	Sagar Damani <sagardmni@gmail.com>
	Remove more comments from semant	06fe308	Sagar Damani <sagardmni@gmail.com>
	Remove Ast from grid.ml	3da8218	Sagar Damani <sagardmni@gmail.com>
	Remove some semant comments	7c50a4f	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	c2a8694	Sagar Damani <sagardmni@gmail.com>
	Remove some warnings, snakes_and_ladders_fix	8c8fca6	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	ab59b82	Akshay Nagpal <akshay2626@gmail.com>
	ast warnings gone	6e9d0bf	Akshay Nagpal <akshay2626@gmail.com>
	Remove some warnings	7ea83ac	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	d59e498	Akshay Nagpal <akshay2626@gmail.com>
	ALL tests passed	1e60053	Akshay Nagpal <akshay2626@gmail.com>
	Fix some chess logic	1f61b44	Sagar Damani <sagardmni@gmail.com>
	renamed chess	bb2c49b	Akshay Nagpal <akshay2626@gmail.com>
	Remove chess error	2233c24	Sagar Damani <sagardmni@gmail.com>
	Fix golden for semant illegal struct test	d9f14df	Sagar Damani <sagardmni@gmail.com>
	Fix fail-assign3 test	8e56fba	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	73df353	Sagar Damani <sagardmni@gmail.com>
	added errs	8e1a000	Akshay Nagpal <akshay2626@gmail.com>
	Semantic checking for dereferencing non-pointer types	469233f	Sagar Damani <sagardmni@gmail.com>
	Fix nested-struct test	32b8845	Sagar Damani <sagardmni@gmail.com>
	Fix test-move-pawn test	4c3aee6	Sagar Damani <sagardmni@gmail.com>
	Fix semantic checking for 2d access	1ca3e21	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	6876219	Sagar Damani <sagardmni@gmail.com>
	fixed struct tests	f691e05	Akshay Nagpal <akshay2626@gmail.com>
	Modified snakes_and_ladders to remove 'grid'	af3c38a	Sagar Damani <sagardmni@gmail.com>
	Modified snakes_and_ladders to remove 'grid'	af3c38a	Sagar Damani <sagardmni@gmail.com>
	Remove 'grid' from chess	e1c77f8	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	fe16a21	Sagar Damani <sagardmni@gmail.com>
	Check for duplicates in Item rules	8f3da2b	Sagar Damani <sagardmni@gmail.com>
	all failure tests passed	f8bc526	Akshay Nagpal <akshay2626@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	66b569e	parthpanchmatia <psp2137@columbia.edu>
	removed grid_init grid var declaration	fd6e482	parthpanchmatia <psp2137@columbia.edu>
	Enable checking for compulsory functions	a86c0e2	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	0c9d226	Sagar Damani <sagardmni@gmail.com>
	Printing bool fix	652b046	Sagar Damani <sagardmni@gmail.com>
	partially passing tests with semant	0999b9a	Akshay Nagpal <akshay2626@gmail.com>
	Enable semantic checking for struct functions	14f74a0	Sagar Damani <sagardmni@gmail.com>
	Array literals fix	ff3fc88	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	8f11a2a	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	aeb60c3	parthpanchmatia <psp2137@columbia.edu>
	Chess without playerOrder	d399312	Sagar Damani <sagardmni@gmail.com>
	tried fixing playerOrder preprocessing	0d3d835	parthpanchmatia <psp2137@columbia.edu>
	adding rule to Player struct works	fb1fa18	parthpanchmatia <psp2137@columbia.edu>
	Actually add snakes and ladders, and modify setup()	7861b77	Sagar Damani <sagardmni@gmail.com>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	3516bcd	Sagar Damani <sagardmni@gmail.com>
	modified file for playerOrderSize error	8006c89	parthpanchmatia <psp2137@columbia.edu>
	Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	cb8f677	Sagar Damani <sagardmni@gmail.com>
	Add snakes and ladders (and semant for dicethrow)	7e54325	Sagar Damani <sagardmni@gmail.com>
	Fix gridbasics bug	1ce39cc	Sagar Damani <sagardmni@gmail.com>
	removed merge conflicts	bfd59bf	parthpanchmatia <psp2137@columbia.edu>
	preprocessed to make code minimal, made setup main	b17ac67	parthpanchmatia <psp2137@columbia.edu>
	Merge pull request #67 from dhruvshekhawat/traverse	ff41187	Dhruv Shekhawat <dhruvbits315@gmail.com>
	origin/traverse semant enabled with traverse	9447648	dhruvshekhawat <dhruvbits315@gmail.com>



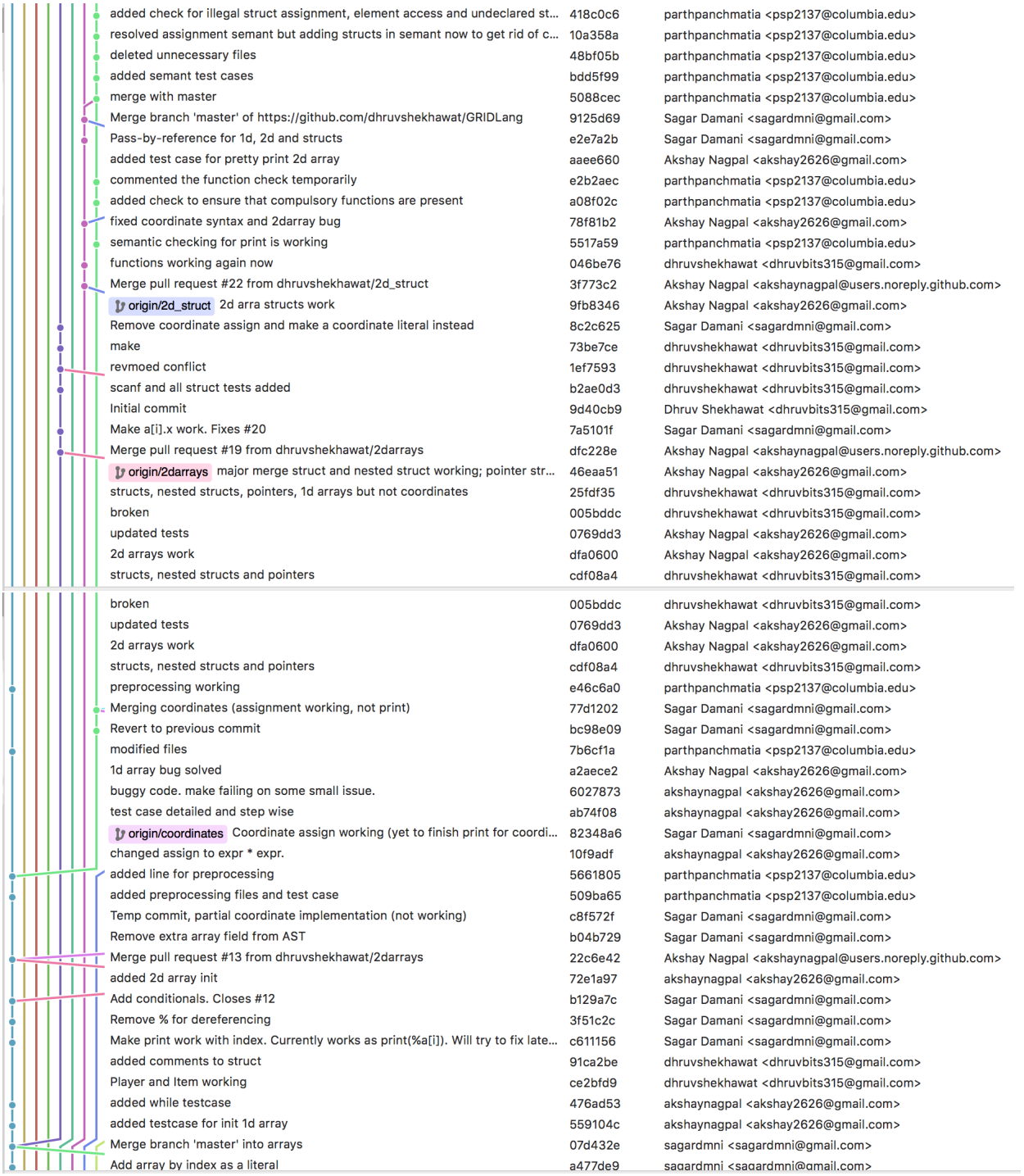
traverse completed	918b140	dhruvshekhawat <dhruvbits315@gmail.com>
traverse done	3ab005b	dhruvshekhawat <dhruvbits315@gmail.com>
traverse	767fb5a	dhruvshekhawat <dhruvbits315@gmail.com>
Make moveOnGrid take listNode instead of source coordinates	af6054f	Sagar Damani <sagardmni@gmail.com>
Change ret type of checkIfUnderAttack to bool	403ad3b	Sagar Damani <sagardmni@gmail.com>
Add comma changes	b1a333a	Sagar Damani <sagardmni@gmail.com>
Modify piece rules in chess to check if moving results in check	d0d4557	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	59c7b92	Sagar Damani <sagardmni@gmail.com>
Add abs in semant	c4ddd88	Sagar Damani <sagardmni@gmail.com>
Merge pull request #66 from dhruvshekhawat/new_define	ec0035d	Dhruv Shekhawat <dhruvbits315@gmail.com>
origin/new_define multi decl in same line	8b68831	Akshay Nagpal <akshay2626@gmail.com>
Some additional semant changes. Also renaming DeletePlayer to DeleteItem	bb8539a	Sagar Damani <sagardmni@gmail.com>
Add nextPlayer()	f4e526f	Sagar Damani <sagardmni@gmail.com>
Modified chess	25fa7f0	Sagar Damani <sagardmni@gmail.com>
Remove forced library declaration from helloWorld (along with GridInit)	266ea32	Sagar Damani <sagardmni@gmail.com>
Add global pointer declarations	8344021	Sagar Damani <sagardmni@gmail.com>
Remove some comments	d30e007	Sagar Damani <sagardmni@gmail.com>
Remove some dead code, rename some variables	97195e7	Sagar Damani <sagardmni@gmail.com>
Rename GridNew to GridData	c027b88	Sagar Damani <sagardmni@gmail.com>
Semant now working with chess (tests might still be failing	e0e25c4	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	2874c08	Sagar Damani <sagardmni@gmail.com>
Enable semant (helloworld passing, most tests will fail)	e9d4a94	Sagar Damani <sagardmni@gmail.com>
Merge pull request #63 from dhruvshekhawat/new_define	73ef731	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
prompt fixed along with minor change in testall.sh	dfa55b7	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	0eb9743	Sagar Damani <sagardmni@gmail.com>
Remove coordinate type (currently using an int x, int y inside struct for acces...	76c6207	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	4c3df27	Sagar Damani <sagardmni@gmail.com>
Rename array*access to array*assign	1893457	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	0c193b6	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	a00a043	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	2c7dd21	Akshay Nagpal <akshay2626@gmail.com>
ast modulo error removed	3e45b92	Akshay Nagpal <akshay2626@gmail.com>
Nested struct test	86730e1	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	949e5fa	Akshay Nagpal <akshay2626@gmail.com>
removed notes	3551467	Akshay Nagpal <akshay2626@gmail.com>
remove redundant files	22f48fe	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	3150ca7	Sagar Damani <sagardmni@gmail.com>
Add test-move-pawn test	c3f68de	Sagar Damani <sagardmni@gmail.com>
delete test case pretty print	fe52c4e	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	11c5310	Akshay Nagpal <akshay2626@gmail.com>
test cases fixed 1	201ed1d	Akshay Nagpal <akshay2626@gmail.com>
Some test updates	d0d5036	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	cc23672	Sagar Damani <sagardmni@gmail.com>
Even more minimal helloworld (need to push some of this stuff to preproc	0361768	Sagar Damani <sagardmni@gmail.com>
update gb inside tests	db6835f	Akshay Nagpal <akshay2626@gmail.com>
Remove player decls in helloworld	9f182b3	Sagar Damani <sagardmni@gmail.com>
Minimal helloworld	e37b241	Sagar Damani <sagardmni@gmail.com>
Updated helloworld with pointer change	a68d31e	Sagar Damani <sagardmni@gmail.com>
0 SR conflicts mini chess works	ba91d07	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	5d093f6	Sagar Damani <sagardmni@gmail.com>
Add checkbounds in move, update helloworld to boilerplate code	2ab459f	Sagar Damani <sagardmni@gmail.com>
2 conflicts left	78e43c5	Akshay Nagpal <akshay2626@gmail.com>
39 to 3 SR conflicts	f3fb027	Akshay Nagpal <akshay2626@gmail.com>
parser old revert	45f22d0	Akshay Nagpal <akshay2626@gmail.com>
Merge pull request #50 from dhruvshekhawat/no_SR_conflicts	c84d0a7	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
origin/no_SR_conflicts Merge branch 'master' into no_SR_conflicts	2bce918	Akshay Nagpal <akshaynagpal@users.noreply.github.com>

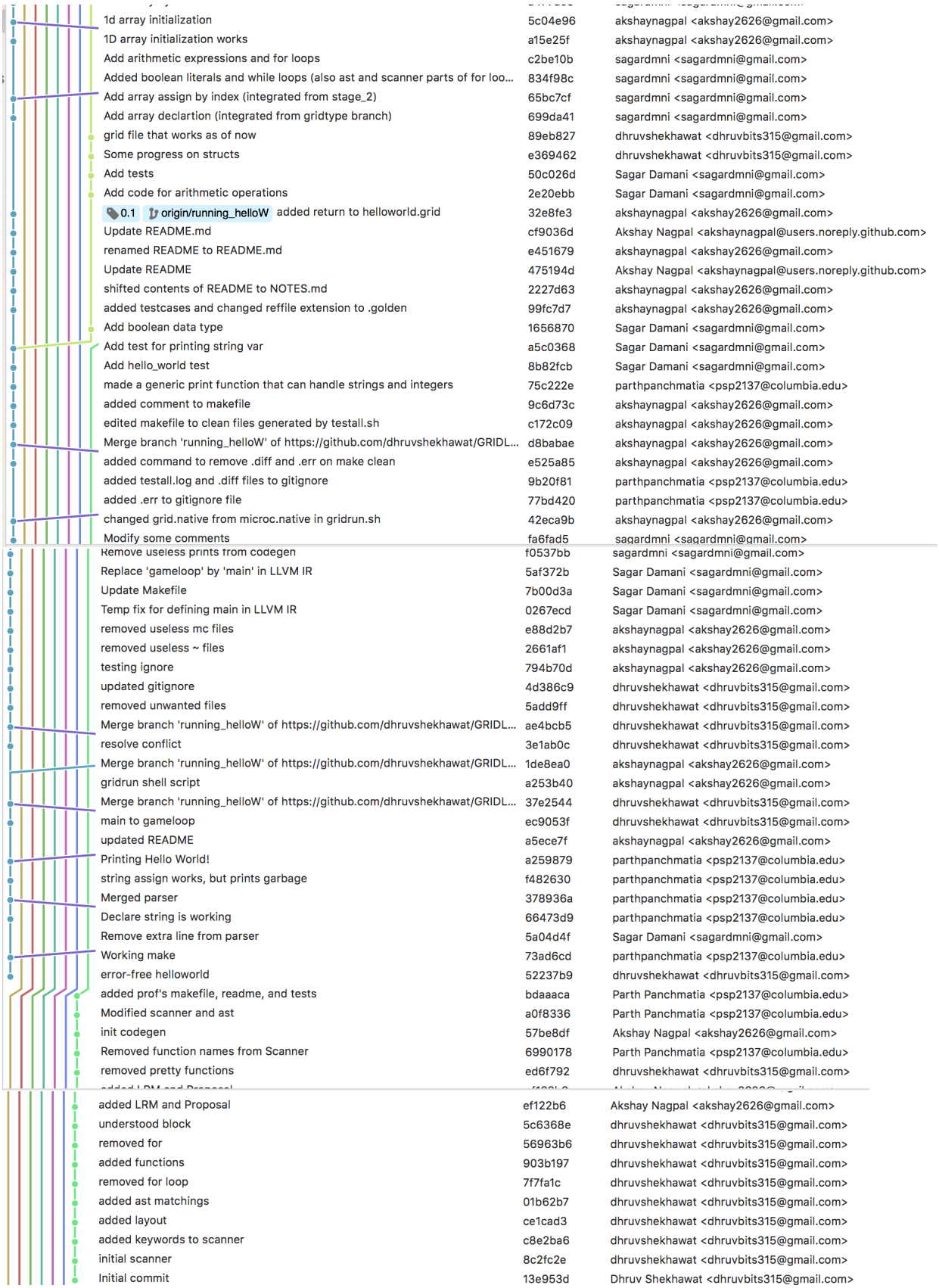


solve merge conflict in gridBasics	ba6947f	Akshay Nagpal <akshay2626@gmail.com>
added checkBound method	d1aa600	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	a868554	Sagar Damani <sagardmni@gmail.com>
Mostly working chess	8524b5d	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	a2be023	Akshay Nagpal <akshay2626@gmail.com>
row and col nums in printgrid	c9b65bf	Akshay Nagpal <akshay2626@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	6ec98c8	Sagar Damani <sagardmni@gmail.com>
Make displaystring change	26e0a2f	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	db4ab2e	Akshay Nagpal <akshay2626@gmail.com>
printgrid works with max sized columns	28a5785	Akshay Nagpal <akshay2626@gmail.com>
Completed king.	1217c19	Sagar Damani <sagardmni@gmail.com>
Partial king remaining	17bd4fb	Sagar Damani <sagardmni@gmail.com>
Add knight to chess	7e27c1d	Sagar Damani <sagardmni@gmail.com>
Complete abs func, and make bishop in chess	58aff47	Sagar Damani <sagardmni@gmail.com>
Make triggerRule take typetag rather than listNode	cbefde2	Sagar Damani <sagardmni@gmail.com>
More chess stuff	9b955bc	Sagar Damani <sagardmni@gmail.com>
Some chess progress (+make currentPlayerIndex properly global	fbd4df1	Sagar Damani <sagardmni@gmail.com>
Move moveOnGrid() back to library (working)	3985355	Sagar Damani <sagardmni@gmail.com>
Add typetag	108821c	Sagar Damani <sagardmni@gmail.com>
Move moveOnGrid to codegen (not working yet)	605e43e	Sagar Damani <sagardmni@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	5c17f83	Sagar Damani <sagardmni@gmail.com>
Change location from struct to int x, int y	5d70c0f	Sagar Damani <sagardmni@gmail.com>
test-move-pawn added	d383fab	Akshay Nagpal <akshay2626@gmail.com>
Actually add start_chess	235f166	Sagar Damani <sagardmni@gmail.com>
Initial chess commit (changes prompt to accept ints), some params of library...	61289e0	Sagar Damani <sagardmni@gmail.com>
Add modulo operation for curPlayerIndex	1ee827f	Sagar Damani <sagardmni@gmail.com>
Add global players and playerorder	11ec2d0	Sagar Damani <sagardmni@gmail.com>
Add moveOnGrid	7c5ac4e	Sagar Damani <sagardmni@gmail.com>
item_list merge	a6fbc93	Sagar Damani <sagardmni@gmail.com>
origin/item_list Fix gridbasics's colocations	1d79ab9	Sagar Damani <sagardmni@gmail.com>
Basic colocation (explicitly passing x and y)	cb2ead0	Sagar Damani <sagardmni@gmail.com>
Remove adding of location from codegen (done in preprocessing)	4e1d2de	Sagar Damani <sagardmni@gmail.com>
added get listnode head func	06afe10	dhruvshekhawat <dhruvbits315@gmail.com>
Fix delete to send the right tag	abb46f9	Sagar Damani <sagardmni@gmail.com>
origin/playerOrder Change array literals to take arbitrary types (also make...	5f3a199	Sagar Damani <sagardmni@gmail.com>
buggy printgrid fixed	b040db6	dhruvshekhawat <dhruvbits315@gmail.com>
Add player owner for items	f560ab3	Sagar Damani <sagardmni@gmail.com>
Modified helloworld and gridBasics after previous commit	929d51d	Sagar Damani <sagardmni@gmail.com>
Change Player player to just Player. Also add nametag for owned items.	9b985ec	Sagar Damani <sagardmni@gmail.com>
fixed rule bug for items	ae5b65b	parthpanchmatia <psp2137@columbia.edu>
Add global vars 'rows' and 'cols'	30c033d	Sagar Damani <sagardmni@gmail.com>
decrementing playerOrderIndex, add continue statement left	9794562	parthpanchmatia <psp2137@columbia.edu>
playerOrder works, decrement index on rule fail left	c85db62	parthpanchmatia <psp2137@columbia.edu>
no SR conflicts	9e42855	Akshay Nagpal <akshay2626@gmail.com>
reduced SR conflict from 4 to 3	911f600	Akshay Nagpal <akshay2626@gmail.com>
modified grid.ml	db56994	parthpanchmatia <psp2137@columbia.edu>
add delete and print work	804d812	dhruvshekhawat <dhruvbits315@gmail.com>
Add nametag	ec4e613	Sagar Damani <sagardmni@gmail.com>
add player done	1d66671	dhruvshekhawat <dhruvbits315@gmail.com>
Change listnode to listnode ptrs in grid cells	4140fc2	Sagar Damani <sagardmni@gmail.com>
added rule to item struct via preprocessing	11648a0	parthpanchmatia <psp2137@columbia.edu>
origin/prepare-preprocess-to-merge-with-master made initialSetup the main...	fba697b	parthpanchmatia <psp2137@columbia.edu>
Merge branch 'master' into prepare-preprocess-to-merge-with-master	20e4203	parthpanchmatia <psp2137@columbia.edu>
Add global grid	93d5d11	Sagar Damani <sagardmni@gmail.com>



added absl method	959d95b	Akshay Nagpal <akshay2626@gmail.com>
Chess - modifications	b20b7ad	Sagar Damani <sagardmni@gmail.com>
Merge pull request #43 from dhruvshekhawat/pp2	e60c9ea	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
origin/pp2 pretty print 2d grid in c done	ce42ad6	Akshay Nagpal <akshay2626@gmail.com>
added rook and bishop rules	23b8124	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
getting parse error	9b7be64	parthpanchmatia <psp2137@columbia.edu>
fixed pawn and started knight rule	5875647	Akshay Nagpal <akshay2626@gmail.com>
merging master into this branch	699cb09	parthpanchmatia <psp2137@columbia.edu>
added chess code	c063a12	Akshay Nagpal <akshay2626@gmail.com>
Only all addToGrid() if rule returns 1.	ff41db9	Sagar Damani <sagardmni@gmail.com>
Init grid's next fields to Null	1622835	Sagar Damani <sagardmni@gmail.com>
dummy node null removed	a1e89a7	dhruvshekhawat <dhruvbits315@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	4f1dee1	dhruvshekhawat <dhruvbits315@gmail.com>
Null added. Its called None on front end like Python	2e8bfcf	dhruvshekhawat <dhruvbits315@gmail.com>
added string type to listNode in preprocess	1e1cf00	parthpanchmatia <psp2137@columbia.edu>
adding and deleting player with <-- and --> done	9374371	dhruvshekhawat <dhruvbits315@gmail.com>
Grid<1,2> = p1 works	c8d412e	dhruvshekhawat <dhruvbits315@gmail.com>
after merge with sagar	d762d75	dhruvshekhawat <dhruvbits315@gmail.com>
player adding to grid now done with syntax grid<2,3> = p1	c730098	dhruvshekhawat <dhruvbits315@gmail.com>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang	72c32d3	Sagar Damani <sagardmni@gmail.com>
origin/grid_node Grid with datatype listnode	654784b	Sagar Damani <sagardmni@gmail.com>
Intermediate working	a940867	Sagar Damani <sagardmni@gmail.com>
Modify stringmap to hashtable for local vars	cd219ad	Sagar Damani <sagardmni@gmail.com>
origin/rules Rules working	0bcae9c	Sagar Damani <sagardmni@gmail.com>
added listNode next pointer within player listNode struct	035e0bf	parthpanchmatia <psp2137@columbia.edu>
Merging preproc for rule	5a579e4	Sagar Damani <sagardmni@gmail.com>
added listNode to preprocess	c14d881	parthpanchmatia <psp2137@columbia.edu>
Trigger rule	72478e0	Sagar Damani <sagardmni@gmail.com>
most test cases are working	899fefc	parthpanchmatia <psp2137@columbia.edu>
origin/printgrid added random number generator called diceThrow and cha...	b21be9b	dhruvshekhawat <dhruvbits315@gmail.com>
printgrid done	42aa350	dhruvshekhawat <dhruvbits315@gmail.com>
Add checkGameEnd() to helloworld (This is required now)	5c1dd75	Sagar Damani <sagardmni@gmail.com>
made initialSetup the main function and preprocessing it to call the function...	428cf52	parthpanchmatia <psp2137@columbia.edu>
Add checkGameEnd and while loop for gameloop	23f95c6	Sagar Damani <sagardmni@gmail.com>
Added an implicit location field	242a267	Sagar Damani <sagardmni@gmail.com>
Trigger rule corresponding to struct assigned to 2d array	dc4507c	Sagar Damani <sagardmni@gmail.com>
Add support for rule function inside Players/Items	6827760	Sagar Damani <sagardmni@gmail.com>
indentation added to codegen and ast	7932c58	Akshay Nagpal <akshay2626@gmail.com>
Add printing for coordinates. Fixes #8	0274de8	Sagar Damani <sagardmni@gmail.com>
importing works perfectly	dce7b97	dhruvshekhawat <dhruvbits315@gmail.com>
Merge pull request #28 from dhruvshekhawat/stdlib	7804f2a	Dhruv Shekhawat <dhruvbits315@gmail.com>
origin/stdlib removed make clean files	fa32c51	dhruvshekhawat <dhruvbits315@gmail.com>
resolved conflicts	5bbd52b	dhruvshekhawat <dhruvbits315@gmail.com>
added delete and add player features	78e41fc	dhruvshekhawat <dhruvbits315@gmail.com>
adding struct player and default struct player formals works	f079dd1	parthpanchmatia <psp2137@columbia.edu>
lib functions working	e256550	dhruvshekhawat <dhruvbits315@gmail.com>
added preprocessing test case, merge with semant branch sucessful	a81aece	parthpanchmatia <psp2137@columbia.edu>
merging semant branch into preprocess	cbc385a	parthpanchmatia <psp2137@columbia.edu>
modified preprocess to add import file content before main	247dd17	parthpanchmatia <psp2137@columbia.edu>
resolved preprocessing run error	8b40089	parthpanchmatia <psp2137@columbia.edu>
origin/semant import Str (module for regex) works	564b81c	parthpanchmatia <psp2137@columbia.edu>
moved struct semant tests to tests folder	9519a5b	parthpanchmatia <psp2137@columbia.edu>

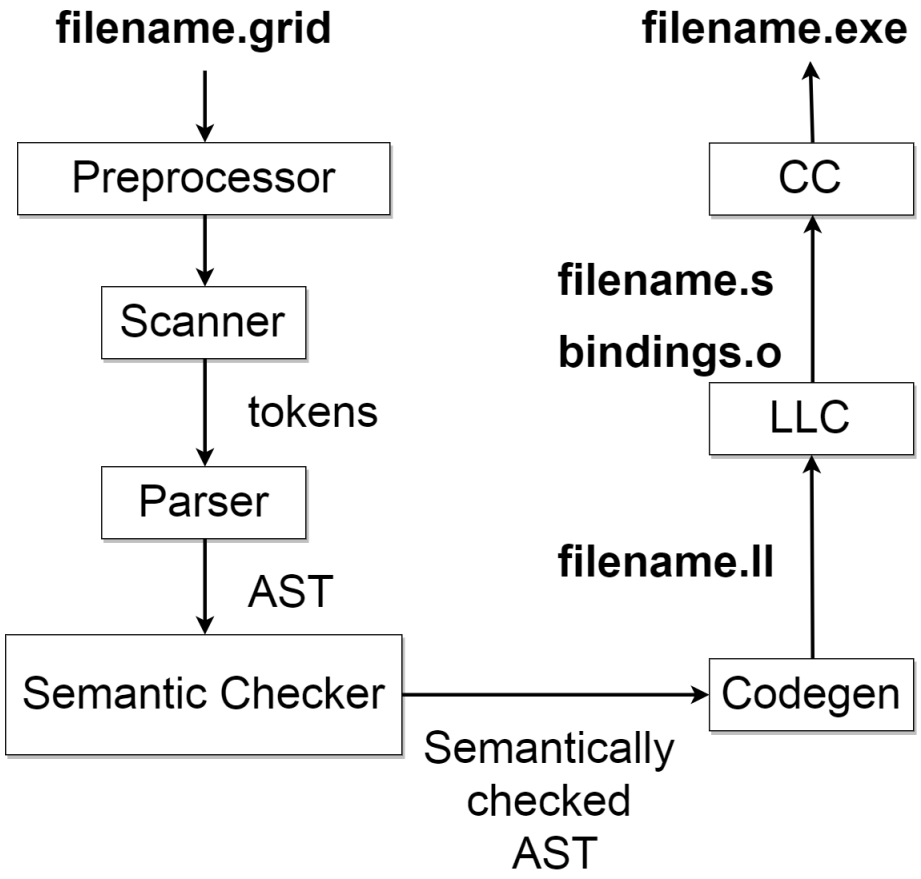

added check for illegal struct assignment, element access and undeclared st... 418c0c6 parthpanchmatia <psp2137@columbia.edu>
resolved assignment semant but adding structs in semant now to get rid of c... 10a358a parthpanchmatia <psp2137@columbia.edu>
deleted unnecessary files 48bf05b parthpanchmatia <psp2137@columbia.edu>
added semant test cases bdd5f99 parthpanchmatia <psp2137@columbia.edu>
merge with master 5088cec parthpanchmatia <psp2137@columbia.edu>
Merge branch 'master' of https://github.com/dhruvshekhawat/GRIDLang 9125d69 Sagar Damani <sagardmni@gmail.com>
Pass-by-reference for 1d, 2d and structs e2e7a2b Sagar Damani <sagardmni@gmail.com>
added test case for pretty print 2d array aaee660 Akshay Nagpal <akshay2626@gmail.com>
commented the function check temporarily e2b2aec parthpanchmatia <psp2137@columbia.edu>
added check to ensure that compulsory functions are present a08f02c parthpanchmatia <psp2137@columbia.edu>
fixed coordinate syntax and 2darray bug 78f81b2 Akshay Nagpal <akshay2626@gmail.com>
semantic checking for print is working 5517a59 parthpanchmatia <psp2137@columbia.edu>
functions working again now 046be76 dhruvshekhawat <dhruvbits315@gmail.com>
Merge pull request #22 from dhruvshekhawat/2d_struct 3f773c2 Akshay Nagpal <akshaynagpal@users.noreply.github.com>
origin/2d_struct 2d arra structs work 9fb8346 Akshay Nagpal <akshay2626@gmail.com>
Remove coordinate assign and make a coordinate literal instead 8c2c625 Sagar Damani <sagardmni@gmail.com>
make 73be7ce dhruvshekhawat <dhruvbits315@gmail.com>
revmoed conflict 1ef7593 dhruvshekhawat <dhruvbits315@gmail.com>
scanf and all struct tests added b2ae0d3 dhruvshekhawat <dhruvbits315@gmail.com>
Initial commit 9d40cb9 Dhruv Shekhawat <dhruvbits315@gmail.com>
Make a[i].x work. Fixes #20 7a5101f Sagar Damani <sagardmni@gmail.com>
Merge pull request #19 from dhruvshekhawat/2darrays dfc228e Akshay Nagpal <akshaynagpal@users.noreply.github.com>
origin/2darrays major merge struct and nested struct working; pointer str... 46eaa51 Akshay Nagpal <akshay2626@gmail.com>
structs, nested structs, pointers, 1d arrays but not coordinates 25fdf35 dhruvshekhawat <dhruvbits315@gmail.com>
broken 005bddc dhruvshekhawat <dhruvbits315@gmail.com>
updated tests 0769dd3 Akshay Nagpal <akshay2626@gmail.com>
2d arrays work dfa0600 Akshay Nagpal <akshay2626@gmail.com>
structs, nested structs and pointers cdf08a4 dhruvshekhawat <dhruvbits315@gmail.com>
broken 005bddc dhruvshekhawat <dhruvbits315@gmail.com>
updated tests 0769dd3 Akshay Nagpal <akshay2626@gmail.com>
2d arrays work dfa0600 Akshay Nagpal <akshay2626@gmail.com>
structs, nested structs and pointers cdf08a4 dhruvshekhawat <dhruvbits315@gmail.com>
preprocessing working e46c6a0 parthpanchmatia <psp2137@columbia.edu>
Merging coordinates (assignment working, not print) 77d1202 Sagar Damani <sagardmni@gmail.com>
Revert to previous commit bc98e09 Sagar Damani <sagardmni@gmail.com>
modified files 7b6cf1a parthpanchmatia <psp2137@columbia.edu>
1d array bug solved a2aece2 Akshay Nagpal <akshay2626@gmail.com>
buggy code. make failing on some small issue. 6027873 akshaynagpal <akshay2626@gmail.com>
test case detailed and step wise ab74f08 akshaynagpal <akshay2626@gmail.com>
origin/coordinates Coordinate assign working (yet to finish print for coordi... 82348a6 Sagar Damani <sagardmni@gmail.com>
changed assign to expr * expr. 10f9adf akshaynagpal <akshay2626@gmail.com>
added line for preprocessing 5661805 parthpanchmatia <psp2137@columbia.edu>
added preprocessing files and test case 509ba65 parthpanchmatia <psp2137@columbia.edu>
Temp commit, partial coordinate implementation (not working) c8f572f Sagar Damani <sagardmni@gmail.com>
Remove extra array field from AST b04b729 Sagar Damani <sagardmni@gmail.com>
Merge pull request #13 from dhruvshekhawat/2darrays 22c6e42 Akshay Nagpal <akshaynagpal@users.noreply.github.com>
added 2d array init 72e1a97 akshaynagpal <akshay2626@gmail.com>
Add conditionals. Closes #12 b129a7c Sagar Damani <sagardmni@gmail.com>
Remove % for dereferencing 3f51c2c Sagar Damani <sagardmni@gmail.com>
Make print work with index. Currently works as print(%a[i]). Will try to fix late... c611156 Sagar Damani <sagardmni@gmail.com>
added comments to struct 91ca2be dhruvshekhawat <dhruvbits315@gmail.com>
Player and Item working ce2bfd9 dhruvshekhawat <dhruvbits315@gmail.com>
added while testcase 476ad53 akshaynagpal <akshay2626@gmail.com>
added testcase for init 1d array 559104c akshaynagpal <akshay2626@gmail.com>
Merge branch 'master' into arrays 07d432e sagardmni <sagardmni@gmail.com>
Add array by index as a literal a477de9 sagardmni <sagardmni@gmail.com>



1d array initialization	5c04e96	akshaynagpal <akshay2626@gmail.com>
1D array initialization works	a15e25f	akshaynagpal <akshay2626@gmail.com>
Add arithmetic expressions and for loops	c2be10b	sagardmni <sagardmni@gmail.com>
Added boolean literals and while loops (also ast and scanner parts of for loo...	834f98c	sagardmni <sagardmni@gmail.com>
Add array assign by index (integrated from stage_2)	65bc7cf	sagardmni <sagardmni@gmail.com>
Add array declartion (integrated from gridtype branch)	699da41	sagardmni <sagardmni@gmail.com>
grid file that works as of now	89eb827	dhruvshekhawat <dhruvbits315@gmail.com>
Some progress on structs	e369462	dhruvshekhawat <dhruvbits315@gmail.com>
Add tests	50c026d	Sagar Damani <sagardmni@gmail.com>
Add code for arithmetic operations	2e20ebb	Sagar Damani <sagardmni@gmail.com>
0.1 origin/running_helloW added return to helloworld.grid	32e8fe3	akshaynagpal <akshay2626@gmail.com>
Update README.md	cf9036d	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
renamed README to README.md	e451679	akshaynagpal <akshay2626@gmail.com>
Update README	475194d	Akshay Nagpal <akshaynagpal@users.noreply.github.com>
shifted contents of README to NOTES.md	2227d63	akshaynagpal <akshay2626@gmail.com>
added testcases and changed reffile extension to .golden	99fc7d7	akshaynagpal <akshay2626@gmail.com>
Add boolean data type	1656870	Sagar Damani <sagardmni@gmail.com>
Add test for printing string var	a5c0368	Sagar Damani <sagardmni@gmail.com>
Add hello_world test	8b82fcb	Sagar Damani <sagardmni@gmail.com>
made a generic print function that can handle strings and integers	75c222e	parthpanchmatia <psp2137@columbia.edu>
added comment to makefile	9c6d73c	akshaynagpal <akshay2626@gmail.com>
edited makefile to clean files generated by testall.sh	c172c09	akshaynagpal <akshay2626@gmail.com>
Merge branch 'running_helloW' of https://github.com/dhruvshekhawat/GRIDL...	d8babae	akshaynagpal <akshay2626@gmail.com>
added command to remove .diff and .err on make clean	e525a85	akshaynagpal <akshay2626@gmail.com>
added testall.log and .diff files to gitignore	9b20f81	parthpanchmatia <psp2137@columbia.edu>
added .err to gitignore file	77bd420	parthpanchmatia <psp2137@columbia.edu>
changed grid.native from microc.native in gridrun.sh	42eca9b	akshaynagpal <akshay2626@gmail.com>
Modify some comments	fa6fad5	sagardmni <sagardmni@gmail.com>
Remove useless prints from codegen	f0537bb	sagardmni <sagardmni@gmail.com>
Replace 'gameloop' by 'main' in LLVM IR	5af372b	Sagar Damani <sagardmni@gmail.com>
Update Makefile	7b00d3a	Sagar Damani <sagardmni@gmail.com>
Temp fix for defining main in LLVM IR	0267ecd	Sagar Damani <sagardmni@gmail.com>
removed useless mc files	e88d2b7	akshaynagpal <akshay2626@gmail.com>
removed useless ~ files	2661af1	akshaynagpal <akshay2626@gmail.com>
testing ignore	794b70d	akshaynagpal <akshay2626@gmail.com>
updated gitignore	4d386c9	dhruvshekhawat <dhruvbits315@gmail.com>
removed unwanted files	5add9ff	dhruvshekhawat <dhruvbits315@gmail.com>
Merge branch 'running_helloW' of https://github.com/dhruvshekhawat/GRIDL...	ae4bcb5	dhruvshekhawat <dhruvbits315@gmail.com>
resolve conflict	3e1ab0c	dhruvshekhawat <dhruvbits315@gmail.com>
Merge branch 'running_helloW' of https://github.com/dhruvshekhawat/GRIDL...	1de8ea0	akshaynagpal <akshay2626@gmail.com>
gridrun shell script	a253b40	akshaynagpal <akshay2626@gmail.com>
Merge branch 'running_helloW' of https://github.com/dhruvshekhawat/GRIDL...	37e2544	dhruvshekhawat <dhruvbits315@gmail.com>
main to gameloop	ec9053f	dhruvshekhawat <dhruvbits315@gmail.com>
updated README	a5ece7f	akshaynagpal <akshay2626@gmail.com>
Printing Hello World!	a259879	parthpanchmatia <psp2137@columbia.edu>
string assign works, but prints garbage	f482630	parthpanchmatia <psp2137@columbia.edu>
Merged parser	378936a	parthpanchmatia <psp2137@columbia.edu>
Declare string is working	66473d9	parthpanchmatia <psp2137@columbia.edu>
Remove extra line from parser	5a04d4f	Sagar Damani <sagardmni@gmail.com>
Working make	73ad6cd	parthpanchmatia <psp2137@columbia.edu>
error-free helloworld	52237b9	dhruvshekhawat <dhruvbits315@gmail.com>
added prof's makefile, readme, and tests	bdaaaca	Parth Panchmatia <psp2137@columbia.edu>
Modified scanner and ast	a0f8336	Parth Panchmatia <psp2137@columbia.edu>
init codegen	57be8df	Akshay Nagpal <akshay2626@gmail.com>
Removed function names from Scanner	6990178	Parth Panchmatia <psp2137@columbia.edu>
removed pretty functions	ed6f792	dhruvshekhawat <dhruvbits315@gmail.com>
added LRM and Proposal	ef122b6	Akshay Nagpal <akshay2626@gmail.com>
understood block	5c6368e	dhruvshekhawat <dhruvbits315@gmail.com>
removed for	56963b6	dhruvshekhawat <dhruvbits315@gmail.com>
added functions	903b197	dhruvshekhawat <dhruvbits315@gmail.com>
removed for loop	7f7fa1c	dhruvshekhawat <dhruvbits315@gmail.com>
added ast matchings	01b62b7	dhruvshekhawat <dhruvbits315@gmail.com>
added layout	ce1cad3	dhruvshekhawat <dhruvbits315@gmail.com>
added keywords to scanner	c8e2ba6	dhruvshekhawat <dhruvbits315@gmail.com>
initial scanner	8c2fc2e	dhruvshekhawat <dhruvbits315@gmail.com>
Initial commit	13e953d	Dhruv Shekhawat <dhruvbits315@gmail.com>

5 Architectural Design

5.1 Components of Translator



The GRIDLang Compiler is written purely in OCaml. The compiler will convert the GRID source code into an LLVM intermediate representation (IR) that will eventually be translated to machine code by the LLVM backend. Our language can be run on various platforms as LLVM guarantees machine independence. Let us understand each component within the architecture:

5.1.1 Preprocessor

This module receives the .grid file and modifies it before the sending it to the scanner. It adds certain programming declarations to the code in order to facilitate succinct code writing for the developer. The preprocessor appends the following content:

- Libraries imported by the programmer
- GenericPiece struct with pointers to all the Piece structs defined by the programmer in order to facilitate storage of these items on the grid without having the programmer to explicity write it. Additionally, GenericPiece will have a typetag field indicating the type of Piece that has been filled in, and a nametag indicating the displayString of that Piece. Each GenericPiece will also have a location (denoted by int x and int y) on the grid.
- TriggerRule() function that maps a typetag to its respective rule() function
- Preprocessing appends the name of a Piece to its rule. So a rule function inside a Piece Pawn would be processed into a function Pawnrule().

- Default Piece attribute displayString to the Piece struct
- A default rule block within each Piece struct, which would approve every move, in case the game does not need move validation

5.1.2 Scanner

This module receives the preprocessed .grid file. It tokenizes every word in the file and rejects some wrong programs if they don't meet the criteria. Some of our language specific tokens are as follows:

Grid
Player
Piece
Grid_Init

5.1.3 Parser

The parser generates an AST from the tokenized input it receives. Other than the primitive data types the Parser deals with the following:

Structs
1D Arrays
2D Arrays
Multiple variable declarations on a single line

5.1.4 Semantic Checker

The AST generated by the parser undergoes semantic checking in this module. The semantic checker validates every statement in the code. We have taken care of the following semantic issues:

- Reports duplicates for:
  - Global variables
  - Local variables
  - Functions
  - Function Arguments
  - Structures
- Assignment operations: Checks that the type of the left and right side of an assignment are same
- Missing declarations:
  - Global variables
  - Functions
  - Structures
- Missing fields within a struct
- Illegal void globals
- Error for in-built function re-declaration
- In-built function re-declaration
- Force certain functions to be defined

5.1.5 Codegen

Once semantic checking has passed, Code Generation receives the AST generated by the parser, and generates the corresponding LLVM IR. Pattern matching is utilized to match the node in the AST to the LLVM code that is required to be generated for it.

6 Test Plan

6.1 Representative Programs

Chess chess.grid

```
import gridBasics.grid

int flag;
int checkGameEnd()
{
  /*First trigger checkIfUnderAttack for current location of opposing king.
  Then, trigger rule for king to see if he can move anywhere.
  But triggerRule() takes a destination coordinate.*/

  Player* p;
  Piece GenericPiece* kp;
  int dst_x, dst_y, tempx, tempy;

  /*Temporarily change current player*/
  nextPlayer();

  if (currentPlayerIndex == 0)
  {
    kp = getPieceFromGrid(p1.king.displayString);
  }
  else
  {
    kp = getPieceFromGrid(p2.king.displayString);
  }

  if (checkIfUnderAttack(kp.x, kp.y))
  {
    print("Currently under check");

    flag = 1;
    for(dst_x = 0; dst_x < rows; dst_x = dst_x + 1)
    {
      for(dst_y = 0; dst_y < cols; dst_y = dst_y + 1)
      {
        if (triggerRule(kp.x, kp.y, dst_x, dst_y, kp.typetag) == 1)
        {
          print("Can move to: ");
          print(dst_x);
          print(dst_y);
          nextPlayer();
          flag = 0;
          return 0;
        }
```

```
      }
    }
    flag = 0;
    printGrid();
    print("Checkmate!");
    return 1;
  }
  /*Change back*/
  nextPlayer();
  return 0;
}

int attx;
int atty;
bool checkIfUnderAttack(int dst_x, int dst_y)
{
  Player* p;
  int x, y;
  Piece GenericPiece* l;

  /*Switch players to see if any of enemy's items are attacking the king*/
  nextPlayer();
  if (currentPlayerIndex == 0)
  {
    p = &p1;
  }
  else
  {
    p = &p2;
  }

  for(x = 0; x < rows; x = x+1)
  {
    for(y = 0; y < cols; y = y+1)
    {
      l = getPieceAtLocation(x,y);
      if (l != None)
      {
        if (l.owner == p)
        {
          if (triggerRule(x, y, dst_x, dst_y, l.typetag) == 1)
          {
            attx = x;
            atty = y;
            nextPlayer();
            return true;
          }
        }
      }
    }
  }
  /*Switch back*/
  nextPlayer();
  return false;
```

```
}

Piece King
{
  string displayString;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    Piece GenericPiece* src, dst;
    Player* owner1, owner2;
    Player cur;
    int xdiff, ydiff;

    src = getPieceAtLocation(src_x, src_y);
    dst = getPieceAtLocation(dst_x, dst_y);

    /*Check if your piece*/
    cur = playerOrder[currentPlayerIndex];
    owner1 = src.owner;
    if (cur.color != owner1.color)
    {
      return 0;
    }

    xdiff = abs(dst_x - src_x);
    ydiff = abs(dst_y - src_y);
    if (xdiff > 1 || ydiff > 1)
    {
      return 0;
    }

    if (dst != None)
    {
      owner2 = dst.owner;
      if(owner2.color == owner1.color)
      {
        return 0;
      }
    }

    if (checkIfUnderAttack(dst_x, dst_y))
    {
      return 0;
    }
    return 1;
  }
}

Piece Pawn
{
  string displayString;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    int direction;
    Piece GenericPiece* src, dst, king_item;
```

```
Player* owner1, owner2;
Player cur;

src = getPieceAtLocation(src_x, src_y);
dst = getPieceAtLocation(dst_x, dst_y);

/*Check if your piece*/
cur = playerOrder[currentPlayerIndex];
owner1 = src.owner;
if (cur.color != owner1.color)
{
  return 0;
}

direction = getDirectionFromIndex();

if(dst_x != src_x + direction)
{
  return 0;
}
if (dst_y - src_y > 1 || dst_y - src_y < -1)
{
  return 0;
}
if(dst_y == src_y - 1 || dst_y == src_y + 1)
{
  if (dst != None)
  {
    owner2 = dst.owner;
    if (owner1.color == owner2.color && flag!=1)
    {
      return 0;
    }
    else
    {
      return 1;
    }
  }
  else
  {
    return 0;
  }
}
if (dst!=None)
{
  return 0;
}


if (currentPlayerIndex == 0)
{
  king_item = getPieceFromGrid(p1.king.displayString);
}
else
```

```
    {
      king_item = getPieceFromGrid(p2.king.displayString);
    }
    if (checkIfUnderAttack(king_item.x, king_item.y))
    {
      return 0;
    }

    return 1;
  }
}

Piece Bishop
{
  string displayString;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    Piece GenericPiece* src, dst, king_item;
    Player* owner1, owner2;
    Player cur;
    src = getPieceAtLocation(src_x, src_y);
    dst = getPieceAtLocation(dst_x, dst_y);

    /*Check if your piece*/
    cur = playerOrder[currentPlayerIndex];
    owner1 = src.owner;
    if (cur.color != owner1.color)
    {
      return 0;
    }

    if(abs(dst_x - src_x) == abs(dst_y - src_y))  /* moves diagonally only*/
    {
      if (dst != None)
      {
        owner2 = dst.owner;
        if(owner2.color == owner1.color && flag!=1)
        {
          return 0;
        }
      }

      if (traverse(src_x, src_y, dst_x, dst_y) == 1)
      {
        return 0;
      }

      if (currentPlayerIndex == 0)
      {
        king_item = getPieceFromGrid(p1.king.displayString);
      }
      else
      {
        king_item = getPieceFromGrid(p2.king.displayString);
```

```
      }
      if (checkIfUnderAttack(king_item.x, king_item.y))
      {
        return 0;
      }
      return 1;
    }
    return 0;
  }
}

Piece Knight
{
  string displayString;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    Piece GenericPiece* src, dst, king_item;
    Player* owner1, owner2;
    Player cur;
    int xdiff, ydiff;

    src = getPieceAtLocation(src_x, src_y);
    dst = getPieceAtLocation(dst_x, dst_y);

    /*Check if your piece*/
    cur = playerOrder[currentPlayerIndex];
    owner1 = src.owner;
    if (cur.color != owner1.color)
    {
      return 0;
    }

    xdiff = abs(dst_x - src_x);
    ydiff = abs(dst_y - src_y);
    if(xdiff == 2 && ydiff == 1 || xdiff == 1 && ydiff == 2)
    {
      if (dst!= None)
      {
        owner2 = dst.owner;
        if(owner2.color == owner1.color && flag!=1)
        {
          return 0;
        }
      }

      if (currentPlayerIndex == 0)
      {
        king_item = getPieceFromGrid(p1.king.displayString);
      }
      else
      {
        king_item = getPieceFromGrid(p2.king.displayString);
      }
      if (checkIfUnderAttack(king_item.x, king_item.y))
```

```
      {
        return 0;
      }
      return 1;
    }
    return 0;
  }
}

Piece Rook
{
  string displayString;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    Piece GenericPiece* src, dst, king_item;
    Player* owner1, owner2;
    Player cur;
    int xdiff, ydiff;

    src = getPieceAtLocation(src_x, src_y);
    dst = getPieceAtLocation(dst_x, dst_y);

    /*Check if your piece*/
    cur = playerOrder[currentPlayerIndex];
    owner1 = src.owner;
    if (cur.color != owner1.color)
    {
      return 0;
    }

    if(dst_x == src_x || dst_y == src_y)
    {
      if (dst != None)
      {
        owner2 = dst.owner;
        if(owner2.color == owner1.color && flag!=1)
        {
          return 0;
        }
      }
      /*Check if there is an obstacle in the way*/
      if (traverse(src_x, src_y, dst_x, dst_y) == 1)
      {
        return 0;
      }
      if (currentPlayerIndex == 0)
      {
        king_item = getPieceFromGrid(p1.king.displayString);
      }
      else
      {
        king_item = getPieceFromGrid(p2.king.displayString);
      }
      if (checkIfUnderAttack(king_item.x, king_item.y))
```

```
      {
        return 0;
      }
      return 1;
    }
    return 0;
  }
}

Player
{
  Piece Pawn pawn;
  Piece Bishop bishop;
  Piece Knight knight;
  Piece Rook rook;
  Piece King king;
  string color;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    return 1;
  }
}

int getDirectionFromIndex()
{
  int direction;
  if(currentPlayerIndex == 0)
  {
    direction = -1;
  }
  else
  {
    direction = 1;
  }
  return direction;
}

int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
  deleteFromGrid(x,y,i2.nametag);
  return 0;
}

Grid_Init<5,4>;
Player p1, p2;
Player[2] playerOrder;
int count;

int setup(){
  p1.color = "White";
  p2.color = "Black";
  playerOrder[0] = p1;
  playerOrder[1] = p2;
  flag = 0;
```

```
  p1.king.displayString = "W-King";
  p1.rook.displayString = "W-Rook";
  p1.knight.displayString = "W-Knight";
  p1.bishop.displayString = "W-Bishop";
  p1.pawn.displayString = "W-Pawn";

  p2.king.displayString = "B-King";
  p2.rook.displayString = "B-Rook";
  p2.knight.displayString = "B-Knight";
  p2.bishop.displayString = "B-Bishop";
  p2.pawn.displayString = "B-Pawn";

  Grid <4,3> <-- p1.king;
  Grid <3,3> <-- p1.pawn;
  Grid <4,1> <-- p1.bishop;
  Grid <4,2> <-- p1.knight;
  Grid <4,0> <-- p1.rook;

  Grid <0,0> <-- p2.king;
  Grid <1,0> <-- p2.pawn;
  Grid <0,2> <-- p2.bishop;
  Grid <0,1> <-- p2.knight;
  Grid <0,3> <-- p2.rook;
  return 0;
}

int gameloop(){
  int src_x, src_y, dst_x, dst_y;
  Piece GenericPiece* headnode, p_n;
  Player cur;
  cur = playerOrder[currentPlayerIndex];
  printGrid();
  print(cur.color);
  src_x = prompt("Enter source x:");
  src_y = prompt("Enter source y:");
  dst_x = prompt("Enter destination x:");
  dst_y = prompt("Enter destination y:");
  p_n = getPieceAtLocation(src_x, src_y);
  if (moveOnGrid(p_n, dst_x,dst_y) != 1)
  {
    print("Invalid move");
  }
  return 0;
}
```

chess.ll

```
; ModuleID = 'GridLang'

%Player = type <{ i8*, %King, %Rook, %Knight, %Bishop, %Pawn }> King = type <{
%i8*, i1, i8*, i1 }> Rook = type <{ i8*, i1, i8*, i1 }> Knight = type <{ i8*,
%i1, i8*, i1 }> Bishop = type <{ i8*, i1, i8*, i1 }> Pawn = type <{ i8*, i1,
%i8*, i1 }> GenericPiece = type <{ %Player*, i8*, i8*, %GenericPiece*, i32, i32,
%%Rook*, %Knight*, %Bishop*, %Pawn*, %King* }>
```

```
@count = global i32 0 @playerOrder = global [2 x %Player] zeroinitializer @p2 =
global %Player zeroinitializer @p1 = global %Player zeroinitializer @rows =
global i32 5 @cols = global i32 4 @GridData = global [5 x [4 x %GenericPiece*]]
zeroinitializer @atty = global i32 0 @attx = global i32 0 @flag = global i32 0
@playerOrderSize = global i32 0 @currentPlayerIndex = global i32 0 @fmt =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt1 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt2 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt3 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt4 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt5 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt6 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt7 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt8 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt9 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt10 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt11 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt12 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt13 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt14 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @name =
private unnamed_addr constant [5 x i8] c"King\00" @name15 = private unnamed_addr
constant [5 x i8] c"Pawn\00" @name16 = private unnamed_addr constant [7 x i8]
c"Bishop\00" @name17 = private unnamed_addr constant [7 x i8] c"Knight\00"
@name18 = private unnamed_addr constant [5 x i8] c"Rook\00" @fmt19 = private
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt20 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt21 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt22 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt23 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt24 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt25 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt26 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt27 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt28 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt29 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt30 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @name31
= private unnamed_addr constant [17 x i8] c"No piece on cell\00" @name32 =
private unnamed_addr constant [17 x i8] c"No piece on cell\00" @fmt33 = private
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt34 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @name35 = private unnamed_addr constant [2 x i8]
c"_\00" @name36 = private unnamed_addr constant [2 x i8] c"_\00" @name37 =
private unnamed_addr constant [2 x i8] c"_\00" @name38 = private unnamed_addr
constant [2 x i8] c"|\00" @name39 = private unnamed_addr constant [3 x i8] c",
\00" @name40 = private unnamed_addr constant [2 x i8] c" \00" @name41 = private
unnamed_addr constant [2 x i8] c"|\00" @name42 = private unnamed_addr constant
[2 x i8] c"-\00" @fmt43 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@mmt44 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @name45 = private
unnamed_addr constant [30 x i8] c"Not found on given coordinate\00" @name46 =
private unnamed_addr constant [30 x i8] c"Not found on given coordinate\00"
@fmt47 = private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt48 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt49 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt50 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @name51 = private unnamed_addr constant [16 x i8] c"Enter source
x:\00" @name52 = private unnamed_addr constant [16 x i8] c"Enter source y:\00"
@name53 = private unnamed_addr constant [21 x i8] c"Enter destination x:\00"
@name54 = private unnamed_addr constant [21 x i8] c"Enter destination y:\00"
@name55 = private unnamed_addr constant [13 x i8] c"Invalid move\00" @name56 =
private unnamed_addr constant [13 x i8] c"Invalid move\00" @fmt57 = private
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt58 = private unnamed_addr
```

```
constant [4 x i8] c"%s\0A\00" @name59 = private unnamed_addr constant [6 x i8]
c"White\00" @name60 = private unnamed_addr constant [6 x i8] c"Black\00" @name61
= private unnamed_addr constant [7 x i8] c"W-King\00" @name62 = private
unnamed_addr constant [7 x i8] c"W-Rook\00" @name63 = private unnamed_addr
constant [9 x i8] c"W-Knight\00" @name64 = private unnamed_addr constant [9 x
i8] c"W-Bishop\00" @name65 = private unnamed_addr constant [7 x i8] c"W-Pawn\00"
@name66 = private unnamed_addr constant [7 x i8] c"B-King\00" @name67 = private
unnamed_addr constant [7 x i8] c"B-Rook\00" @name68 = private unnamed_addr
constant [9 x i8] c"B-Knight\00" @name69 = private unnamed_addr constant [9 x
i8] c"B-Bishop\00" @name70 = private unnamed_addr constant [7 x i8] c"B-Pawn\00"
@name71 = private unnamed_addr constant [5 x i8] c"King\00" @name72 = private
unnamed_addr constant [5 x i8] c"Pawn\00" @name73 = private unnamed_addr
constant [7 x i8] c"Bishop\00" @name74 = private unnamed_addr constant [7 x i8]
c"Knight\00" @name75 = private unnamed_addr constant [5 x i8] c"Rook\00" @name76
= private unnamed_addr constant [5 x i8] c"King\00" @name77 = private
unnamed_addr constant [5 x i8] c"Pawn\00" @name78 = private unnamed_addr
constant [7 x i8] c"Bishop\00" @name79 = private unnamed_addr constant [7 x i8]
c"Knight\00" @name80 = private unnamed_addr constant [5 x i8] c"Rook\00" @fmt81
= private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt82 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt83 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt84 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @fmt85 = private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt86 =
private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt87 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt88 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @name89 = private unnamed_addr constant [22 x i8] c"Currently under
check\00" @name90 = private unnamed_addr constant [22 x i8] c"Currently under
check\00" @name91 = private unnamed_addr constant [14 x i8] c"Can move to: \00"
@name92 = private unnamed_addr constant [14 x i8] c"Can move to: \00" @name93 =
private unnamed_addr constant [11 x i8] c"Checkmate!\00" @name94 = private
unnamed_addr constant [11 x i8] c"Checkmate!\00"

declare i32 @printf(i8*, ...)

declare i32 @prompt(i8*)

declare i32 @abs(i32)

declare i32 @print_endline()

declare i32 @print_sameline(i8*)

declare i32 @diceThrow()

declare i32 @getLen(i8*)

declare i32 @print_int_sameline(i32)

define i32 @Playerrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
```

```
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @Rookrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %src = alloca %GenericPiece* dst = alloca %GenericPiece* king_item = alloca
  %%GenericPiece* owner1 = alloca %Player* owner2 = alloca %Player* cur = alloca
  %%Player xdiff = alloca i32 ydiff = alloca i32 repeat = alloca i32 src_y5 =
  %load i32* %src_y2 src_x6 = load i32* %src_x1 getPieceAtLocation_result = call
  %%GenericPiece* @getPieceAtLocation(i32 %src_x6, i32 %src_y5)
  store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %src
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3
  %getPieceAtLocation_result9 = call %GenericPiece* @getPieceAtLocation(i32
  %%dst_x8, i32 %dst_y7)
  store %GenericPiece* %getPieceAtLocation_result9, %GenericPiece** %dst
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name10 =
  %load %Player* %name
  store %Player %name10, %Player* %cur
  %src11 = load %GenericPiece** %src loaded_deref = load %GenericPiece* %src11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %src11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal
  store %Player* %loaded_dotop_terminal, %Player** %owner1
  %cur12 = load %Player* %cur dotop_terminal13 = getelementptr inbounds %Player*
  %%cur, i32 0, i32 0 loaded_dotop_terminal14 = load i8** %dotop_terminal13
  %owner115 = load %Player** %owner1 loaded_deref16 = load %Player* %owner115
  %dotop_terminal17 = getelementptr inbounds %Player* %owner115, i32 0, i32 0
  %loaded_dotop_terminal18 = load i8** %dotop_terminal17 tmp = icmp ne i8*
  %%loaded_dotop_terminal14, %loaded_dotop_terminal18
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %dst_x19 = load i32* %dst_x3 src_x20 = load i32* %src_x1 tmp21 = icmp eq i32
  %%dst_x19, %src_x20 dst_y22 = load i32* %dst_y4 src_y23 = load i32* %src_y2
  %tmp24 = icmp eq i32 %dst_y22, %src_y23 tmp25 = or i1 %tmp21, %tmp24
  br i1 %tmp25, label %then27, label %else83

then:                                             ; preds = %entry ret i32 0

else:                                             ; preds = %entry br label
%merge

merge26:                                          ; preds = %else83 ret i32 0

then27:                                           ; preds = %merge
```

```
  %dst28 = load %GenericPiece** %dst tmp29 = icmp ne %GenericPiece* %dst28, null
  br i1 %tmp29, label %then31, label %else50

merge30:                                          ; preds = %else50, %merge47
  %dst_y51 = load i32* %dst_y4 dst_x52 = load i32* %dst_x3 src_y53 = load i32*
  %%src_y2 src_x54 = load i32* %src_x1 traverse_result = call i32 @traverse(i32
  %%src_x54, i32 %src_y53, i32 %dst_x52, i32 %dst_y51) tmp55 = icmp eq i32
  %%traverse_result, 1
  br i1 %tmp55, label %then57, label %else58

then31:                                           ; preds = %then27
  %dst32 = load %GenericPiece** %dst loaded_deref33 = load %GenericPiece* %dst32
  %dotop_terminal34 = getelementptr inbounds %GenericPiece* %dst32, i32 0, i32 0
  %loaded_dotop_terminal35 = load %Player** %dotop_terminal34
  store %Player* %loaded_dotop_terminal35, %Player** %owner2
  %owner236 = load %Player** %owner2 loaded_deref37 = load %Player* %owner236
  %dotop_terminal38 = getelementptr inbounds %Player* %owner236, i32 0, i32 0
  %loaded_dotop_terminal39 = load i8** %dotop_terminal38 owner140 = load
  %%Player** %owner1 loaded_deref41 = load %Player* %owner140 dotop_terminal42 =
  %getelementptr inbounds %Player* %owner140, i32 0, i32 0
  %loaded_dotop_terminal43 = load i8** %dotop_terminal42 tmp44 = icmp eq i8*
  %%loaded_dotop_terminal39, %loaded_dotop_terminal43 flag = load i32* @flag
  %tmp45 = icmp ne i32 %flag, 1 tmp46 = and i1 %tmp44, %tmp45
  br i1 %tmp46, label %then48, label %else49

merge47:                                          ; preds = %else49 br label
%merge30

then48:                                           ; preds = %then31 ret i32 0

else49:                                           ; preds = %then31 br label
%merge47

else50:                                           ; preds = %then27 br label
%merge30

merge56:                                          ; preds = %else58
  %currentPlayerIndex59 = load i32* @currentPlayerIndex tmp60 = icmp eq i32
  %%currentPlayerIndex59, 0
  br i1 %tmp60, label %then62, label %else66

then57:                                           ; preds = %merge30 ret i32 0

else58:                                           ; preds = %merge30 br label
%merge56

merge61:                                          ; preds = %else66, %then62
  %king_item72 = load %GenericPiece** %king_item loaded_deref73 = load
  %%GenericPiece* %king_item72 dotop_terminal74 = getelementptr inbounds
  %%GenericPiece* %king_item72, i32 0, i32 4 loaded_dotop_terminal75 = load i32*
  %%dotop_terminal74 king_item76 = load %GenericPiece** %king_item
  %loaded_deref77 = load %GenericPiece* %king_item76 dotop_terminal78 =
  %getelementptr inbounds %GenericPiece* %king_item76, i32 0, i32 5
  %loaded_dotop_terminal79 = load i32* %dotop_terminal78
```

```
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32
  %%loaded_dotop_terminal79, i32 %loaded_dotop_terminal75)
  br i1 %checkIfUnderAttack_result, label %then81, label %else82

then62:                                           ; preds = %merge56
  %p1 = load %Player* @p1 loaded_dotop_terminal63 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p164 = load %Player* @p1
  %loaded_dotop_terminal65 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop)
  store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %king_item br
label %merge61

else66:                                           ; preds = %merge56
  %p2 = load %Player* @p2 loaded_dotop_terminal67 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p268 = load %Player* @p2
  %loaded_dotop_terminal69 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop70 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 1, i32 2) getPieceFromGrid_result71 = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop70)
  store %GenericPiece* %getPieceFromGrid_result71, %GenericPiece** %king_item br
label %merge61

merge80:                                          ; preds = %else82 ret i32 1

then81:                                           ; preds = %merge61 ret i32 0

else82:                                           ; preds = %merge61 br label
%merge80

else83:                                           ; preds = %merge br label
%merge26 }

define i32 @Knightrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %src = alloca %GenericPiece* dst = alloca %GenericPiece* king_item = alloca
  %%GenericPiece* owner1 = alloca %Player* owner2 = alloca %Player* cur = alloca
  %%Player xdiff = alloca i32 ydiff = alloca i32 repeat = alloca i32 src_y5 =
  %load i32* %src_y2 src_x6 = load i32* %src_x1 getPieceAtLocation_result = call
  %%GenericPiece* @getPieceAtLocation(i32 %src_x6, i32 %src_y5)
  store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %src
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3
  %getPieceAtLocation_result9 = call %GenericPiece* @getPieceAtLocation(i32
  %%dst_x8, i32 %dst_y7)
  store %GenericPiece* %getPieceAtLocation_result9, %GenericPiece** %dst
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
```

```
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name10 =
  %load %Player* %name
   store %Player %name10, %Player* %cur
  %src11 = load %GenericPiece** %src loaded_deref = load %GenericPiece* %src11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %src11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal
   store %Player* %loaded_dotop_terminal, %Player** %owner1
  %cur12 = load %Player* %cur dotop_terminal13 = getelementptr inbounds %Player*
  %%cur, i32 0, i32 0 loaded_dotop_terminal14 = load i8** %dotop_terminal13
  %owner115 = load %Player** %owner1 loaded_deref16 = load %Player* %owner115
  %dotop_terminal17 = getelementptr inbounds %Player* %owner115, i32 0, i32 0
  %loaded_dotop_terminal18 = load i8** %dotop_terminal17 tmp = icmp ne i8*
  %%loaded_dotop_terminal14, %loaded_dotop_terminal18
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %dst_x19 = load i32* %dst_x3 src_x20 = load i32* %src_x1 tmp21 = sub i32
  %%dst_x19, %src_x20 absl = call i32 @abs(i32 %tmp21)
   store i32 %absl, i32* %xdiff
  %dst_y22 = load i32* %dst_y4 src_y23 = load i32* %src_y2 tmp24 = sub i32
  %%dst_y22, %src_y23 absl25 = call i32 @abs(i32 %tmp24)
   store i32 %absl25, i32* %ydiff
  %xdiff26 = load i32* %xdiff tmp27 = icmp eq i32 %xdiff26, 2 ydiff28 = load
  %i32* %ydiff tmp29 = icmp eq i32 %ydiff28, 1 tmp30 = and i1 %tmp27, %tmp29
  %xdiff31 = load i32* %xdiff tmp32 = icmp eq i32 %xdiff31, 1 ydiff33 = load
  %i32* %ydiff tmp34 = icmp eq i32 %ydiff33, 2 tmp35 = and i1 %tmp32, %tmp34
  %tmp36 = or i1 %tmp30, %tmp35
   br i1 %tmp36, label %then38, label %else86

then:                                             ; preds = %entry ret i32 0

else:                                             ; preds = %entry br label
%merge

merge37:                                          ; preds = %else86 ret i32 0

then38:                                           ; preds = %merge
  %dst39 = load %GenericPiece** %dst tmp40 = icmp ne %GenericPiece* %dst39, null
   br i1 %tmp40, label %then42, label %else61

merge41:                                          ; preds = %else61, %merge58
  %currentPlayerIndex62 = load i32* @currentPlayerIndex tmp63 = icmp eq i32
  %%currentPlayerIndex62, 0
   br i1 %tmp63, label %then65, label %else69

then42:                                           ; preds = %then38
  %dst43 = load %GenericPiece** %dst loaded_deref44 = load %GenericPiece* %dst43
  %dotop_terminal45 = getelementptr inbounds %GenericPiece* %dst43, i32 0, i32 0
  %loaded_dotop_terminal46 = load %Player** %dotop_terminal45
   store %Player* %loaded_dotop_terminal46, %Player** %owner2
  %owner247 = load %Player** %owner2 loaded_deref48 = load %Player* %owner247
  %dotop_terminal49 = getelementptr inbounds %Player* %owner247, i32 0, i32 0
  %loaded_dotop_terminal50 = load i8** %dotop_terminal49 owner151 = load
  %%Player** %owner1 loaded_deref52 = load %Player* %owner151 dotop_terminal53 =
```

```
  %getelementptr inbounds %Player* %owner151, i32 0, i32 0
  %loaded_dotop_terminal54 = load i8** %dotop_terminal53 tmp55 = icmp eq i8*
  %%loaded_dotop_terminal50, %loaded_dotop_terminal54 flag = load i32* @flag
  %tmp56 = icmp ne i32 %flag, 1 tmp57 = and i1 %tmp55, %tmp56
   br i1 %tmp57, label %then59, label %else60

merge58:                                          ; preds = %else60 br label
%merge41

then59:                                           ; preds = %then42 ret i32 0

else60:                                           ; preds = %then42 br label
%merge58

else61:                                           ; preds = %then38 br label
%merge41

merge64:                                          ; preds = %else69, %then65
  %king_item75 = load %GenericPiece** %king_item loaded_deref76 = load
  %%GenericPiece* %king_item75 dotop_terminal77 = getelementptr inbounds
  %%GenericPiece* %king_item75, i32 0, i32 4 loaded_dotop_terminal78 = load i32*
  %%dotop_terminal77 king_item79 = load %GenericPiece** %king_item
  %loaded_deref80 = load %GenericPiece* %king_item79 dotop_terminal81 =
  %getelementptr inbounds %GenericPiece* %king_item79, i32 0, i32 5
  %loaded_dotop_terminal82 = load i32* %dotop_terminal81
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32
  %%loaded_dotop_terminal82, i32 %loaded_dotop_terminal78)
   br i1 %checkIfUnderAttack_result, label %then84, label %else85

then65:                                           ; preds = %merge41
  %p1 = load %Player* @p1 loaded_dotop_terminal66 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p167 = load %Player* @p1
  %loaded_dotop_terminal68 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop)
   store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %king_item br
label %merge64

else69:                                           ; preds = %merge41
  %p2 = load %Player* @p2 loaded_dotop_terminal70 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p271 = load %Player* @p2
  %loaded_dotop_terminal72 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop73 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 1, i32 2) getPieceFromGrid_result74 = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop73)
   store %GenericPiece* %getPieceFromGrid_result74, %GenericPiece** %king_item br
label %merge64

merge83:                                          ; preds = %else85 ret i32 1

then84:                                           ; preds = %merge64 ret i32 0

else85:                                           ; preds = %merge64 br label
```

```
%merge83

else86:                                           ; preds = %merge br label
%merge37 }

define i32 @Bishoprule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %src = alloca %GenericPiece* dst = alloca %GenericPiece* king_item = alloca
  %%GenericPiece* owner1 = alloca %Player* owner2 = alloca %Player* cur = alloca
  %%Player repeat = alloca i32 src_y5 = load i32* %src_y2 src_x6 = load i32*
  %%src_x1 getPieceAtLocation_result = call %GenericPiece*
  %@getPieceAtLocation(i32 %src_x6, i32 %src_y5)
  store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %src
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3
  %getPieceAtLocation_result9 = call %GenericPiece* @getPieceAtLocation(i32
  %%dst_x8, i32 %dst_y7)
  store %GenericPiece* %getPieceAtLocation_result9, %GenericPiece** %dst
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name10 =
  %load %Player* %name
  store %Player %name10, %Player* %cur
  %src11 = load %GenericPiece** %src loaded_deref = load %GenericPiece* %src11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %src11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal
  store %Player* %loaded_dotop_terminal, %Player** %owner1
  %cur12 = load %Player* %cur dotop_terminal13 = getelementptr inbounds %Player*
  %%cur, i32 0, i32 0 loaded_dotop_terminal14 = load i8** %dotop_terminal13
  %owner115 = load %Player** %owner1 loaded_deref16 = load %Player* %owner115
  %dotop_terminal17 = getelementptr inbounds %Player* %owner115, i32 0, i32 0
  %loaded_dotop_terminal18 = load i8** %dotop_terminal17 tmp = icmp ne i8*
  %%loaded_dotop_terminal14, %loaded_dotop_terminal18
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %dst_x19 = load i32* %dst_x3 src_x20 = load i32* %src_x1 tmp21 = sub i32
  %%dst_x19, %src_x20 absl = call i32 @abs(i32 %tmp21) dst_y22 = load i32*
  %%dst_y4 src_y23 = load i32* %src_y2 tmp24 = sub i32 %dst_y22, %src_y23 absl25
  %= call i32 @abs(i32 %tmp24) tmp26 = icmp eq i32 %absl, %absl25
  br i1 %tmp26, label %then28, label %else84

then:                                             ; preds = %entry ret i32 0

else:                                             ; preds = %entry br label
%merge

merge27:                                          ; preds = %else84 ret i32 0
```

```
then28:                                           ; preds = %merge
  %dst29 = load %GenericPiece** %dst tmp30 = icmp ne %GenericPiece* %dst29, null
  br i1 %tmp30, label %then32, label %else51

merge31:                                          ; preds = %else51, %merge48
  %dst_y52 = load i32* %dst_y4 dst_x53 = load i32* %dst_x3 src_y54 = load i32*
  %%src_y2 src_x55 = load i32* %src_x1 traverse_result = call i32 @traverse(i32
  %%src_x55, i32 %src_y54, i32 %dst_x53, i32 %dst_y52) tmp56 = icmp eq i32
  %%traverse_result, 1
  br i1 %tmp56, label %then58, label %else59

then32:                                           ; preds = %then28
  %dst33 = load %GenericPiece** %dst loaded_deref34 = load %GenericPiece* %dst33
  %dotop_terminal35 = getelementptr inbounds %GenericPiece* %dst33, i32 0, i32 0
  %loaded_dotop_terminal36 = load %Player** %dotop_terminal35
  store %Player* %loaded_dotop_terminal36, %Player** %owner2
  %owner237 = load %Player** %owner2 loaded_deref38 = load %Player* %owner237
  %dotop_terminal39 = getelementptr inbounds %Player* %owner237, i32 0, i32 0
  %loaded_dotop_terminal40 = load i8** %dotop_terminal39 owner141 = load
  %%Player** %owner1 loaded_deref42 = load %Player* %owner141 dotop_terminal43 =
  %getelementptr inbounds %Player* %owner141, i32 0, i32 0
  %loaded_dotop_terminal44 = load i8** %dotop_terminal43 tmp45 = icmp eq i8*
  %%loaded_dotop_terminal40, %loaded_dotop_terminal44 flag = load i32* @flag
  %tmp46 = icmp ne i32 %flag, 1 tmp47 = and i1 %tmp45, %tmp46
  br i1 %tmp47, label %then49, label %else50

merge48:                                          ; preds = %else50 br label
%merge31

then49:                                           ; preds = %then32 ret i32 0

else50:                                           ; preds = %then32 br label
%merge48

else51:                                           ; preds = %then28 br label
%merge31

merge57:                                          ; preds = %else59
  %currentPlayerIndex60 = load i32* @currentPlayerIndex tmp61 = icmp eq i32
  %%currentPlayerIndex60, 0
  br i1 %tmp61, label %then63, label %else67

then58:                                           ; preds = %merge31 ret i32 0

else59:                                           ; preds = %merge31 br label
%merge57

merge62:                                          ; preds = %else67, %then63
  %king_item73 = load %GenericPiece** %king_item loaded_deref74 = load
  %%GenericPiece* %king_item73 dotop_terminal75 = getelementptr inbounds
  %%GenericPiece* %king_item73, i32 0, i32 4 loaded_dotop_terminal76 = load i32*
  %%dotop_terminal75 king_item77 = load %GenericPiece** %king_item
  %loaded_deref78 = load %GenericPiece* %king_item77 dotop_terminal79 =
  %getelementptr inbounds %GenericPiece* %king_item77, i32 0, i32 5
```

```
  %loaded_dotop_terminal80 = load i32* %dotop_terminal79
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32
  %%loaded_dotop_terminal80, i32 %loaded_dotop_terminal76)
   br i1 %checkIfUnderAttack_result, label %then82, label %else83

then63:                                           ; preds = %merge57
  %p1 = load %Player* @p1 loaded_dotop_terminal64 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p165 = load %Player* @p1
  %loaded_dotop_terminal66 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop)
   store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %king_item br
label %merge62

else67:                                           ; preds = %merge57
  %p2 = load %Player* @p2 loaded_dotop_terminal68 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p269 = load %Player* @p2
  %loaded_dotop_terminal70 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop71 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 1, i32 2) getPieceFromGrid_result72 = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop71)
   store %GenericPiece* %getPieceFromGrid_result72, %GenericPiece** %king_item br
label %merge62

merge81:                                          ; preds = %else83 ret i32 1

then82:                                           ; preds = %merge62 ret i32 0

else83:                                           ; preds = %merge62 br label
%merge81

else84:                                           ; preds = %merge br label
%merge27 }

define i32 @Pawnrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
   store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
   store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
   store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
   store i32 %dst_y, i32* %dst_y4
  %direction = alloca i32 src = alloca %GenericPiece* dst = alloca
  %%GenericPiece* king_item = alloca %GenericPiece* owner1 = alloca %Player*
  %owner2 = alloca %Player* cur = alloca %Player repeat = alloca i32 src_y5 =
  %load i32* %src_y2 src_x6 = load i32* %src_x1 getPieceAtLocation_result = call
  %%GenericPiece* @getPieceAtLocation(i32 %src_x6, i32 %src_y5)
   store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %src
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3
  %getPieceAtLocation_result9 = call %GenericPiece* @getPieceAtLocation(i32
  %%dst_x8, i32 %dst_y7)
   store %GenericPiece* %getPieceAtLocation_result9, %GenericPiece** %dst
```

```
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name10 =
  %load %Player* %name
   store %Player %name10, %Player* %cur
  %src11 = load %GenericPiece** %src loaded_deref = load %GenericPiece* %src11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %src11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal
   store %Player* %loaded_dotop_terminal, %Player** %owner1
  %cur12 = load %Player* %cur dotop_terminal13 = getelementptr inbounds %Player*
  %%cur, i32 0, i32 0 loaded_dotop_terminal14 = load i8** %dotop_terminal13
  %owner115 = load %Player** %owner1 loaded_deref16 = load %Player* %owner115
  %dotop_terminal17 = getelementptr inbounds %Player* %owner115, i32 0, i32 0
  %loaded_dotop_terminal18 = load i8** %dotop_terminal17 tmp = icmp ne i8*
  %%loaded_dotop_terminal14, %loaded_dotop_terminal18
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %getDirectionFromIndex_result = call i32 @getDirectionFromIndex()
   store i32 %getDirectionFromIndex_result, i32* %direction
  %dst_x19 = load i32* %dst_x3 src_x20 = load i32* %src_x1 direction21 = load
  %i32* %direction tmp22 = add i32 %src_x20, %direction21 tmp23 = icmp ne i32
  %%dst_x19, %tmp22
   br i1 %tmp23, label %then25, label %else26

then:                                             ; preds = %entry ret i32 0

else:                                             ; preds = %entry br label
%merge

merge24:                                          ; preds = %else26
  %dst_y27 = load i32* %dst_y4 src_y28 = load i32* %src_y2 tmp29 = sub i32
  %%dst_y27, %src_y28 tmp30 = icmp sgt i32 %tmp29, 1 dst_y31 = load i32* %dst_y4
  %src_y32 = load i32* %src_y2 tmp33 = sub i32 %dst_y31, %src_y32 tmp34 = icmp
  %slt i32 %tmp33, -1 tmp35 = or i1 %tmp30, %tmp34
   br i1 %tmp35, label %then37, label %else38

then25:                                           ; preds = %merge ret i32 0

else26:                                           ; preds = %merge br label
%merge24

merge36:                                          ; preds = %else38
  %dst_y39 = load i32* %dst_y4 src_y40 = load i32* %src_y2 tmp41 = sub i32
  %%src_y40, 1 tmp42 = icmp eq i32 %dst_y39, %tmp41 dst_y43 = load i32* %dst_y4
  %src_y44 = load i32* %src_y2 tmp45 = add i32 %src_y44, 1 tmp46 = icmp eq i32
  %%dst_y43, %tmp45 tmp47 = or i1 %tmp42, %tmp46
   br i1 %tmp47, label %then49, label %else73

then37:                                           ; preds = %merge24 ret i32 0

else38:                                           ; preds = %merge24 br label
%merge36

merge48:                                          ; preds = %else73, %merge52
```

```
  %dst74 = load %GenericPiece** %dst tmp75 = icmp ne %GenericPiece* %dst74, null
  br i1 %tmp75, label %then77, label %else78

then49:                                           ; preds = %merge36
  %dst50 = load %GenericPiece** %dst tmp51 = icmp ne %GenericPiece* %dst50, null
  br i1 %tmp51, label %then53, label %else72

merge52:                                          ; preds = %merge69 br label
%merge48

then53:                                           ; preds = %then49
  %dst54 = load %GenericPiece** %dst loaded_deref55 = load %GenericPiece* %dst54
  %dotop_terminal56 = getelementptr inbounds %GenericPiece* %dst54, i32 0, i32 0
  %loaded_dotop_terminal57 = load %Player** %dotop_terminal56
  store %Player* %loaded_dotop_terminal57, %Player** %owner2
  %owner158 = load %Player** %owner1 loaded_deref59 = load %Player* %owner158
  %dotop_terminal60 = getelementptr inbounds %Player* %owner158, i32 0, i32 0
  %loaded_dotop_terminal61 = load i8** %dotop_terminal60 owner262 = load
  %%Player** %owner2 loaded_deref63 = load %Player* %owner262 dotop_terminal64 =
  %getelementptr inbounds %Player* %owner262, i32 0, i32 0
  %loaded_dotop_terminal65 = load i8** %dotop_terminal64 tmp66 = icmp eq i8*
  %%loaded_dotop_terminal61, %loaded_dotop_terminal65 flag = load i32* @flag
  %tmp67 = icmp ne i32 %flag, 1 tmp68 = and i1 %tmp66, %tmp67
  br i1 %tmp68, label %then70, label %else71

merge69:                                          ; No predecessors!  br label
%merge52

then70:                                           ; preds = %then53 ret i32 0

else71:                                           ; preds = %then53 ret i32 1

else72:                                           ; preds = %then49 ret i32 0

else73:                                           ; preds = %merge36 br label
%merge48

merge76:                                          ; preds = %else78
  %currentPlayerIndex79 = load i32* @currentPlayerIndex tmp80 = icmp eq i32
  %%currentPlayerIndex79, 0
  br i1 %tmp80, label %then82, label %else86

then77:                                           ; preds = %merge48 ret i32 0

else78:                                           ; preds = %merge48 br label
%merge76

merge81:                                          ; preds = %else86, %then82
  %king_item92 = load %GenericPiece** %king_item loaded_deref93 = load
  %%GenericPiece* %king_item92 dotop_terminal94 = getelementptr inbounds
  %%GenericPiece* %king_item92, i32 0, i32 4 loaded_dotop_terminal95 = load i32*
  %%dotop_terminal94 king_item96 = load %GenericPiece** %king_item
  %loaded_deref97 = load %GenericPiece* %king_item96 dotop_terminal98 =
  %getelementptr inbounds %GenericPiece* %king_item96, i32 0, i32 5
```

```
  %loaded_dotop_terminal99 = load i32* %dotop_terminal98
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32
  %%loaded_dotop_terminal99, i32 %loaded_dotop_terminal95)
   br i1 %checkIfUnderAttack_result, label %then101, label %else102

then82:                                           ; preds = %merge76
  %p1 = load %Player* @p1 loaded_dotop_terminal83 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p184 = load %Player* @p1
  %loaded_dotop_terminal85 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop)
   store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %king_item br
label %merge81

else86:                                           ; preds = %merge76
  %p2 = load %Player* @p2 loaded_dotop_terminal87 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p288 = load %Player* @p2
  %loaded_dotop_terminal89 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop90 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 1, i32 2) getPieceFromGrid_result91 = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop90)
   store %GenericPiece* %getPieceFromGrid_result91, %GenericPiece** %king_item br
label %merge81

merge100:                                         ; preds = %else102 ret i32 1

then101:                                          ; preds = %merge81 ret i32 0

else102:                                          ; preds = %merge81 br label
%merge100 }

define i32 @Kingrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
   store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
   store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
   store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
   store i32 %dst_y, i32* %dst_y4
  %src = alloca %GenericPiece* dst = alloca %GenericPiece* owner1 = alloca
  %%Player* owner2 = alloca %Player* cur = alloca %Player xdiff = alloca i32
  %ydiff = alloca i32 repeat = alloca i32 src_y5 = load i32* %src_y2 src_x6 =
  %load i32* %src_x1 getPieceAtLocation_result = call %GenericPiece*
  %@getPieceAtLocation(i32 %src_x6, i32 %src_y5)
   store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %src
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3
  %getPieceAtLocation_result9 = call %GenericPiece* @getPieceAtLocation(i32
  %%dst_x8, i32 %dst_y7)
   store %GenericPiece* %getPieceAtLocation_result9, %GenericPiece** %dst
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name10 =
  %load %Player* %name
```

```
  store %Player %name10, %Player* %cur
  %src11 = load %GenericPiece** %src loaded_deref = load %GenericPiece* %src11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %src11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal
  store %Player* %loaded_dotop_terminal, %Player** %owner1
  %cur12 = load %Player* %cur dotop_terminal13 = getelementptr inbounds %Player*
  %%cur, i32 0, i32 0 loaded_dotop_terminal14 = load i8** %dotop_terminal13
  %owner115 = load %Player** %owner1 loaded_deref16 = load %Player* %owner115
  %dotop_terminal17 = getelementptr inbounds %Player* %owner115, i32 0, i32 0
  %loaded_dotop_terminal18 = load i8** %dotop_terminal17 tmp = icmp ne i8*
  %%loaded_dotop_terminal14, %loaded_dotop_terminal18
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %dst_x19 = load i32* %dst_x3 src_x20 = load i32* %src_x1 tmp21 = sub i32
  %%dst_x19, %src_x20 absl = call i32 @abs(i32 %tmp21)
  store i32 %absl, i32* %xdiff
  %dst_y22 = load i32* %dst_y4 src_y23 = load i32* %src_y2 tmp24 = sub i32
  %%dst_y22, %src_y23 absl25 = call i32 @abs(i32 %tmp24)
  store i32 %absl25, i32* %ydiff
  %xdiff26 = load i32* %xdiff tmp27 = icmp sgt i32 %xdiff26, 1 ydiff28 = load
  %i32* %ydiff tmp29 = icmp sgt i32 %ydiff28, 1 tmp30 = or i1 %tmp27, %tmp29
  br i1 %tmp30, label %then32, label %else33

then:                                             ; preds = %entry ret i32 0

else:                                             ; preds = %entry br label
%merge

merge31:                                          ; preds = %else33
  %dst34 = load %GenericPiece** %dst tmp35 = icmp ne %GenericPiece* %dst34, null
  br i1 %tmp35, label %then37, label %else54

then32:                                           ; preds = %merge ret i32 0

else33:                                           ; preds = %merge br label
%merge31

merge36:                                          ; preds = %else54, %merge51
  %dst_y55 = load i32* %dst_y4 dst_x56 = load i32* %dst_x3
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32 %dst_x56, i32
  %%dst_y55)
  br i1 %checkIfUnderAttack_result, label %then58, label %else59

then37:                                           ; preds = %merge31
  %dst38 = load %GenericPiece** %dst loaded_deref39 = load %GenericPiece* %dst38
  %dotop_terminal40 = getelementptr inbounds %GenericPiece* %dst38, i32 0, i32 0
  %loaded_dotop_terminal41 = load %Player** %dotop_terminal40
  store %Player* %loaded_dotop_terminal41, %Player** %owner2
  %owner242 = load %Player** %owner2 loaded_deref43 = load %Player* %owner242
  %dotop_terminal44 = getelementptr inbounds %Player* %owner242, i32 0, i32 0
  %loaded_dotop_terminal45 = load i8** %dotop_terminal44 owner146 = load
  %%Player** %owner1 loaded_deref47 = load %Player* %owner146 dotop_terminal48 =
  %getelementptr inbounds %Player* %owner146, i32 0, i32 0
```

```
  %loaded_dotop_terminal49 = load i8** %dotop_terminal48 tmp50 = icmp eq i8*
  %%loaded_dotop_terminal45, %loaded_dotop_terminal49
  br i1 %tmp50, label %then52, label %else53

merge51:                                          ; preds = %else53 br label
%merge36

then52:                                           ; preds = %then37 ret i32 0

else53:                                           ; preds = %then37 br label
%merge51

else54:                                           ; preds = %merge31 br label
%merge36

merge57:                                          ; preds = %else59 ret i32 1

then58:                                           ; preds = %merge36 ret i32 0

else59:                                           ; preds = %merge36 br label
%merge57 }

define i32 @GenericPiecerule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) {
entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @triggerRule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y, i8*
%typetag) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %typetag5 = alloca i8*
  store i8* %typetag, i8** %typetag5
  %x = alloca i32 repeat = alloca i32 typetag6 = load i8** %typetag5 tmp = icmp
  %eq i8* %typetag6, getelementptr inbounds ([5 x i8]* @name, i32 0, i32 0)
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %typetag12 = load i8** %typetag5 tmp13 = icmp eq i8* %typetag12, getelementptr
  %inbounds ([5 x i8]* @name15, i32 0, i32 0)
```

```
  br i1 %tmp13, label %then15, label %else21

then:                                             ; preds = %entry
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3 src_y9 = load i32*
  %%src_y2 src_x10 = load i32* %src_x1 Kingrule_result = call i32 @Kingrule(i32
  %%src_x10, i32 %src_y9, i32 %dst_x8, i32 %dst_y7)
  store i32 %Kingrule_result, i32* %x
  %x11 = load i32* %x
  ret i32 %x11

else:                                             ; preds = %entry br label
%merge

merge14:                                          ; preds = %else21
  %typetag22 = load i8** %typetag5 tmp23 = icmp eq i8* %typetag22, getelementptr
  %inbounds ([7 x i8]* @name16, i32 0, i32 0)
  br i1 %tmp23, label %then25, label %else31

then15:                                           ; preds = %merge
  %dst_y16 = load i32* %dst_y4 dst_x17 = load i32* %dst_x3 src_y18 = load i32*
  %%src_y2 src_x19 = load i32* %src_x1 Pawnrule_result = call i32 @Pawnrule(i32
  %%src_x19, i32 %src_y18, i32 %dst_x17, i32 %dst_y16)
  store i32 %Pawnrule_result, i32* %x
  %x20 = load i32* %x
  ret i32 %x20

else21:                                           ; preds = %merge br label
%merge14

merge24:                                          ; preds = %else31
  %typetag32 = load i8** %typetag5 tmp33 = icmp eq i8* %typetag32, getelementptr
  %inbounds ([7 x i8]* @name17, i32 0, i32 0)
  br i1 %tmp33, label %then35, label %else41

then25:                                           ; preds = %merge14
  %dst_y26 = load i32* %dst_y4 dst_x27 = load i32* %dst_x3 src_y28 = load i32*
  %%src_y2 src_x29 = load i32* %src_x1 Bishoprule_result = call i32
  %@Bishoprule(i32 %src_x29, i32 %src_y28, i32 %dst_x27, i32 %dst_y26)
  store i32 %Bishoprule_result, i32* %x
  %x30 = load i32* %x
  ret i32 %x30

else31:                                           ; preds = %merge14 br label
%merge24

merge34:                                          ; preds = %else41
  %typetag42 = load i8** %typetag5 tmp43 = icmp eq i8* %typetag42, getelementptr
  %inbounds ([5 x i8]* @name18, i32 0, i32 0)
  br i1 %tmp43, label %then45, label %else51

then35:                                           ; preds = %merge24
  %dst_y36 = load i32* %dst_y4 dst_x37 = load i32* %dst_x3 src_y38 = load i32*
  %%src_y2 src_x39 = load i32* %src_x1 Knightrule_result = call i32
  %@Knightrule(i32 %src_x39, i32 %src_y38, i32 %dst_x37, i32 %dst_y36)
```

```
  store i32 %Knightrule_result, i32* %x
  %x40 = load i32* %x
  ret i32 %x40

else41:                                           ; preds = %merge24 br label
%merge34

merge44:                                          ; preds = %else51 ret i32 0

then45:                                           ; preds = %merge34
  %dst_y46 = load i32* %dst_y4 dst_x47 = load i32* %dst_x3 src_y48 = load i32*
  %%src_y2 src_x49 = load i32* %src_x1 Rookrule_result = call i32 @Rookrule(i32
  %%src_x49, i32 %src_y48, i32 %dst_x47, i32 %dst_y46)
  store i32 %Rookrule_result, i32* %x
  %x50 = load i32* %x
  ret i32 %x50

else51:                                           ; preds = %merge34 br label
%merge44 }

define void @nextPlayer() { entry:
  %repeat = alloca i32 currentPlayerIndex = load i32* @currentPlayerIndex tmp =
  %add i32 %currentPlayerIndex, 1 playerOrderSize = load i32* @playerOrderSize
  %tmp1 = urem i32 %tmp, %playerOrderSize
  store i32 %tmp1, i32* @currentPlayerIndex ret void }

define i32 @traverse(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %gx = alloca i32 sx = alloca i32 tx = alloca i32 gy = alloca i32 sy = alloca
  %i32 start_x = alloca i32 start_y = alloca i32 end_x = alloca i32 end_y =
  %alloca i32 iter = alloca %GenericPiece* repeat = alloca i32 dst_y5 = load
  %i32* %dst_y4 src_y6 = load i32* %src_y2 tmp = icmp eq i32 %dst_y5, %src_y6
  br i1 %tmp, label %then, label %else32

merge:                                            ; preds = %else32
  %src_x33 = load i32* %src_x1 dst_x34 = load i32* %dst_x3 tmp35 = icmp eq i32
  %%src_x33, %dst_x34
  br i1 %tmp35, label %then37, label %else67

then:                                             ; preds = %entry
  %src_x7 = load i32* %src_x1 dst_x8 = load i32* %dst_x3 tmp9 = icmp sgt i32
  %%src_x7, %dst_x8
  br i1 %tmp9, label %then11, label %else

merge10:                                          ; preds = %else, %then11
  %sx16 = load i32* %sx tmp17 = add i32 %sx16, 1
  store i32 %tmp17, i32* %tx br label %while
```

```
then11:                                           ; preds = %then
  %src_x12 = load i32* %src_x1
  store i32 %src_x12, i32* %gx
  %dst_x13 = load i32* %dst_x3
  store i32 %dst_x13, i32* %sx br label %merge10

else:                                             ; preds = %then
  %dst_x14 = load i32* %dst_x3
  store i32 %dst_x14, i32* %gx
  %src_x15 = load i32* %src_x1
  store i32 %src_x15, i32* %sx br label %merge10

while:                                            ; preds = %merge23, %merge10
  %tx28 = load i32* %tx gx29 = load i32* %gx tmp30 = icmp slt i32 %tx28, %gx29
  br i1 %tmp30, label %while_body, label %merge31

while_body:                                       ; preds = %while
  %tx18 = load i32* %tx src_y19 = load i32* %src_y2 name = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %tx18, i32
  %%src_y19 name20 = load %GenericPiece** %name
  store %GenericPiece* %name20, %GenericPiece** %iter
  %iter21 = load %GenericPiece** %iter tmp22 = icmp ne %GenericPiece* %iter21,
  %null
  br i1 %tmp22, label %then24, label %else25

merge23:                                          ; preds = %else25
  %tx26 = load i32* %tx tmp27 = add i32 %tx26, 1
  store i32 %tmp27, i32* %tx br label %while

then24:                                           ; preds = %while_body ret i32
1

else25:                                           ; preds = %while_body br label
%merge23

merge31:                                          ; preds = %while ret i32 0

else32:                                           ; preds = %entry br label
%merge

merge36:                                          ; preds = %else67
  %src_x68 = load i32* %src_x1
  store i32 %src_x68, i32* %start_x
  %src_y69 = load i32* %src_y2
  store i32 %src_y69, i32* %start_y
  %dst_x70 = load i32* %dst_x3
  store i32 %dst_x70, i32* %end_x
  %dst_y71 = load i32* %dst_y4
  store i32 %dst_y71, i32* %end_y
  %dst_y72 = load i32* %dst_y4 src_y73 = load i32* %src_y2 tmp74 = sub i32
  %%dst_y72, %src_y73 dst_x75 = load i32* %dst_x3 src_x76 = load i32* %src_x1
  %tmp77 = sub i32 %dst_x75, %src_x76 tmp78 = sdiv i32 %tmp74, %tmp77 absl =
  %call i32 @abs(i32 %tmp78) tmp79 = icmp eq i32 %absl, 1
```

```
  br i1 %tmp79, label %then81, label %else206

then37:                                           ; preds = %merge
  %src_y38 = load i32* %src_y2 dst_y39 = load i32* %dst_y4 tmp40 = icmp sgt i32
  %%src_y38, %dst_y39
  br i1 %tmp40, label %then42, label %else45

merge41:                                          ; preds = %else45, %then42
  %sy48 = load i32* %sy tmp49 = add i32 %sy48, 1
  store i32 %tmp49, i32* %tx br label %while50

then42:                                           ; preds = %then37
  %src_y43 = load i32* %src_y2
  store i32 %src_y43, i32* %gy
  %dst_y44 = load i32* %dst_y4
  store i32 %dst_y44, i32* %sy br label %merge41

else45:                                           ; preds = %then37
  %dst_y46 = load i32* %dst_y4
  store i32 %dst_y46, i32* %gy
  %src_y47 = load i32* %src_y2
  store i32 %src_y47, i32* %sy br label %merge41

while50:                                          ; preds = %merge58, %merge41
  %tx63 = load i32* %tx gy64 = load i32* %gy tmp65 = icmp slt i32 %tx63, %gy64
  br i1 %tmp65, label %while_body51, label %merge66

while_body51:                                     ; preds = %while50
  %src_x52 = load i32* %src_x1 tx53 = load i32* %tx name54 = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %src_x52, i32
  %%tx53 name55 = load %GenericPiece** %name54
  store %GenericPiece* %name55, %GenericPiece** %iter
  %iter56 = load %GenericPiece** %iter tmp57 = icmp ne %GenericPiece* %iter56,
  %null
  br i1 %tmp57, label %then59, label %else60

merge58:                                          ; preds = %else60
  %tx61 = load i32* %tx tmp62 = add i32 %tx61, 1
  store i32 %tmp62, i32* %tx br label %while50

then59:                                           ; preds = %while_body51 ret
i32 1

else60:                                           ; preds = %while_body51 br
label %merge58

merge66:                                          ; preds = %while50 ret i32 0

else67:                                           ; preds = %merge br label
%merge36

merge80:                                          ; preds = %else206, %merge182
ret i32 0
```

```
then81:                                           ; preds = %merge36
  %dst_x82 = load i32* %dst_x3 src_x83 = load i32* %src_x1 tmp84 = icmp sgt i32
  %%dst_x82, %src_x83 dst_y85 = load i32* %dst_y4 src_y86 = load i32* %src_y2
  %tmp87 = icmp sgt i32 %dst_y85, %src_y86 tmp88 = and i1 %tmp84, %tmp87
   br i1 %tmp88, label %then90, label %else112

merge89:                                          ; preds = %else112
  %dst_x113 = load i32* %dst_x3 src_x114 = load i32* %src_x1 tmp115 = icmp sgt
  %i32 %dst_x113, %src_x114 dst_y116 = load i32* %dst_y4 src_y117 = load i32*
  %%src_y2 tmp118 = icmp slt i32 %dst_y116, %src_y117 tmp119 = and i1 %tmp115,
  %%tmp118
   br i1 %tmp119, label %then121, label %else143

then90:                                           ; preds = %then81
  %start_x91 = load i32* %start_x tmp92 = add i32 %start_x91, 1
   store i32 %tmp92, i32* %tx br label %while93

while93:                                          ; preds = %merge103, %then90
  %tx108 = load i32* %tx end_x109 = load i32* %end_x tmp110 = icmp slt i32
  %%tx108, %end_x109
   br i1 %tmp110, label %while_body94, label %merge111

while_body94:                                     ; preds = %while93
  %start_y95 = load i32* %start_y tmp96 = add i32 %start_y95, 1
   store i32 %tmp96, i32* %start_y
  %tx97 = load i32* %tx start_y98 = load i32* %start_y name99 = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %tx97, i32
  %%start_y98 name100 = load %GenericPiece** %name99
   store %GenericPiece* %name100, %GenericPiece** %iter
  %iter101 = load %GenericPiece** %iter tmp102 = icmp ne %GenericPiece*
  %%iter101, null
   br i1 %tmp102, label %then104, label %else105

merge103:                                         ; preds = %else105
  %tx106 = load i32* %tx tmp107 = add i32 %tx106, 1
   store i32 %tmp107, i32* %tx br label %while93

then104:                                          ; preds = %while_body94 ret
i32 1

else105:                                          ; preds = %while_body94 br
label %merge103

merge111:                                         ; preds = %while93 ret i32 0

else112:                                          ; preds = %then81 br label
%merge89

merge120:                                         ; preds = %else143
  %dst_x144 = load i32* %dst_x3 src_x145 = load i32* %src_x1 tmp146 = icmp slt
  %i32 %dst_x144, %src_x145 dst_y147 = load i32* %dst_y4 src_y148 = load i32*
  %%src_y2 tmp149 = icmp slt i32 %dst_y147, %src_y148 tmp150 = and i1 %tmp146,
  %%tmp149
   br i1 %tmp150, label %then152, label %else174
```

```
then121:                                          ; preds = %merge89
  %start_x122 = load i32* %start_x tmp123 = add i32 %start_x122, 1
   store i32 %tmp123, i32* %tx br label %while124

while124:                                         ; preds = %merge134, %then121
  %tx139 = load i32* %tx end_x140 = load i32* %end_x tmp141 = icmp slt i32
  %%tx139, %end_x140
   br i1 %tmp141, label %while_body125, label %merge142

while_body125:                                    ; preds = %while124
  %start_y126 = load i32* %start_y tmp127 = sub i32 %start_y126, 1
   store i32 %tmp127, i32* %start_y
  %tx128 = load i32* %tx start_y129 = load i32* %start_y name130 = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %tx128, i32
  %%start_y129 name131 = load %GenericPiece** %name130
   store %GenericPiece* %name131, %GenericPiece** %iter
  %iter132 = load %GenericPiece** %iter tmp133 = icmp ne %GenericPiece*
  %%iter132, null
   br i1 %tmp133, label %then135, label %else136

merge134:                                         ; preds = %else136
  %tx137 = load i32* %tx tmp138 = add i32 %tx137, 1
   store i32 %tmp138, i32* %tx br label %while124

then135:                                          ; preds = %while_body125 ret
i32 1

else136:                                          ; preds = %while_body125 br
label %merge134

merge142:                                         ; preds = %while124 ret i32 0

else143:                                          ; preds = %merge89 br label
%merge120

merge151:                                         ; preds = %else174
  %dst_x175 = load i32* %dst_x3 src_x176 = load i32* %src_x1 tmp177 = icmp slt
  %i32 %dst_x175, %src_x176 dst_y178 = load i32* %dst_y4 src_y179 = load i32*
  %%src_y2 tmp180 = icmp sgt i32 %dst_y178, %src_y179 tmp181 = and i1 %tmp177,
  %%tmp180
   br i1 %tmp181, label %then183, label %else205

then152:                                          ; preds = %merge120
  %start_x153 = load i32* %start_x tmp154 = sub i32 %start_x153, 1
   store i32 %tmp154, i32* %tx br label %while155

while155:                                         ; preds = %merge165, %then152
  %tx170 = load i32* %tx end_x171 = load i32* %end_x tmp172 = icmp sgt i32
  %%tx170, %end_x171
   br i1 %tmp172, label %while_body156, label %merge173

while_body156:                                    ; preds = %while155
  %start_y157 = load i32* %start_y tmp158 = sub i32 %start_y157, 1
```

```
  store i32 %tmp158, i32* %start_y
  %tx159 = load i32* %tx start_y160 = load i32* %start_y name161 = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %tx159, i32
  %%start_y160 name162 = load %GenericPiece** %name161
  store %GenericPiece* %name162, %GenericPiece** %iter
  %iter163 = load %GenericPiece** %iter tmp164 = icmp ne %GenericPiece*
  %%iter163, null
  br i1 %tmp164, label %then166, label %else167

merge165:                                         ; preds = %else167
  %tx168 = load i32* %tx tmp169 = sub i32 %tx168, 1
  store i32 %tmp169, i32* %tx br label %while155

then166:                                          ; preds = %while_body156 ret
i32 1

else167:                                          ; preds = %while_body156 br
label %merge165

merge173:                                         ; preds = %while155 ret i32 0

else174:                                          ; preds = %merge120 br label
%merge151

merge182:                                         ; preds = %else205 br label
%merge80

then183:                                          ; preds = %merge151
  %start_x184 = load i32* %start_x tmp185 = sub i32 %start_x184, 1
  store i32 %tmp185, i32* %tx br label %while186

while186:                                         ; preds = %merge196, %then183
  %tx201 = load i32* %tx end_x202 = load i32* %end_x tmp203 = icmp sgt i32
  %%tx201, %end_x202
  br i1 %tmp203, label %while_body187, label %merge204

while_body187:                                    ; preds = %while186
  %start_y188 = load i32* %start_y tmp189 = add i32 %start_y188, 1
  store i32 %tmp189, i32* %start_y
  %tx190 = load i32* %tx start_y191 = load i32* %start_y name192 = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %tx190, i32
  %%start_y191 name193 = load %GenericPiece** %name192
  store %GenericPiece* %name193, %GenericPiece** %iter
  %iter194 = load %GenericPiece** %iter tmp195 = icmp ne %GenericPiece*
  %%iter194, null
  br i1 %tmp195, label %then197, label %else198

merge196:                                         ; preds = %else198
  %tx199 = load i32* %tx tmp200 = sub i32 %tx199, 1
  store i32 %tmp200, i32* %tx br label %while186

then197:                                          ; preds = %while_body187 ret
i32 1
```

```
else198:                                          ; preds = %while_body187 br
label %merge196

merge204:                                         ; preds = %while186 ret i32 0

else205:                                          ; preds = %merge151 br label
%merge182

else206:                                          ; preds = %merge36 br label
%merge80 }

define i32 @checkBound(i32 %x, i32 %y) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %repeat = alloca i32 x3 = load i32* %x1 tmp = icmp sgt i32 %x3, -1 x4 = load
  %i32* %x1 rows = load i32* @rows tmp5 = icmp sle i32 %x4, %rows tmp6 = and i1
  %%tmp, %tmp5 y7 = load i32* %y2 tmp8 = icmp sgt i32 %y7, -1 tmp9 = and i1
  %%tmp6, %tmp8 y10 = load i32* %y2 cols = load i32* @cols tmp11 = icmp sle i32
  %%y10, %cols tmp12 = and i1 %tmp9, %tmp11
  br i1 %tmp12, label %then, label %else

merge:                                            ; No predecessors!  ret i32 0

then:                                             ; preds = %entry ret i32 1

else:                                             ; preds = %entry ret i32 0 }

define %GenericPiece* @getPieceFromGrid(i8* %displayString) { entry:
  %displayString1 = alloca i8*
  store i8* %displayString, i8** %displayString1
  %iterator = alloca %GenericPiece* x = alloca i32 y = alloca i32 repeat =
  %alloca i32
  store i32 0, i32* %x br label %while

while:                                            ; preds = %merge23, %entry
  %x26 = load i32* %x rows = load i32* @rows tmp27 = icmp slt i32 %x26, %rows
  br i1 %tmp27, label %while_body, label %merge28

while_body:                                       ; preds = %while store i32 0,
i32* %y br label %while2

while2:                                           ; preds = %merge18,
%while_body
  %y21 = load i32* %y cols = load i32* @cols tmp22 = icmp slt i32 %y21, %cols
  br i1 %tmp22, label %while_body3, label %merge23

while_body3:                                      ; preds = %while2
  %x4 = load i32* %x y5 = load i32* %y name = getelementptr inbounds [5 x [4 x
  %%GenericPiece*]]* @GridData, i32 0, i32 %x4, i32 %y5 name6 = load
  %%GenericPiece** %name
  store %GenericPiece* %name6, %GenericPiece** %iterator br label %while7
```

```
while7:                                           ; preds = %merge, %while_body3
  %iterator16 = load %GenericPiece** %iterator tmp17 = icmp ne %GenericPiece*
  %%iterator16, null
  br i1 %tmp17, label %while_body8, label %merge18

while_body8:                                      ; preds = %while7
  %iterator9 = load %GenericPiece** %iterator loaded_deref = load %GenericPiece*
  %%iterator9 dotop_terminal = getelementptr inbounds %GenericPiece* %iterator9,
  %i32 0, i32 2 loaded_dotop_terminal = load i8** %dotop_terminal
  %displayString10 = load i8** %displayString1 tmp = icmp eq i8*
  %%loaded_dotop_terminal, %displayString10
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %iterator12 = load %GenericPiece** %iterator loaded_deref13 = load
  %%GenericPiece* %iterator12 dotop_terminal14 = getelementptr inbounds
  %%GenericPiece* %iterator12, i32 0, i32 3 loaded_dotop_terminal15 = load
  %%GenericPiece** %dotop_terminal14
  store %GenericPiece* %loaded_dotop_terminal15, %GenericPiece** %iterator br
label %while7

then:                                             ; preds = %while_body8
  %iterator11 = load %GenericPiece** %iterator
  ret %GenericPiece* %iterator11

else:                                             ; preds = %while_body8 br
label %merge

merge18:                                          ; preds = %while7
  %y19 = load i32* %y tmp20 = add i32 %y19, 1
  store i32 %tmp20, i32* %y br label %while2

merge23:                                          ; preds = %while2
  %x24 = load i32* %x tmp25 = add i32 %x24, 1
  store i32 %tmp25, i32* %x br label %while

merge28:                                          ; preds = %while store
%GenericPiece* null, %GenericPiece** %iterator
  %iterator29 = load %GenericPiece** %iterator
  ret %GenericPiece* %iterator29 }

define %GenericPiece* @getPieceAtLocation(i32 %x, i32 %y) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %head = alloca %GenericPiece* repeat = alloca i32 x3 = load i32* %x1 y4 = load
  %i32* %y2 name = getelementptr inbounds [5 x [4 x %GenericPiece*]]* @GridData,
  %i32 0, i32 %x3, i32 %y4 name5 = load %GenericPiece** %name
  store %GenericPiece* %name5, %GenericPiece** %head
  %head6 = load %GenericPiece** %head
  ret %GenericPiece* %head6 }

define i32 @moveOnGrid(%GenericPiece* %p_n, i32 %dst_x, i32 %dst_y) { entry:
```

```
  %p_n1 = alloca %GenericPiece*
  store %GenericPiece* %p_n, %GenericPiece** %p_n1
  %dst_x2 = alloca i32
  store i32 %dst_x, i32* %dst_x2
  %dst_y3 = alloca i32
  store i32 %dst_y, i32* %dst_y3
  %result = alloca i32 src_x = alloca i32 src_y = alloca i32 repeat = alloca i32
  %p_n4 = load %GenericPiece** %p_n1 tmp = icmp ne %GenericPiece* %p_n4, null
  br i1 %tmp, label %then, label %else32

merge:                                            ; preds = %merge20
  %result35 = load i32* %result
  ret i32 %result35

then:                                             ; preds = %entry
  %p_n5 = load %GenericPiece** %p_n1 loaded_deref = load %GenericPiece* %p_n5
  %dotop_terminal = getelementptr inbounds %GenericPiece* %p_n5, i32 0, i32 5
  %loaded_dotop_terminal = load i32* %dotop_terminal
  store i32 %loaded_dotop_terminal, i32* %src_x
  %p_n6 = load %GenericPiece** %p_n1 loaded_deref7 = load %GenericPiece* %p_n6
  %dotop_terminal8 = getelementptr inbounds %GenericPiece* %p_n6, i32 0, i32 4
  %loaded_dotop_terminal9 = load i32* %dotop_terminal8
  store i32 %loaded_dotop_terminal9, i32* %src_y
  %p_n10 = load %GenericPiece** %p_n1 loaded_deref11 = load %GenericPiece*
  %%p_n10 dotop_terminal12 = getelementptr inbounds %GenericPiece* %p_n10, i32
  %0, i32 1 loaded_dotop_terminal13 = load i8** %dotop_terminal12 dst_y14 = load
  %i32* %dst_y3 dst_x15 = load i32* %dst_x2 src_y16 = load i32* %src_y src_x17 =
  %load i32* %src_x triggerRule_result = call i32 @triggerRule(i32 %src_x17, i32
  %%src_y16, i32 %dst_x15, i32 %dst_y14, i8* %loaded_dotop_terminal13)
  store i32 %triggerRule_result, i32* %result
  %result18 = load i32* %result tmp19 = icmp eq i32 %result18, 1
  br i1 %tmp19, label %then21, label %else

merge20:                                          ; preds = %else, %then21 br
label %merge

then21:                                           ; preds = %then
  %p_n22 = load %GenericPiece** %p_n1 loaded_deref23 = load %GenericPiece*
  %%p_n22 dotop_terminal24 = getelementptr inbounds %GenericPiece* %p_n22, i32
  %0, i32 2 loaded_dotop_terminal25 = load i8** %dotop_terminal24 src_y26 = load
  %i32* %src_y src_x27 = load i32* %src_x
  call void @deleteFromGrid(i32 %src_x27, i32 %src_y26, i8*
%loaded_dotop_terminal25)
  %p_n28 = load %GenericPiece** %p_n1 dst_y29 = load i32* %dst_y3 dst_x30 = load
  %i32* %dst_x2
  call void @addToGrid(i32 %dst_x30, i32 %dst_y29, %GenericPiece* %p_n28) br
label %merge20

else:                                             ; preds = %then
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp31 = sub i32
  %%currentPlayerIndex, 1
  store i32 %tmp31, i32* @currentPlayerIndex br label %merge20

else32:                                           ; preds = %entry
```

```
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt30, i32 0, i32 0), i8* getelementptr inbounds ([17 x i8]* @name32, i32 0,
  %i32 0)) currentPlayerIndex33 = load i32* @currentPlayerIndex tmp34 = sub i32
  %%currentPlayerIndex33, 1
   store i32 %tmp34, i32* @currentPlayerIndex ret i32 0 }

define i32 @printGrid() { entry:
  %x = alloca i32 y = alloca i32 i = alloca i32 k = alloca i32 width = alloca
  %i32 tempLen = alloca i32 flag = alloca i32 max_width = alloca i32 border_len
  %= alloca i32 printer = alloca i8* iterator = alloca %GenericPiece* repeat =
  %alloca i32
   store i32 0, i32* %width store i32 0, i32* %max_width store i32 0, i32*
%border_len store i32 0, i32* %x br label %while

while:                                            ; preds = %merge43, %entry
  %x46 = load i32* %x rows = load i32* @rows tmp47 = icmp slt i32 %x46, %rows
   br i1 %tmp47, label %while_body, label %merge48

while_body:                                       ; preds = %while store i32 0,
i32* %y br label %while1

while1:                                           ; preds = %merge35,
%while_body
  %y41 = load i32* %y cols = load i32* @cols tmp42 = icmp slt i32 %y41, %cols
   br i1 %tmp42, label %while_body2, label %merge43

while_body2:                                      ; preds = %while1
  %x3 = load i32* %x y4 = load i32* %y name = getelementptr inbounds [5 x [4 x
  %%GenericPiece*]]* @GridData, i32 0, i32 %x3, i32 %y4 name5 = load
  %%GenericPiece** %name
   store %GenericPiece* %name5, %GenericPiece** %iterator store i32 0, i32*
%width
  %iterator6 = load %GenericPiece** %iterator tmp = icmp ne %GenericPiece*
  %%iterator6, null
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then br
label %while14

then:                                             ; preds = %while_body2
  %width7 = load i32* %width iterator8 = load %GenericPiece** %iterator
  %loaded_deref = load %GenericPiece* %iterator8 dotop_terminal = getelementptr
  %inbounds %GenericPiece* %iterator8, i32 0, i32 2 loaded_dotop_terminal = load
  %i8** %dotop_terminal getLen = call i32 @getLen(i8* %loaded_dotop_terminal)
  %tmp9 = add i32 %width7, %getLen
   store i32 %tmp9, i32* %width
  %iterator10 = load %GenericPiece** %iterator loaded_deref11 = load
  %%GenericPiece* %iterator10 dotop_terminal12 = getelementptr inbounds
  %%GenericPiece* %iterator10, i32 0, i32 3 loaded_dotop_terminal13 = load
  %%GenericPiece** %dotop_terminal12
   store %GenericPiece* %loaded_dotop_terminal13, %GenericPiece** %iterator br
label %merge

else:                                             ; preds = %while_body2 br
```

```
label %merge

while14:                                          ; preds = %while_body15,
%merge
  %iterator29 = load %GenericPiece** %iterator tmp30 = icmp ne %GenericPiece*
  %%iterator29, null
   br i1 %tmp30, label %while_body15, label %merge31

while_body15:                                     ; preds = %while14
  %width16 = load i32* %width tmp17 = add i32 %width16, 2
   store i32 %tmp17, i32* %width
  %width18 = load i32* %width iterator19 = load %GenericPiece** %iterator
  %loaded_deref20 = load %GenericPiece* %iterator19 dotop_terminal21 =
  %getelementptr inbounds %GenericPiece* %iterator19, i32 0, i32 2
  %loaded_dotop_terminal22 = load i8** %dotop_terminal21 getLen23 = call i32
  %@getLen(i8* %loaded_dotop_terminal22) tmp24 = add i32 %width18, %getLen23
   store i32 %tmp24, i32* %width
  %iterator25 = load %GenericPiece** %iterator loaded_deref26 = load
  %%GenericPiece* %iterator25 dotop_terminal27 = getelementptr inbounds
  %%GenericPiece* %iterator25, i32 0, i32 3 loaded_dotop_terminal28 = load
  %%GenericPiece** %dotop_terminal27
   store %GenericPiece* %loaded_dotop_terminal28, %GenericPiece** %iterator br
label %while14

merge31:                                          ; preds = %while14
  %width32 = load i32* %width max_width33 = load i32* %max_width tmp34 = icmp
  %sgt i32 %width32, %max_width33
   br i1 %tmp34, label %then36, label %else38

merge35:                                          ; preds = %else38, %then36
  %y39 = load i32* %y tmp40 = add i32 %y39, 1
   store i32 %tmp40, i32* %y br label %while1

then36:                                           ; preds = %merge31
  %width37 = load i32* %width
   store i32 %width37, i32* %max_width br label %merge35

else38:                                           ; preds = %merge31 br label
%merge35

merge43:                                          ; preds = %while1
  %x44 = load i32* %x tmp45 = add i32 %x44, 1
   store i32 %tmp45, i32* %x br label %while

merge48:                                          ; preds = %while
  %max_width49 = load i32* %max_width cols50 = load i32* @cols tmp51 = mul i32
  %%max_width49, %cols50 cols52 = load i32* @cols tmp53 = add i32 %tmp51,
  %%cols52
   store i32 %tmp53, i32* %border_len store i32 0, i32* %i br label %while54

while54:                                          ; preds = %while_body55,
%merge48
  %i58 = load i32* %i border_len59 = load i32* %border_len tmp60 = icmp slt i32
  %%i58, %border_len59
```

```
  br i1 %tmp60, label %while_body55, label %merge61

while_body55:                                     ; preds = %while54
  %print_sameline = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name35, i32 0, i32 0)) i56 = load i32* %i tmp57 = add i32 %i56, 1
  store i32 %tmp57, i32* %i br label %while54

merge61:                                          ; preds = %while54
  %print_endline = call i32 @print_endline()
  store i32 0, i32* %i br label %while62

while62:                                          ; preds = %merge86, %merge61
  %i89 = load i32* %i cols90 = load i32* @cols tmp91 = icmp slt i32 %i89,
  %%cols90
  br i1 %tmp91, label %while_body63, label %merge92

while_body63:                                     ; preds = %while62 store i32
0, i32* %k br label %while64

while64:                                          ; preds = %while_body65,
%while_body63
  %k69 = load i32* %k max_width70 = load i32* %max_width tmp71 = sdiv i32
  %%max_width70, 2 tmp72 = icmp slt i32 %k69, %tmp71
  br i1 %tmp72, label %while_body65, label %merge73

while_body65:                                     ; preds = %while64
  %print_sameline66 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name36, i32 0, i32 0)) k67 = load i32* %k tmp68 = add i32 %k67, 1
  store i32 %tmp68, i32* %k br label %while64

merge73:                                          ; preds = %while64
  %i74 = load i32* %i print_int_sameline = call i32 @print_int_sameline(i32
  %%i74)
  store i32 0, i32* %k br label %while75

while75:                                          ; preds = %while_body76,
%merge73
  %k80 = load i32* %k max_width81 = load i32* %max_width max_width82 = load i32*
  %%max_width tmp83 = sdiv i32 %max_width82, 2 tmp84 = sub i32 %max_width81,
  %%tmp83 tmp85 = icmp slt i32 %k80, %tmp84
  br i1 %tmp85, label %while_body76, label %merge86

while_body76:                                     ; preds = %while75
  %print_sameline77 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name37, i32 0, i32 0)) k78 = load i32* %k tmp79 = add i32 %k78, 1
  store i32 %tmp79, i32* %k br label %while75

merge86:                                          ; preds = %while75
  %i87 = load i32* %i tmp88 = add i32 %i87, 1
  store i32 %tmp88, i32* %i br label %while62

merge92:                                          ; preds = %while62
  %print_endline93 = call i32 @print_endline()
  store i32 0, i32* %x br label %while94
```

```
while94:                                          ; preds = %merge160, %merge92
  %x167 = load i32* %x rows168 = load i32* @rows tmp169 = icmp slt i32 %x167,
  %%rows168
   br i1 %tmp169, label %while_body95, label %merge170

while_body95:                                     ; preds = %while94 store i32
0, i32* %y br label %while96

while96:                                          ; preds = %merge154,
%while_body95
  %y157 = load i32* %y cols158 = load i32* @cols tmp159 = icmp slt i32 %y157,
  %%cols158
   br i1 %tmp159, label %while_body97, label %merge160

while_body97:                                     ; preds = %while96 store i32
0, i32* %tempLen
  %print_sameline98 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name38, i32 0, i32 0)) x99 = load i32* %x y100 = load i32* %y name101 =
  %getelementptr inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32
  %%x99, i32 %y100 name102 = load %GenericPiece** %name101
   store %GenericPiece* %name102, %GenericPiece** %iterator
  %iterator103 = load %GenericPiece** %iterator tmp104 = icmp ne %GenericPiece*
  %%iterator103, null
   br i1 %tmp104, label %then106, label %else121

merge105:                                         ; preds = %else121, %then106
br label %while122

then106:                                          ; preds = %while_body97
  %iterator107 = load %GenericPiece** %iterator loaded_deref108 = load
  %%GenericPiece* %iterator107 dotop_terminal109 = getelementptr inbounds
  %%GenericPiece* %iterator107, i32 0, i32 2 loaded_dotop_terminal110 = load
  %i8** %dotop_terminal109
   store i8* %loaded_dotop_terminal110, i8** %printer
  %tempLen111 = load i32* %tempLen printer112 = load i8** %printer getLen113 =
  %call i32 @getLen(i8* %printer112) tmp114 = add i32 %tempLen111, %getLen113
   store i32 %tmp114, i32* %tempLen
  %printer115 = load i8** %printer print_sameline116 = call i32
  %@print_sameline(i8* %printer115) iterator117 = load %GenericPiece** %iterator
  %loaded_deref118 = load %GenericPiece* %iterator117 dotop_terminal119 =
  %getelementptr inbounds %GenericPiece* %iterator117, i32 0, i32 3
  %loaded_dotop_terminal120 = load %GenericPiece** %dotop_terminal119
   store %GenericPiece* %loaded_dotop_terminal120, %GenericPiece** %iterator br
label %merge105

else121:                                          ; preds = %while_body97 br
label %merge105

while122:                                         ; preds = %while_body123,
%merge105
  %iterator141 = load %GenericPiece** %iterator tmp142 = icmp ne %GenericPiece*
  %%iterator141, null
   br i1 %tmp142, label %while_body123, label %merge143
```

```
while_body123:                                    ; preds = %while122
  %print_sameline124 = call i32 @print_sameline(i8* getelementptr inbounds ([3 x
  %i8]* @name39, i32 0, i32 0)) iterator125 = load %GenericPiece** %iterator
  %loaded_deref126 = load %GenericPiece* %iterator125 dotop_terminal127 =
  %getelementptr inbounds %GenericPiece* %iterator125, i32 0, i32 2
  %loaded_dotop_terminal128 = load i8** %dotop_terminal127
   store i8* %loaded_dotop_terminal128, i8** %printer
  %tempLen129 = load i32* %tempLen printer130 = load i8** %printer getLen131 =
  %call i32 @getLen(i8* %printer130) tmp132 = add i32 %tempLen129, %getLen131
   store i32 %tmp132, i32* %tempLen
  %tempLen133 = load i32* %tempLen tmp134 = add i32 %tempLen133, 2
   store i32 %tmp134, i32* %tempLen
  %printer135 = load i8** %printer print_sameline136 = call i32
  %@print_sameline(i8* %printer135) iterator137 = load %GenericPiece** %iterator
  %loaded_deref138 = load %GenericPiece* %iterator137 dotop_terminal139 =
  %getelementptr inbounds %GenericPiece* %iterator137, i32 0, i32 3
  %loaded_dotop_terminal140 = load %GenericPiece** %dotop_terminal139
   store %GenericPiece* %loaded_dotop_terminal140, %GenericPiece** %iterator br
label %while122

merge143:                                         ; preds = %while122 store i32
0, i32* %k br label %while144

while144:                                         ; preds = %while_body145,
%merge143
  %k149 = load i32* %k max_width150 = load i32* %max_width tempLen151 = load
  %i32* %tempLen tmp152 = sub i32 %max_width150, %tempLen151 tmp153 = icmp slt
  %i32 %k149, %tmp152
   br i1 %tmp153, label %while_body145, label %merge154

while_body145:                                    ; preds = %while144
  %print_sameline146 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name40, i32 0, i32 0)) k147 = load i32* %k tmp148 = add i32 %k147, 1
   store i32 %tmp148, i32* %k br label %while144

merge154:                                         ; preds = %while144
  %y155 = load i32* %y tmp156 = add i32 %y155, 1
   store i32 %tmp156, i32* %y br label %while96

merge160:                                         ; preds = %while96
  %print_sameline161 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name41, i32 0, i32 0)) x162 = load i32* %x print_int_sameline163 = call
  %i32 @print_int_sameline(i32 %x162) print_endline164 = call i32
  %@print_endline() x165 = load i32* %x tmp166 = add i32 %x165, 1
   store i32 %tmp166, i32* %x br label %while94

merge170:                                         ; preds = %while94 store i32
0, i32* %i br label %while171

while171:                                         ; preds = %while_body172,
%merge170
  %i176 = load i32* %i border_len177 = load i32* %border_len tmp178 = icmp slt
  %i32 %i176, %border_len177
```

```
  br i1 %tmp178, label %while_body172, label %merge179

while_body172:                                    ; preds = %while171
  %print_sameline173 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name42, i32 0, i32 0)) i174 = load i32* %i tmp175 = add i32 %i174, 1
  store i32 %tmp175, i32* %i br label %while171

merge179:                                         ; preds = %while171
  %print_endline180 = call i32 @print_endline()
  ret i32 0 }

define void @deleteFromGrid(i32 %x, i32 %y, i8* %tag) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %tag3 = alloca i8*
  store i8* %tag, i8** %tag3
  %iterator = alloca %GenericPiece* next_iterator = alloca %GenericPiece* repeat
  %= alloca i32 x4 = load i32* %x1 y5 = load i32* %y2 name = getelementptr
  %inbounds [5 x [4 x %GenericPiece*]]* @GridData, i32 0, i32 %x4, i32 %y5 name6
  %= load %GenericPiece** %name
  store %GenericPiece* %name6, %GenericPiece** %iterator
  %iterator7 = load %GenericPiece** %iterator loaded_deref = load %GenericPiece*
  %%iterator7 dotop_terminal = getelementptr inbounds %GenericPiece* %iterator7,
  %i32 0, i32 3 loaded_dotop_terminal = load %GenericPiece** %dotop_terminal
  store %GenericPiece* %loaded_dotop_terminal, %GenericPiece** %next_iterator
  %iterator8 = load %GenericPiece** %iterator loaded_deref9 = load
  %%GenericPiece* %iterator8 dotop_terminal10 = getelementptr inbounds
  %%GenericPiece* %iterator8, i32 0, i32 2 loaded_dotop_terminal11 = load i8**
  %%dotop_terminal10 tag12 = load i8** %tag3 tmp = icmp eq i8*
  %%loaded_dotop_terminal11, %tag12
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else br label
%while

then:                                             ; preds = %entry
  %x13 = load i32* %x1 y14 = load i32* %y2 name15 = getelementptr inbounds [5 x
  %[4 x %GenericPiece*]]* @GridData, i32 0, i32 %x13, i32 %y14 iterator16 = load
  %%GenericPiece** %iterator loaded_deref17 = load %GenericPiece* %iterator16
  %dotop_terminal18 = getelementptr inbounds %GenericPiece* %iterator16, i32 0,
  %i32 3 loaded_dotop_terminal19 = load %GenericPiece** %dotop_terminal18
  store %GenericPiece* %loaded_dotop_terminal19, %GenericPiece** %name15
  %iterator20 = load %GenericPiece** %iterator loaded_deref21 = load
  %%GenericPiece* %iterator20 next = getelementptr inbounds %GenericPiece*
  %%iterator20, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next store %GenericPiece* null,
%GenericPiece** %iterator ret void

else:                                             ; preds = %entry br label
%merge

while:                                            ; preds = %merge28, %merge
```

```
  %iterator49 = load %GenericPiece** %iterator loaded_deref50 = load
  %%GenericPiece* %iterator49 dotop_terminal51 = getelementptr inbounds
  %%GenericPiece* %iterator49, i32 0, i32 3 loaded_dotop_terminal52 = load
  %%GenericPiece** %dotop_terminal51 tmp53 = icmp ne %GenericPiece*
  %%loaded_dotop_terminal52, null
   br i1 %tmp53, label %while_body, label %merge54

while_body:                                       ; preds = %while
  %next_iterator22 = load %GenericPiece** %next_iterator loaded_deref23 = load
  %%GenericPiece* %next_iterator22 dotop_terminal24 = getelementptr inbounds
  %%GenericPiece* %next_iterator22, i32 0, i32 2 loaded_dotop_terminal25 = load
  %i8** %dotop_terminal24 tag26 = load i8** %tag3 tmp27 = icmp eq i8*
  %%loaded_dotop_terminal25, %tag26
   br i1 %tmp27, label %then29, label %else40

merge28:                                          ; preds = %else40
  %next_iterator41 = load %GenericPiece** %next_iterator loaded_deref42 = load
  %%GenericPiece* %next_iterator41 dotop_terminal43 = getelementptr inbounds
  %%GenericPiece* %next_iterator41, i32 0, i32 3 loaded_dotop_terminal44 = load
  %%GenericPiece** %dotop_terminal43
   store %GenericPiece* %loaded_dotop_terminal44, %GenericPiece** %next_iterator
  %iterator45 = load %GenericPiece** %iterator loaded_deref46 = load
  %%GenericPiece* %iterator45 dotop_terminal47 = getelementptr inbounds
  %%GenericPiece* %iterator45, i32 0, i32 3 loaded_dotop_terminal48 = load
  %%GenericPiece** %dotop_terminal47
   store %GenericPiece* %loaded_dotop_terminal48, %GenericPiece** %iterator br
label %while

then29:                                           ; preds = %while_body
  %next_iterator30 = load %GenericPiece** %next_iterator loaded_deref31 = load
  %%GenericPiece* %next_iterator30 dotop_terminal32 = getelementptr inbounds
  %%GenericPiece* %next_iterator30, i32 0, i32 3 loaded_dotop_terminal33 = load
  %%GenericPiece** %dotop_terminal32 iterator34 = load %GenericPiece** %iterator
  %loaded_deref35 = load %GenericPiece* %iterator34 next36 = getelementptr
  %inbounds %GenericPiece* %iterator34, i32 0, i32 3
   store %GenericPiece* %loaded_dotop_terminal33, %GenericPiece** %next36
  %next_iterator37 = load %GenericPiece** %next_iterator loaded_deref38 = load
  %%GenericPiece* %next_iterator37 next39 = getelementptr inbounds
  %%GenericPiece* %next_iterator37, i32 0, i32 3
   store %GenericPiece* null, %GenericPiece** %next39 store %GenericPiece* null,
%GenericPiece** %next_iterator ret void

else40:                                           ; preds = %while_body br label
%merge28

merge54:                                          ; preds = %while
  %iterator55 = load %GenericPiece** %iterator loaded_deref56 = load
  %%GenericPiece* %iterator55 dotop_terminal57 = getelementptr inbounds
  %%GenericPiece* %iterator55, i32 0, i32 3 loaded_dotop_terminal58 = load
  %%GenericPiece** %dotop_terminal57 tmp59 = icmp eq %GenericPiece*
  %%loaded_dotop_terminal58, null
   br i1 %tmp59, label %then61, label %else62

merge60:                                          ; preds = %else62 ret void
```

```
then61:                                           ; preds = %merge54
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt44, i32 0, i32 0), i8* getelementptr inbounds ([30 x i8]* @name46, i32 0,
  %i32 0))
  ret void

else62:                                           ; preds = %merge54 br label
%merge60 }

define void @addToGrid(i32 %x, i32 %y, %GenericPiece* %p_n) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %p_n3 = alloca %GenericPiece*
  store %GenericPiece* %p_n, %GenericPiece** %p_n3
  %iterator = alloca %GenericPiece* repeat = alloca i32 x4 = load i32* %x1 p_n5
  %= load %GenericPiece** %p_n3 loaded_deref = load %GenericPiece* %p_n5 x6 =
  %getelementptr inbounds %GenericPiece* %p_n5, i32 0, i32 5
  store i32 %x4, i32* %x6
  %y7 = load i32* %y2 p_n8 = load %GenericPiece** %p_n3 loaded_deref9 = load
  %%GenericPiece* %p_n8 y10 = getelementptr inbounds %GenericPiece* %p_n8, i32
  %0, i32 4
  store i32 %y7, i32* %y10
  %x11 = load i32* %x1 y12 = load i32* %y2 name = getelementptr inbounds [5 x [4
  %x %GenericPiece*]]* @GridData, i32 0, i32 %x11, i32 %y12 name13 = load
  %%GenericPiece** %name tmp = icmp eq %GenericPiece* %name13, null
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %merge69 ret void

then:                                             ; preds = %entry
  %x14 = load i32* %x1 y15 = load i32* %y2 name16 = getelementptr inbounds [5 x
  %[4 x %GenericPiece*]]* @GridData, i32 0, i32 %x14, i32 %y15 p_n17 = load
  %%GenericPiece** %p_n3
  store %GenericPiece* %p_n17, %GenericPiece** %name16
  %x18 = load i32* %x1 y19 = load i32* %y2 name20 = getelementptr inbounds [5 x
  %[4 x %GenericPiece*]]* @GridData, i32 0, i32 %x18, i32 %y19 name21 = load
  %%GenericPiece** %name20
  store %GenericPiece* %name21, %GenericPiece** %iterator
  %iterator22 = load %GenericPiece** %iterator loaded_deref23 = load
  %%GenericPiece* %iterator22 next = getelementptr inbounds %GenericPiece*
  %%iterator22, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next ret void

else:                                             ; preds = %entry
  %x24 = load i32* %x1 y25 = load i32* %y2 name26 = getelementptr inbounds [5 x
  %[4 x %GenericPiece*]]* @GridData, i32 0, i32 %x24, i32 %y25 name27 = load
  %%GenericPiece** %name26
  store %GenericPiece* %name27, %GenericPiece** %iterator br label %while

while:                                            ; preds = %while_body, %else
  %iterator30 = load %GenericPiece** %iterator loaded_deref31 = load
```

```
  %%GenericPiece* %iterator30 dotop_terminal32 = getelementptr inbounds
  %%GenericPiece* %iterator30, i32 0, i32 3 loaded_dotop_terminal33 = load
  %%GenericPiece** %dotop_terminal32 tmp34 = icmp ne %GenericPiece*
  %%loaded_dotop_terminal33, null
  br i1 %tmp34, label %while_body, label %merge35

while_body:                                       ; preds = %while
  %iterator28 = load %GenericPiece** %iterator loaded_deref29 = load
  %%GenericPiece* %iterator28 dotop_terminal = getelementptr inbounds
  %%GenericPiece* %iterator28, i32 0, i32 3 loaded_dotop_terminal = load
  %%GenericPiece** %dotop_terminal
  store %GenericPiece* %loaded_dotop_terminal, %GenericPiece** %iterator br
label %while

merge35:                                          ; preds = %while
  %p_n36 = load %GenericPiece** %p_n3 iterator37 = load %GenericPiece**
  %%iterator loaded_deref38 = load %GenericPiece* %iterator37 next39 =
  %getelementptr inbounds %GenericPiece* %iterator37, i32 0, i32 3
  store %GenericPiece* %p_n36, %GenericPiece** %next39
  %iterator40 = load %GenericPiece** %iterator loaded_deref41 = load
  %%GenericPiece* %iterator40 dotop_terminal42 = getelementptr inbounds
  %%GenericPiece* %iterator40, i32 0, i32 3 loaded_dotop_terminal43 = load
  %%GenericPiece** %dotop_terminal42
  store %GenericPiece* %loaded_dotop_terminal43, %GenericPiece** %iterator
  %iterator44 = load %GenericPiece** %iterator loaded_deref45 = load
  %%GenericPiece* %iterator44 next46 = getelementptr inbounds %GenericPiece*
  %%iterator44, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next46
  %x47 = load i32* %x1 y48 = load i32* %y2 name49 = getelementptr inbounds [5 x
  %[4 x %GenericPiece*]]* @GridData, i32 0, i32 %x47, i32 %y48 name50 = load
  %%GenericPiece** %name49
  store %GenericPiece* %name50, %GenericPiece** %iterator br label %while51

while51:                                          ; preds = %merge56, %merge35
  %iterator67 = load %GenericPiece** %iterator tmp68 = icmp ne %GenericPiece*
  %%iterator67, null
  br i1 %tmp68, label %while_body52, label %merge69

while_body52:                                     ; preds = %while51
  %iterator53 = load %GenericPiece** %iterator p_n54 = load %GenericPiece**
  %%p_n3 tmp55 = icmp ne %GenericPiece* %iterator53, %p_n54
  br i1 %tmp55, label %then57, label %else62

merge56:                                          ; preds = %else62, %then57
  %iterator63 = load %GenericPiece** %iterator loaded_deref64 = load
  %%GenericPiece* %iterator63 dotop_terminal65 = getelementptr inbounds
  %%GenericPiece* %iterator63, i32 0, i32 3 loaded_dotop_terminal66 = load
  %%GenericPiece** %dotop_terminal65
  store %GenericPiece* %loaded_dotop_terminal66, %GenericPiece** %iterator br
label %while51

then57:                                           ; preds = %while_body52
  %iterator58 = load %GenericPiece** %iterator p_n59 = load %GenericPiece**
  %%p_n3 y60 = load i32* %y2 x61 = load i32* %x1 colocation_result = call i32
```

```
  %@colocation(i32 %x61, i32 %y60, %GenericPiece* %p_n59, %GenericPiece*
  %%iterator58)
   br label %merge56

else62:                                           ; preds = %while_body52 br
label %merge56

merge69:                                          ; preds = %while51 br label
%merge }

define i32 @gameloop() { entry:
  %src_x = alloca i32 src_y = alloca i32 dst_x = alloca i32 dst_y = alloca i32
  %headnode = alloca %GenericPiece* p_n = alloca %GenericPiece* cur = alloca
  %%Player repeat = alloca i32
   store i32 0, i32* %repeat store i32 0, i32* @currentPlayerIndex br label
%while

while:                                            ; preds = %merge, %entry
  %repeat18 = load i32* %repeat tmp19 = icmp eq i32 %repeat18, 0
   br i1 %tmp19, label %while_body, label %merge20

while_body:                                       ; preds = %while
  %currentPlayerIndex = load i32* @currentPlayerIndex name = getelementptr
  %inbounds [2 x %Player]* @playerOrder, i32 0, i32 %currentPlayerIndex name1 =
  %load %Player* %name
   store %Player %name1, %Player* %cur
  %printGrid_result = call i32 @printGrid() cur2 = load %Player* %cur
  %dotop_terminal = getelementptr inbounds %Player* %cur, i32 0, i32 0
  %loaded_dotop_terminal = load i8** %dotop_terminal cur3 = load %Player* %cur
  %dotop_terminal4 = getelementptr inbounds %Player* %cur, i32 0, i32 0
  %loaded_dotop_terminal5 = load i8** %dotop_terminal4 printf = call i32 (i8*,
  %...)* @printf(i8* getelementptr inbounds ([4 x i8]* @mmt50, i32 0, i32 0),
  %i8* %loaded_dotop_terminal5) prompt = call i32 @prompt(i8* getelementptr
  %inbounds ([16 x i8]* @name51, i32 0, i32 0))
   store i32 %prompt, i32* %src_x
  %prompt6 = call i32 @prompt(i8* getelementptr inbounds ([16 x i8]* @name52,
  %i32 0, i32 0))
   store i32 %prompt6, i32* %src_y
  %prompt7 = call i32 @prompt(i8* getelementptr inbounds ([21 x i8]* @name53,
  %i32 0, i32 0))
   store i32 %prompt7, i32* %dst_x
  %prompt8 = call i32 @prompt(i8* getelementptr inbounds ([21 x i8]* @name54,
  %i32 0, i32 0))
   store i32 %prompt8, i32* %dst_y
  %src_y9 = load i32* %src_y src_x10 = load i32* %src_x
  %getPieceAtLocation_result = call %GenericPiece* @getPieceAtLocation(i32
  %%src_x10, i32 %src_y9)
   store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %p_n
  %dst_y11 = load i32* %dst_y dst_x12 = load i32* %dst_x p_n13 = load
  %%GenericPiece** %p_n moveOnGrid_result = call i32 @moveOnGrid(%GenericPiece*
  %%p_n13, i32 %dst_x12, i32 %dst_y11) tmp = icmp ne i32 %moveOnGrid_result, 1
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
```

```
  %checkGameEnd_result = call i32 @checkGameEnd()
   store i32 %checkGameEnd_result, i32* %repeat
  %currentPlayerIndex15 = load i32* @currentPlayerIndex tmp16 = add i32
  %%currentPlayerIndex15, 1 playerOrderSize = load i32* @playerOrderSize tmp17 =
  %urem i32 %tmp16, %playerOrderSize
   store i32 %tmp17, i32* @currentPlayerIndex br label %while

then:                                             ; preds = %while_body
  %printf14 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt50, i32 0, i32 0), i8* getelementptr inbounds ([13 x i8]* @name56, i32 0,
  %i32 0))
   br label %merge

else:                                             ; preds = %while_body br label
%merge

merge20:                                          ; preds = %while ret i32 0 }

define i32 @main() { entry:
  %repeat = alloca i32 p1 = load %Player* @p1
   store i8* getelementptr inbounds ([6 x i8]* @name59, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 0)
  %p2 = load %Player* @p2
   store i8* getelementptr inbounds ([6 x i8]* @name60, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 0)
  %p11 = load %Player* @p1
   store %Player %p11, %Player* getelementptr inbounds ([2 x %Player]*
@playerOrder, i32 0, i32 0)
  %p22 = load %Player* @p2
   store %Player %p22, %Player* getelementptr inbounds ([2 x %Player]*
@playerOrder, i32 0, i32 1) store i32 0, i32* @flag
  %p13 = load %Player* @p1 loaded_dotop_terminal = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p14 = load %Player* @p1
  %loaded_dotop_terminal5 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1)
   store i8* getelementptr inbounds ([7 x i8]* @name61, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 1, i32 2)
  %p16 = load %Player* @p1 loaded_dotop_terminal7 = load %Rook* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 2) p18 = load %Player* @p1
  %loaded_dotop_terminal9 = load %Rook* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 2)
   store i8* getelementptr inbounds ([7 x i8]* @name62, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 2, i32 2)
  %p110 = load %Player* @p1 loaded_dotop_terminal11 = load %Knight*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 3) p112 = load %Player* @p1
  %loaded_dotop_terminal13 = load %Knight* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 3)
   store i8* getelementptr inbounds ([9 x i8]* @name63, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 3, i32 2)
  %p114 = load %Player* @p1 loaded_dotop_terminal15 = load %Bishop*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 4) p116 = load %Player* @p1
  %loaded_dotop_terminal17 = load %Bishop* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 4)
   store i8* getelementptr inbounds ([9 x i8]* @name64, i32 0, i32 0), i8**
```

```
getelementptr inbounds (%Player* @p1, i32 0, i32 4, i32 2)
  %p118 = load %Player* @p1 loaded_dotop_terminal19 = load %Pawn* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 5) p120 = load %Player* @p1
  %loaded_dotop_terminal21 = load %Pawn* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 5)
   store i8* getelementptr inbounds ([7 x i8]* @name65, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 5, i32 2)
  %p222 = load %Player* @p2 loaded_dotop_terminal23 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p224 = load %Player* @p2
  %loaded_dotop_terminal25 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1)
   store i8* getelementptr inbounds ([7 x i8]* @name66, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 1, i32 2)
  %p226 = load %Player* @p2 loaded_dotop_terminal27 = load %Rook* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 2) p228 = load %Player* @p2
  %loaded_dotop_terminal29 = load %Rook* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 2)
   store i8* getelementptr inbounds ([7 x i8]* @name67, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 2, i32 2)
  %p230 = load %Player* @p2 loaded_dotop_terminal31 = load %Knight*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 3) p232 = load %Player* @p2
  %loaded_dotop_terminal33 = load %Knight* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 3)
   store i8* getelementptr inbounds ([9 x i8]* @name68, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 3, i32 2)
  %p234 = load %Player* @p2 loaded_dotop_terminal35 = load %Bishop*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 4) p236 = load %Player* @p2
  %loaded_dotop_terminal37 = load %Bishop* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 4)
   store i8* getelementptr inbounds ([9 x i8]* @name69, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 4, i32 2)
  %p238 = load %Player* @p2 loaded_dotop_terminal39 = load %Pawn* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 5) p240 = load %Player* @p2
  %loaded_dotop_terminal41 = load %Pawn* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 5)
   store i8* getelementptr inbounds ([7 x i8]* @name70, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 5, i32 2)
  %p142 = load %Player* @p1 loaded_dotop_terminal43 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) newNode = alloca %GenericPiece p144 =
  %load %Player* @p1 newNode45 = load %GenericPiece* %newNode owner =
  %getelementptr inbounds %GenericPiece* %newNode, i32 0, i32 0
   store %Player* @p1, %Player** %owner
  %p146 = load %Player* @p1 loaded_dotop_terminal47 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) newNode48 = load %GenericPiece*
  %%newNode King_node = getelementptr inbounds %GenericPiece* %newNode, i32 0,
  %i32 10
   store %King* getelementptr inbounds (%Player* @p1, i32 0, i32 1), %King**
%King_node
  %p149 = load %Player* @p1 loaded_dotop_terminal50 = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p151 = load %Player* @p1
  %loaded_dotop_terminal52 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) newNode53 = load %GenericPiece* %newNode nametag =
  %getelementptr inbounds %GenericPiece* %newNode, i32 0, i32 2
```

```
  store i8* %loaded_dotop, i8** %nametag
  %newNode54 = load %GenericPiece* %newNode typetag = getelementptr inbounds
  %%GenericPiece* %newNode, i32 0, i32 1
  store i8* getelementptr inbounds ([5 x i8]* @name71, i32 0, i32 0), i8**
%typetag
  %newNode55 = load %GenericPiece* %newNode
  call void @addToGrid(i32 4, i32 3, %GenericPiece* %newNode)
  %p156 = load %Player* @p1 loaded_dotop_terminal57 = load %Pawn* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 5) newNode58 = alloca %GenericPiece p159 =
  %load %Player* @p1 newNode60 = load %GenericPiece* %newNode58 owner61 =
  %getelementptr inbounds %GenericPiece* %newNode58, i32 0, i32 0
  store %Player* @p1, %Player** %owner61
  %p162 = load %Player* @p1 loaded_dotop_terminal63 = load %Pawn* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 5) newNode64 = load %GenericPiece*
  %%newNode58 Pawn_node = getelementptr inbounds %GenericPiece* %newNode58, i32
  %0, i32 9
  store %Pawn* getelementptr inbounds (%Player* @p1, i32 0, i32 5), %Pawn**
%Pawn_node
  %p165 = load %Player* @p1 loaded_dotop_terminal66 = load %Pawn* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 5) p167 = load %Player* @p1
  %loaded_dotop_terminal68 = load %Pawn* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 5) loaded_dotop69 = load i8** getelementptr inbounds (%Player*
  %@p1, i32 0, i32 5, i32 2) newNode70 = load %GenericPiece* %newNode58
  %nametag71 = getelementptr inbounds %GenericPiece* %newNode58, i32 0, i32 2
  store i8* %loaded_dotop69, i8** %nametag71
  %newNode72 = load %GenericPiece* %newNode58 typetag73 = getelementptr inbounds
  %%GenericPiece* %newNode58, i32 0, i32 1
  store i8* getelementptr inbounds ([5 x i8]* @name72, i32 0, i32 0), i8**
%typetag73
  %newNode74 = load %GenericPiece* %newNode58
  call void @addToGrid(i32 3, i32 3, %GenericPiece* %newNode58)
  %p175 = load %Player* @p1 loaded_dotop_terminal76 = load %Bishop*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 4) newNode77 = alloca
  %%GenericPiece p178 = load %Player* @p1 newNode79 = load %GenericPiece*
  %%newNode77 owner80 = getelementptr inbounds %GenericPiece* %newNode77, i32 0,
  %i32 0
  store %Player* @p1, %Player** %owner80
  %p181 = load %Player* @p1 loaded_dotop_terminal82 = load %Bishop*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 4) newNode83 = load
  %%GenericPiece* %newNode77 Bishop_node = getelementptr inbounds %GenericPiece*
  %%newNode77, i32 0, i32 8
  store %Bishop* getelementptr inbounds (%Player* @p1, i32 0, i32 4), %Bishop**
%Bishop_node
  %p184 = load %Player* @p1 loaded_dotop_terminal85 = load %Bishop*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 4) p186 = load %Player* @p1
  %loaded_dotop_terminal87 = load %Bishop* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 4) loaded_dotop88 = load i8** getelementptr inbounds (%Player*
  %@p1, i32 0, i32 4, i32 2) newNode89 = load %GenericPiece* %newNode77
  %nametag90 = getelementptr inbounds %GenericPiece* %newNode77, i32 0, i32 2
  store i8* %loaded_dotop88, i8** %nametag90
  %newNode91 = load %GenericPiece* %newNode77 typetag92 = getelementptr inbounds
  %%GenericPiece* %newNode77, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name73, i32 0, i32 0), i8**
%typetag92
```

```
  %newNode93 = load %GenericPiece* %newNode77
   call void @addToGrid(i32 4, i32 1, %GenericPiece* %newNode77)
  %p194 = load %Player* @p1 loaded_dotop_terminal95 = load %Knight*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 3) newNode96 = alloca
  %%GenericPiece p197 = load %Player* @p1 newNode98 = load %GenericPiece*
  %%newNode96 owner99 = getelementptr inbounds %GenericPiece* %newNode96, i32 0,
  %i32 0
   store %Player* @p1, %Player** %owner99
  %p1100 = load %Player* @p1 loaded_dotop_terminal101 = load %Knight*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 3) newNode102 = load
  %%GenericPiece* %newNode96 Knight_node = getelementptr inbounds %GenericPiece*
  %%newNode96, i32 0, i32 7
   store %Knight* getelementptr inbounds (%Player* @p1, i32 0, i32 3), %Knight**
%Knight_node
  %p1103 = load %Player* @p1 loaded_dotop_terminal104 = load %Knight*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 3) p1105 = load %Player* @p1
  %loaded_dotop_terminal106 = load %Knight* getelementptr inbounds (%Player*
  %@p1, i32 0, i32 3) loaded_dotop107 = load i8** getelementptr inbounds
  %(%Player* @p1, i32 0, i32 3, i32 2) newNode108 = load %GenericPiece*
  %%newNode96 nametag109 = getelementptr inbounds %GenericPiece* %newNode96, i32
  %0, i32 2
   store i8* %loaded_dotop107, i8** %nametag109
  %newNode110 = load %GenericPiece* %newNode96 typetag111 = getelementptr
  %inbounds %GenericPiece* %newNode96, i32 0, i32 1
   store i8* getelementptr inbounds ([7 x i8]* @name74, i32 0, i32 0), i8**
%typetag111
  %newNode112 = load %GenericPiece* %newNode96
   call void @addToGrid(i32 4, i32 2, %GenericPiece* %newNode96)
  %p1113 = load %Player* @p1 loaded_dotop_terminal114 = load %Rook*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 2) newNode115 = alloca
  %%GenericPiece p1116 = load %Player* @p1 newNode117 = load %GenericPiece*
  %%newNode115 owner118 = getelementptr inbounds %GenericPiece* %newNode115, i32
  %0, i32 0
   store %Player* @p1, %Player** %owner118
  %p1119 = load %Player* @p1 loaded_dotop_terminal120 = load %Rook*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 2) newNode121 = load
  %%GenericPiece* %newNode115 Rook_node = getelementptr inbounds %GenericPiece*
  %%newNode115, i32 0, i32 6
   store %Rook* getelementptr inbounds (%Player* @p1, i32 0, i32 2), %Rook**
%Rook_node
  %p1122 = load %Player* @p1 loaded_dotop_terminal123 = load %Rook*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 2) p1124 = load %Player* @p1
  %loaded_dotop_terminal125 = load %Rook* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 2) loaded_dotop126 = load i8** getelementptr inbounds (%Player*
  %@p1, i32 0, i32 2, i32 2) newNode127 = load %GenericPiece* %newNode115
  %nametag128 = getelementptr inbounds %GenericPiece* %newNode115, i32 0, i32 2
   store i8* %loaded_dotop126, i8** %nametag128
  %newNode129 = load %GenericPiece* %newNode115 typetag130 = getelementptr
  %inbounds %GenericPiece* %newNode115, i32 0, i32 1
   store i8* getelementptr inbounds ([5 x i8]* @name75, i32 0, i32 0), i8**
%typetag130
  %newNode131 = load %GenericPiece* %newNode115
   call void @addToGrid(i32 4, i32 0, %GenericPiece* %newNode115)
  %p2132 = load %Player* @p2 loaded_dotop_terminal133 = load %King*
```

```
  %getelementptr inbounds (%Player* @p2, i32 0, i32 1) newNode134 = alloca
  %%GenericPiece p2135 = load %Player* @p2 newNode136 = load %GenericPiece*
  %%newNode134 owner137 = getelementptr inbounds %GenericPiece* %newNode134, i32
  %0, i32 0
  store %Player* @p2, %Player** %owner137
  %p2138 = load %Player* @p2 loaded_dotop_terminal139 = load %King*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 1) newNode140 = load
  %%GenericPiece* %newNode134 King_node141 = getelementptr inbounds
  %%GenericPiece* %newNode134, i32 0, i32 10
  store %King* getelementptr inbounds (%Player* @p2, i32 0, i32 1), %King**
%King_node141
  %p2142 = load %Player* @p2 loaded_dotop_terminal143 = load %King*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 1) p2144 = load %Player* @p2
  %loaded_dotop_terminal145 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop146 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 1, i32 2) newNode147 = load %GenericPiece* %newNode134
  %nametag148 = getelementptr inbounds %GenericPiece* %newNode134, i32 0, i32 2
  store i8* %loaded_dotop146, i8** %nametag148
  %newNode149 = load %GenericPiece* %newNode134 typetag150 = getelementptr
  %inbounds %GenericPiece* %newNode134, i32 0, i32 1
  store i8* getelementptr inbounds ([5 x i8]* @name76, i32 0, i32 0), i8**
%typetag150
  %newNode151 = load %GenericPiece* %newNode134
  call void @addToGrid(i32 0, i32 0, %GenericPiece* %newNode134)
  %p2152 = load %Player* @p2 loaded_dotop_terminal153 = load %Pawn*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 5) newNode154 = alloca
  %%GenericPiece p2155 = load %Player* @p2 newNode156 = load %GenericPiece*
  %%newNode154 owner157 = getelementptr inbounds %GenericPiece* %newNode154, i32
  %0, i32 0
  store %Player* @p2, %Player** %owner157
  %p2158 = load %Player* @p2 loaded_dotop_terminal159 = load %Pawn*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 5) newNode160 = load
  %%GenericPiece* %newNode154 Pawn_node161 = getelementptr inbounds
  %%GenericPiece* %newNode154, i32 0, i32 9
  store %Pawn* getelementptr inbounds (%Player* @p2, i32 0, i32 5), %Pawn**
%Pawn_node161
  %p2162 = load %Player* @p2 loaded_dotop_terminal163 = load %Pawn*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 5) p2164 = load %Player* @p2
  %loaded_dotop_terminal165 = load %Pawn* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 5) loaded_dotop166 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 5, i32 2) newNode167 = load %GenericPiece* %newNode154
  %nametag168 = getelementptr inbounds %GenericPiece* %newNode154, i32 0, i32 2
  store i8* %loaded_dotop166, i8** %nametag168
  %newNode169 = load %GenericPiece* %newNode154 typetag170 = getelementptr
  %inbounds %GenericPiece* %newNode154, i32 0, i32 1
  store i8* getelementptr inbounds ([5 x i8]* @name77, i32 0, i32 0), i8**
%typetag170
  %newNode171 = load %GenericPiece* %newNode154
  call void @addToGrid(i32 1, i32 0, %GenericPiece* %newNode154)
  %p2172 = load %Player* @p2 loaded_dotop_terminal173 = load %Bishop*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 4) newNode174 = alloca
  %%GenericPiece p2175 = load %Player* @p2 newNode176 = load %GenericPiece*
  %%newNode174 owner177 = getelementptr inbounds %GenericPiece* %newNode174, i32
  %0, i32 0
```

```
  store %Player* @p2, %Player** %owner177
  %p2178 = load %Player* @p2 loaded_dotop_terminal179 = load %Bishop*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 4) newNode180 = load
  %%GenericPiece* %newNode174 Bishop_node181 = getelementptr inbounds
  %%GenericPiece* %newNode174, i32 0, i32 8
  store %Bishop* getelementptr inbounds (%Player* @p2, i32 0, i32 4), %Bishop**
%Bishop_node181
  %p2182 = load %Player* @p2 loaded_dotop_terminal183 = load %Bishop*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 4) p2184 = load %Player* @p2
  %loaded_dotop_terminal185 = load %Bishop* getelementptr inbounds (%Player*
  %@p2, i32 0, i32 4) loaded_dotop186 = load i8** getelementptr inbounds
  %(%Player* @p2, i32 0, i32 4, i32 2) newNode187 = load %GenericPiece*
  %%newNode174 nametag188 = getelementptr inbounds %GenericPiece* %newNode174,
  %i32 0, i32 2
  store i8* %loaded_dotop186, i8** %nametag188
  %newNode189 = load %GenericPiece* %newNode174 typetag190 = getelementptr
  %inbounds %GenericPiece* %newNode174, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name78, i32 0, i32 0), i8**
%typetag190
  %newNode191 = load %GenericPiece* %newNode174
  call void @addToGrid(i32 0, i32 2, %GenericPiece* %newNode174)
  %p2192 = load %Player* @p2 loaded_dotop_terminal193 = load %Knight*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 3) newNode194 = alloca
  %%GenericPiece p2195 = load %Player* @p2 newNode196 = load %GenericPiece*
  %%newNode194 owner197 = getelementptr inbounds %GenericPiece* %newNode194, i32
  %0, i32 0
  store %Player* @p2, %Player** %owner197
  %p2198 = load %Player* @p2 loaded_dotop_terminal199 = load %Knight*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 3) newNode200 = load
  %%GenericPiece* %newNode194 Knight_node201 = getelementptr inbounds
  %%GenericPiece* %newNode194, i32 0, i32 7
  store %Knight* getelementptr inbounds (%Player* @p2, i32 0, i32 3), %Knight**
%Knight_node201
  %p2202 = load %Player* @p2 loaded_dotop_terminal203 = load %Knight*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 3) p2204 = load %Player* @p2
  %loaded_dotop_terminal205 = load %Knight* getelementptr inbounds (%Player*
  %@p2, i32 0, i32 3) loaded_dotop206 = load i8** getelementptr inbounds
  %(%Player* @p2, i32 0, i32 3, i32 2) newNode207 = load %GenericPiece*
  %%newNode194 nametag208 = getelementptr inbounds %GenericPiece* %newNode194,
  %i32 0, i32 2
  store i8* %loaded_dotop206, i8** %nametag208
  %newNode209 = load %GenericPiece* %newNode194 typetag210 = getelementptr
  %inbounds %GenericPiece* %newNode194, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name79, i32 0, i32 0), i8**
%typetag210
  %newNode211 = load %GenericPiece* %newNode194
  call void @addToGrid(i32 0, i32 1, %GenericPiece* %newNode194)
  %p2212 = load %Player* @p2 loaded_dotop_terminal213 = load %Rook*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 2) newNode214 = alloca
  %%GenericPiece p2215 = load %Player* @p2 newNode216 = load %GenericPiece*
  %%newNode214 owner217 = getelementptr inbounds %GenericPiece* %newNode214, i32
  %0, i32 0
  store %Player* @p2, %Player** %owner217
  %p2218 = load %Player* @p2 loaded_dotop_terminal219 = load %Rook*
```

```
  %getelementptr inbounds (%Player* @p2, i32 0, i32 2) newNode220 = load
  %%GenericPiece* %newNode214 Rook_node221 = getelementptr inbounds
  %%GenericPiece* %newNode214, i32 0, i32 6
   store %Rook* getelementptr inbounds (%Player* @p2, i32 0, i32 2), %Rook**
%Rook_node221
  %p2222 = load %Player* @p2 loaded_dotop_terminal223 = load %Rook*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 2) p2224 = load %Player* @p2
  %loaded_dotop_terminal225 = load %Rook* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 2) loaded_dotop226 = load i8** getelementptr inbounds (%Player*
  %@p2, i32 0, i32 2, i32 2) newNode227 = load %GenericPiece* %newNode214
  %nametag228 = getelementptr inbounds %GenericPiece* %newNode214, i32 0, i32 2
   store i8* %loaded_dotop226, i8** %nametag228
  %newNode229 = load %GenericPiece* %newNode214 typetag230 = getelementptr
  %inbounds %GenericPiece* %newNode214, i32 0, i32 1
   store i8* getelementptr inbounds ([5 x i8]* @name80, i32 0, i32 0), i8**
%typetag230
  %newNode231 = load %GenericPiece* %newNode214
   call void @addToGrid(i32 0, i32 3, %GenericPiece* %newNode214) store i32 2,
i32* @playerOrderSize
  %gameloop_result = call i32 @gameloop()
   ret i32 0 }

define i32 @colocation(i32 %x, i32 %y, %GenericPiece* %i1, %GenericPiece* %i2) {
entry:
  %x1 = alloca i32
   store i32 %x, i32* %x1
  %y2 = alloca i32
   store i32 %y, i32* %y2
  %i13 = alloca %GenericPiece*
   store %GenericPiece* %i1, %GenericPiece** %i13
  %i24 = alloca %GenericPiece*
   store %GenericPiece* %i2, %GenericPiece** %i24
  %repeat = alloca i32 i25 = load %GenericPiece** %i24 loaded_deref = load
  %%GenericPiece* %i25 dotop_terminal = getelementptr inbounds %GenericPiece*
  %%i25, i32 0, i32 2 loaded_dotop_terminal = load i8** %dotop_terminal y6 =
  %load i32* %y2 x7 = load i32* %x1
   call void @deleteFromGrid(i32 %x7, i32 %y6, i8* %loaded_dotop_terminal) ret
i32 0 }

define i32 @getDirectionFromIndex() { entry:
  %direction = alloca i32 repeat = alloca i32 currentPlayerIndex = load i32*
  %@currentPlayerIndex tmp = icmp eq i32 %currentPlayerIndex, 0
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %direction1 = load i32* %direction
   ret i32 %direction1

then:                                             ; preds = %entry store i32 -1,
i32* %direction br label %merge

else:                                             ; preds = %entry store i32 1,
i32* %direction br label %merge }
```

```
define i1 @checkIfUnderAttack(i32 %dst_x, i32 %dst_y) { entry:
  %dst_x1 = alloca i32
  store i32 %dst_x, i32* %dst_x1
  %dst_y2 = alloca i32
  store i32 %dst_y, i32* %dst_y2
  %p = alloca %Player* x = alloca i32 y = alloca i32 l = alloca %GenericPiece*
  %repeat = alloca i32
  call void @nextPlayer()
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp = icmp eq i32
  %%currentPlayerIndex, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then store
i32 0, i32* %x br label %while

then:                                             ; preds = %entry
  %p1 = load %Player* @p1
  store %Player* @p1, %Player** %p br label %merge

else:                                             ; preds = %entry
  %p2 = load %Player* @p2
  store %Player* @p2, %Player** %p br label %merge

while:                                            ; preds = %merge36, %merge
  %x39 = load i32* %x rows = load i32* @rows tmp40 = icmp slt i32 %x39, %rows
  br i1 %tmp40, label %while_body, label %merge41

while_body:                                       ; preds = %while store i32 0,
i32* %y br label %while3

while3:                                           ; preds = %merge9, %while_body
  %y34 = load i32* %y cols = load i32* @cols tmp35 = icmp slt i32 %y34, %cols
  br i1 %tmp35, label %while_body4, label %merge36

while_body4:                                      ; preds = %while3
  %y5 = load i32* %y x6 = load i32* %x getPieceAtLocation_result = call
  %%GenericPiece* @getPieceAtLocation(i32 %x6, i32 %y5)
  store %GenericPiece* %getPieceAtLocation_result, %GenericPiece** %l
  %l7 = load %GenericPiece** %l tmp8 = icmp ne %GenericPiece* %l7, null
  br i1 %tmp8, label %then10, label %else31

merge9:                                           ; preds = %else31, %merge14
  %y32 = load i32* %y tmp33 = add i32 %y32, 1
  store i32 %tmp33, i32* %y br label %while3

then10:                                           ; preds = %while_body4
  %l11 = load %GenericPiece** %l loaded_deref = load %GenericPiece* %l11
  %dotop_terminal = getelementptr inbounds %GenericPiece* %l11, i32 0, i32 0
  %loaded_dotop_terminal = load %Player** %dotop_terminal p12 = load %Player**
  %%p tmp13 = icmp eq %Player* %loaded_dotop_terminal, %p12
  br i1 %tmp13, label %then15, label %else30

merge14:                                          ; preds = %else30, %merge25 br
label %merge9
```

```
then15:                                           ; preds = %then10
  %l16 = load %GenericPiece** %l loaded_deref17 = load %GenericPiece* %l16
  %dotop_terminal18 = getelementptr inbounds %GenericPiece* %l16, i32 0, i32 1
  %loaded_dotop_terminal19 = load i8** %dotop_terminal18 dst_y20 = load i32*
  %%dst_y2 dst_x21 = load i32* %dst_x1 y22 = load i32* %y x23 = load i32* %x
  %triggerRule_result = call i32 @triggerRule(i32 %x23, i32 %y22, i32 %dst_x21,
  %i32 %dst_y20, i8* %loaded_dotop_terminal19) tmp24 = icmp eq i32
  %%triggerRule_result, 1
   br i1 %tmp24, label %then26, label %else29

merge25:                                          ; preds = %else29 br label
%merge14

then26:                                           ; preds = %then15
  %x27 = load i32* %x
   store i32 %x27, i32* @attx
  %y28 = load i32* %y
   store i32 %y28, i32* @atty call void @nextPlayer() ret i1 true

else29:                                           ; preds = %then15 br label
%merge25

else30:                                           ; preds = %then10 br label
%merge14

else31:                                           ; preds = %while_body4 br
label %merge9

merge36:                                          ; preds = %while3
  %x37 = load i32* %x tmp38 = add i32 %x37, 1
   store i32 %tmp38, i32* %x br label %while

merge41:                                          ; preds = %while call void
@nextPlayer() ret i1 false }

define i32 @checkGameEnd() { entry:
  %p = alloca %Player* kp = alloca %GenericPiece* dst_x = alloca i32 dst_y =
  %alloca i32 tempx = alloca i32 tempy = alloca i32 repeat = alloca i32
   call void @nextPlayer()
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp = icmp eq i32
  %%currentPlayerIndex, 0
   br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %kp8 = load %GenericPiece** %kp loaded_deref = load %GenericPiece* %kp8
  %dotop_terminal = getelementptr inbounds %GenericPiece* %kp8, i32 0, i32 4
  %loaded_dotop_terminal9 = load i32* %dotop_terminal kp10 = load
  %%GenericPiece** %kp loaded_deref11 = load %GenericPiece* %kp10
  %dotop_terminal12 = getelementptr inbounds %GenericPiece* %kp10, i32 0, i32 5
  %loaded_dotop_terminal13 = load i32* %dotop_terminal12
  %checkIfUnderAttack_result = call i1 @checkIfUnderAttack(i32
  %%loaded_dotop_terminal13, i32 %loaded_dotop_terminal9)
   br i1 %checkIfUnderAttack_result, label %then15, label %else54
```

```
then:                                             ; preds = %entry
  %p1 = load %Player* @p1 loaded_dotop_terminal = load %King* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 1) p11 = load %Player* @p1
  %loaded_dotop_terminal2 = load %King* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1) loaded_dotop = load i8** getelementptr inbounds (%Player* @p1,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop)
   store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %kp br label
%merge

else:                                             ; preds = %entry
  %p2 = load %Player* @p2 loaded_dotop_terminal3 = load %King* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 1) p24 = load %Player* @p2
  %loaded_dotop_terminal5 = load %King* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1) loaded_dotop6 = load i8** getelementptr inbounds (%Player* @p2,
  %i32 0, i32 1, i32 2) getPieceFromGrid_result7 = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop6)
   store %GenericPiece* %getPieceFromGrid_result7, %GenericPiece** %kp br label
%merge

merge14:                                          ; preds = %else54 call void
@nextPlayer() ret i32 0

then15:                                           ; preds = %merge
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt88, i32 0, i32 0), i8* getelementptr inbounds ([22 x i8]* @name90, i32 0,
  %i32 0))
   store i32 1, i32* @flag store i32 0, i32* %dst_x br label %while

while:                                            ; preds = %merge47, %then15
  %dst_x50 = load i32* %dst_x rows = load i32* @rows tmp51 = icmp slt i32
  %%dst_x50, %rows
   br i1 %tmp51, label %while_body, label %merge52

while_body:                                       ; preds = %while store i32 0,
i32* %dst_y br label %while16

while16:                                          ; preds = %merge33,
%while_body
  %dst_y45 = load i32* %dst_y cols = load i32* @cols tmp46 = icmp slt i32
  %%dst_y45, %cols
   br i1 %tmp46, label %while_body17, label %merge47

while_body17:                                     ; preds = %while16
  %kp18 = load %GenericPiece** %kp loaded_deref19 = load %GenericPiece* %kp18
  %dotop_terminal20 = getelementptr inbounds %GenericPiece* %kp18, i32 0, i32 1
  %loaded_dotop_terminal21 = load i8** %dotop_terminal20 dst_y22 = load i32*
  %%dst_y dst_x23 = load i32* %dst_x kp24 = load %GenericPiece** %kp
  %loaded_deref25 = load %GenericPiece* %kp24 dotop_terminal26 = getelementptr
  %inbounds %GenericPiece* %kp24, i32 0, i32 4 loaded_dotop_terminal27 = load
  %i32* %dotop_terminal26 kp28 = load %GenericPiece** %kp loaded_deref29 = load
  %%GenericPiece* %kp28 dotop_terminal30 = getelementptr inbounds %GenericPiece*
  %%kp28, i32 0, i32 5 loaded_dotop_terminal31 = load i32* %dotop_terminal30
```

```
  %triggerRule_result = call i32 @triggerRule(i32 %loaded_dotop_terminal31, i32
  %%loaded_dotop_terminal27, i32 %dst_x23, i32 %dst_y22, i8*
  %%loaded_dotop_terminal21) tmp32 = icmp eq i32 %triggerRule_result, 1
  br i1 %tmp32, label %then34, label %else42

merge33:                                          ; preds = %else42
  %dst_y43 = load i32* %dst_y tmp44 = add i32 %dst_y43, 1
  store i32 %tmp44, i32* %dst_y br label %while16

then34:                                           ; preds = %while_body17
  %printf35 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt88, i32 0, i32 0), i8* getelementptr inbounds ([14 x i8]* @name92, i32 0,
  %i32 0)) dst_x36 = load i32* %dst_x dst_x37 = load i32* %dst_x printf38 = call
  %i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt87, i32 0,
  %i32 0), i32 %dst_x37) dst_y39 = load i32* %dst_y dst_y40 = load i32* %dst_y
  %printf41 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@fmt87, i32 0, i32 0), i32 %dst_y40)
  call void @nextPlayer() store i32 0, i32* @flag ret i32 0

else42:                                           ; preds = %while_body17 br
label %merge33

merge47:                                          ; preds = %while16
  %dst_x48 = load i32* %dst_x tmp49 = add i32 %dst_x48, 1
  store i32 %tmp49, i32* %dst_x br label %while

merge52:                                          ; preds = %while store i32 0,
i32* @flag
  %printGrid_result = call i32 @printGrid() printf53 = call i32 (i8*, ...)*
  %@printf(i8* getelementptr inbounds ([4 x i8]* @mmt88, i32 0, i32 0), i8*
  %getelementptr inbounds ([11 x i8]* @name94, i32 0, i32 0))
  ret i32 1

else54:                                           ; preds = %merge br label
%merge14
```

Snakes and Ladders
snakes_and_ladders.grid

```
import gridBasics.grid

int checkGameEnd()
{
  Piece Token *t;
  Piece GenericPiece *token;
  if(currentPlayerIndex == 0)
  {
    t = &p1.token;
  }
  else
  {
    t = &p2.token;
  }
  /*Get grid item from item*/
  token = getPieceFromGrid(t.displayString);
```

```
  if (token.x == 0 && token.y == 5)
  {
    printGrid();
    print("Winner is: ");
    print(t.displayString);
    return 1;
  }
  return 0;
}

Player
{
  Piece Token token;
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    return 1;
  }
}

Piece Ladder
{
}

Piece LadderTop
{
}

Piece Snake
{
}

Piece Snaketail
{
}

Piece Token
{
  int rule(int src_x, int src_y, int dst_x, int dst_y)
  {
    return checkBound(dst_x, dst_y);
  }
}

int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
  Piece GenericPiece* l;
  int dst_x, dst_y;
  if(i1.typetag == "Token" && i2.typetag == "Snake")
  {
    printGrid();
    print("Got bit!");
    if (i2.nametag == "S1-Head")
    {
      l = getPieceFromGrid("S1-Tail");
```

```
    }
    if (i2.nametag == "S2-Head")
    {
      l = getPieceFromGrid("S2-Tail");
    }
    dst_x = l.x;
    dst_y = l.y;
    moveOnGrid(i1, dst_x, dst_y);
  }
  if(i1.typetag == "Token" && i2.typetag == "Ladder")
  {
    printGrid();
    print("Ladder boost!");
    if (i2.nametag == "L1-Bottom")
    {
      l = getPieceFromGrid("L1-Top");
    }
    if (i2.nametag == "L2-Bottom")
    {
      l = getPieceFromGrid("L2-Top");
    }

    dst_x = l.x;
    dst_y = l.y;
    moveOnGrid(i1, dst_x, dst_y);
  }
  return 0;
}

Grid_Init<6,6>;
Player p1, p2;
Player[2] playerOrder;

int setup(){
  Piece Ladder ladder1, ladder2;
  Piece LadderTop ladder1_top, ladder2_top;
  Piece Snake snake1, snake2;
  Piece Snaketail snake1_tail, snake2_tail;
  ladder1.displayString = "L1-Bottom";
  ladder2.displayString = "L2-Bottom";
  ladder1_top.displayString = "L1-Top";
  ladder2_top.displayString = "L2-Top";

  snake1.displayString = "S1-Head";
  snake1_tail.displayString = "S1-Tail";
  snake2.displayString = "S2-Head";
  snake2_tail.displayString = "S2-Tail";

  p1.token.displayString = "P1";
  p2.token.displayString = "P2";

  Grid <4,2> <-- snake1;
  Grid <5,4> <-- snake1_tail;
  Grid <0,4> <-- snake2;
```

```
  Grid <5,1> <-- snake2_tail;

  Grid <2,3> <-- ladder1;
  Grid <0,1> <-- ladder1_top;
  Grid <4,0> <-- ladder2;
  Grid <2,0> <-- ladder2_top;

  Grid <5,5> <-- p1.token;
  Grid <5,5> <-- p2.token;
  return 0;
}

int makeMove(int dice)
{
  int i, src_x, src_y, dst_x, dst_y, direction;
  Piece Token *t;
  Piece GenericPiece *token;
  if(currentPlayerIndex == 0)
  {
    t = &p1.token;
  }
  else
  {
    t = &p2.token;
  }
  /*Get grid item from item*/
  token = getPieceFromGrid(t.displayString);

  src_x = token.x;
  src_y = token.y;
  dst_x = src_x;
  dst_y = src_y;

  /*Find the destination cell using dice and source loc of player token*/
  /*First find out the current direction the token will move in*/
  if (token.x \% 2 == 0)
  {
    direction = 1;
  }
  else
  {
    direction = -1;
  }
  for(i=0; i<dice ; i=i+1)
  {
    if((dst_y == 0 && direction == -1) || (dst_y == 5 && direction == 1))
    {
      dst_x = dst_x - 1;
      direction = -direction;
    }
    else
    {
      dst_y = dst_y + direction;
    }
```

```
  }
  print("Dice roll");
  print(dice);
  moveOnGrid(token, dst_x, dst_y);
}

int gameloop(){
  int dice;
  printGrid();
  prompt("Go!");
  dice = diceThrow();
  makeMove(dice);
  return 0;
}
```

snakes_and_ladders.ll

```
; ModuleID = 'GridLang'

%Player = type <{ %Token }> Token = type <{ i1, i8*, i1 }> GenericPiece = type
%<{ %Player*, i8*, i8*, %GenericPiece*, i32, i32, %Token*, %Snaketail*,
%%Snake*, %LadderTop*, %Ladder* }> Snaketail = type <{ i1, i8*, i1 }> Snake =
%type <{ i1, i8*, i1 }> LadderTop = type <{ i1, i8*, i1 }> Ladder = type <{ i1,
%i8*, i1 }>

@playerOrder = global [2 x %Player] zeroinitializer @p2 = global %Player
zeroinitializer @p1 = global %Player zeroinitializer @rows = global i32 6 @cols
= global i32 6 @GridData = global [6 x [6 x %GenericPiece*]] zeroinitializer
@playerOrderSize = global i32 0 @currentPlayerIndex = global i32 0 @fmt =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt1 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt2 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt3 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt4 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt5 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt6 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt7 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt8 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt9 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt10 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt11
= private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt12 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt13 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt14 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @name = private unnamed_addr constant [7 x i8] c"Ladder\00" @name15
= private unnamed_addr constant [10 x i8] c"LadderTop\00" @name16 = private
unnamed_addr constant [6 x i8] c"Snake\00" @name17 = private unnamed_addr
constant [10 x i8] c"Snaketail\00" @name18 = private unnamed_addr constant [6 x
i8] c"Token\00" @fmt19 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@mmt20 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt21 = private
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt22 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @fmt23 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt24 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt25
= private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt26 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt27 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt28 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @fmt29 = private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt30
= private unnamed_addr constant [4 x i8] c"%s\0A\00" @name31 = private
```

```
unnamed_addr constant [17 x i8] c"No piece on cell\00" @name32 = private
unnamed_addr constant [17 x i8] c"No piece on cell\00" @fmt33 = private
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt34 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @name35 = private unnamed_addr constant [2 x i8]
c"_\00" @name36 = private unnamed_addr constant [2 x i8] c"_\00" @name37 =
private unnamed_addr constant [2 x i8] c"_\00" @name38 = private unnamed_addr
constant [2 x i8] c"|\00" @name39 = private unnamed_addr constant [3 x i8] c",
\00" @name40 = private unnamed_addr constant [2 x i8] c" \00" @name41 = private
unnamed_addr constant [2 x i8] c"|\00" @name42 = private unnamed_addr constant
[2 x i8] c"-\00" @fmt43 = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@mmt44 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @name45 = private
unnamed_addr constant [30 x i8] c"Not found on given coordinate\00" @name46 =
private unnamed_addr constant [30 x i8] c"Not found on given coordinate\00"
@fmt47 = private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt48 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @fmt49 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt50 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @name51 = private unnamed_addr constant [4 x i8] c"Go!\00" @fmt52 =
private unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt53 = private
unnamed_addr constant [4 x i8] c"%s\0A\00" @name54 = private unnamed_addr
constant [10 x i8] c"Dice roll\00" @name55 = private unnamed_addr constant [10
x i8] c"Dice roll\00" @fmt56 = private unnamed_addr constant [4 x i8]
c"%d\0A\00" @mmt57 = private unnamed_addr constant [4 x i8] c"%s\0A\00" @name58
= private unnamed_addr constant [10 x i8] c"L1-Bottom\00" @name59 = private
unnamed_addr constant [10 x i8] c"L2-Bottom\00" @name60 = private unnamed_addr
constant [7 x i8] c"L1-Top\00" @name61 = private unnamed_addr constant [7 x i8]
c"L2-Top\00" @name62 = private unnamed_addr constant [8 x i8] c"S1-Head\00"
@name63 = private unnamed_addr constant [8 x i8] c"S1-Tail\00" @name64 =
private unnamed_addr constant [8 x i8] c"S2-Head\00" @name65 = private
unnamed_addr constant [8 x i8] c"S2-Tail\00" @name66 = private unnamed_addr
constant [3 x i8] c"P1\00" @name67 = private unnamed_addr constant [3 x i8]
c"P2\00" @name68 = private unnamed_addr constant [6 x i8] c"Snake\00" @name69 =
private unnamed_addr constant [10 x i8] c"Snaketail\00" @name70 = private
unnamed_addr constant [6 x i8] c"Snake\00" @name71 = private unnamed_addr
constant [10 x i8] c"Snaketail\00" @name72 = private unnamed_addr constant [7 x
i8] c"Ladder\00" @name73 = private unnamed_addr constant [10 x i8]
c"LadderTop\00" @name74 = private unnamed_addr constant [7 x i8] c"Ladder\00"
@name75 = private unnamed_addr constant [10 x i8] c"LadderTop\00" @name76 =
private unnamed_addr constant [6 x i8] c"Token\00" @name77 = private
unnamed_addr constant [6 x i8] c"Token\00" @fmt78 = private unnamed_addr
constant [4 x i8] c"%d\0A\00" @mmt79 = private unnamed_addr constant [4 x i8]
c"%s\0A\00" @name80 = private unnamed_addr constant [6 x i8] c"Token\00"
@name81 = private unnamed_addr constant [6 x i8] c"Snake\00" @name82 = private
unnamed_addr constant [9 x i8] c"Got bit!\00" @name83 = private unnamed_addr
constant [9 x i8] c"Got bit!\00" @name84 = private unnamed_addr constant [8 x
i8] c"S1-Head\00" @name85 = private unnamed_addr constant [8 x i8]
c"S1-Tail\00" @name86 = private unnamed_addr constant [8 x i8] c"S2-Head\00"
@name87 = private unnamed_addr constant [8 x i8] c"S2-Tail\00" @name88 =
private unnamed_addr constant [6 x i8] c"Token\00" @name89 = private
unnamed_addr constant [7 x i8] c"Ladder\00" @name90 = private unnamed_addr
constant [14 x i8] c"Ladder boost!\00" @name91 = private unnamed_addr constant
[14 x i8] c"Ladder boost!\00" @name92 = private unnamed_addr constant [10 x i8]
c"L1-Bottom\00" @name93 = private unnamed_addr constant [7 x i8] c"L1-Top\00"
@name94 = private unnamed_addr constant [10 x i8] c"L2-Bottom\00" @name95 =
private unnamed_addr constant [7 x i8] c"L2-Top\00" @fmt96 = private
```

```
unnamed_addr constant [4 x i8] c"%d\0A\00" @mmt97 = private unnamed_addr
constant [4 x i8] c"%s\0A\00" @name98 = private unnamed_addr constant [12 x i8]
c"Winner is: \00" @name99 = private unnamed_addr constant [12 x i8] c"Winner
is: \00"

declare i32 @printf(i8*, ...)

declare i32 @prompt(i8*)

declare i32 @abs(i32)

declare i32 @print_endline()

declare i32 @print_sameline(i8*)

declare i32 @diceThrow()

declare i32 @getLen(i8*)

declare i32 @print_int_sameline(i32)

define i32 @Tokenrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32 dst_y5 = load i32* %dst_y4 dst_x6 = load i32* %dst_x3
  %checkBound_result = call i32 @checkBound(i32 %dst_x6, i32 %dst_y5)
  ret i32 %checkBound_result }

define i32 @Snaketailrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) {
entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @Snakerule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
```

```
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @LadderToprule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) {
entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @Ladderrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @Playerrule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }

define i32 @GenericPiecerule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) {
entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %repeat = alloca i32
  ret i32 1 }
```

```
define i32 @triggerRule(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y, i8*
%typetag) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
  store i32 %dst_y, i32* %dst_y4
  %typetag5 = alloca i8*
  store i8* %typetag, i8** %typetag5
  %x = alloca i32 repeat = alloca i32 typetag6 = load i8** %typetag5 tmp = icmp
  %eq i8* %typetag6, getelementptr inbounds ([7 x i8]* @name, i32 0, i32 0)
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %typetag12 = load i8** %typetag5 tmp13 = icmp eq i8* %typetag12,
  %getelementptr inbounds ([10 x i8]* @name15, i32 0, i32 0)
  br i1 %tmp13, label %then15, label %else21

then:                                             ; preds = %entry
  %dst_y7 = load i32* %dst_y4 dst_x8 = load i32* %dst_x3 src_y9 = load i32*
  %%src_y2 src_x10 = load i32* %src_x1 Ladderrule_result = call i32
  %@Ladderrule(i32 %src_x10, i32 %src_y9, i32 %dst_x8, i32 %dst_y7)
  store i32 %Ladderrule_result, i32* %x
  %x11 = load i32* %x
  ret i32 %x11

else:                                             ; preds = %entry br label
%merge

merge14:                                          ; preds = %else21
  %typetag22 = load i8** %typetag5 tmp23 = icmp eq i8* %typetag22,
  %getelementptr inbounds ([6 x i8]* @name16, i32 0, i32 0)
  br i1 %tmp23, label %then25, label %else31

then15:                                           ; preds = %merge
  %dst_y16 = load i32* %dst_y4 dst_x17 = load i32* %dst_x3 src_y18 = load i32*
  %%src_y2 src_x19 = load i32* %src_x1 LadderToprule_result = call i32
  %@LadderToprule(i32 %src_x19, i32 %src_y18, i32 %dst_x17, i32 %dst_y16)
  store i32 %LadderToprule_result, i32* %x
  %x20 = load i32* %x
  ret i32 %x20

else21:                                           ; preds = %merge br label
%merge14

merge24:                                          ; preds = %else31
  %typetag32 = load i8** %typetag5 tmp33 = icmp eq i8* %typetag32,
  %getelementptr inbounds ([10 x i8]* @name17, i32 0, i32 0)
  br i1 %tmp33, label %then35, label %else41
```

```
then25:                                           ; preds = %merge14
  %dst_y26 = load i32* %dst_y4 dst_x27 = load i32* %dst_x3 src_y28 = load i32*
  %%src_y2 src_x29 = load i32* %src_x1 Snakerule_result = call i32
  %@Snakerule(i32 %src_x29, i32 %src_y28, i32 %dst_x27, i32 %dst_y26)
  store i32 %Snakerule_result, i32* %x
  %x30 = load i32* %x
  ret i32 %x30

else31:                                           ; preds = %merge14 br label
%merge24

merge34:                                          ; preds = %else41
  %typetag42 = load i8** %typetag5 tmp43 = icmp eq i8* %typetag42,
  %getelementptr inbounds ([6 x i8]* @name18, i32 0, i32 0)
  br i1 %tmp43, label %then45, label %else51

then35:                                           ; preds = %merge24
  %dst_y36 = load i32* %dst_y4 dst_x37 = load i32* %dst_x3 src_y38 = load i32*
  %%src_y2 src_x39 = load i32* %src_x1 Snaketailrule_result = call i32
  %@Snaketailrule(i32 %src_x39, i32 %src_y38, i32 %dst_x37, i32 %dst_y36)
  store i32 %Snaketailrule_result, i32* %x
  %x40 = load i32* %x
  ret i32 %x40

else41:                                           ; preds = %merge24 br label
%merge34

merge44:                                          ; preds = %else51 ret i32 0

then45:                                           ; preds = %merge34
  %dst_y46 = load i32* %dst_y4 dst_x47 = load i32* %dst_x3 src_y48 = load i32*
  %%src_y2 src_x49 = load i32* %src_x1 Tokenrule_result = call i32
  %@Tokenrule(i32 %src_x49, i32 %src_y48, i32 %dst_x47, i32 %dst_y46)
  store i32 %Tokenrule_result, i32* %x
  %x50 = load i32* %x
  ret i32 %x50

else51:                                           ; preds = %merge34 br label
%merge44 }

define void @nextPlayer() { entry:
  %repeat = alloca i32 currentPlayerIndex = load i32* @currentPlayerIndex tmp =
  %add i32 %currentPlayerIndex, 1 playerOrderSize = load i32* @playerOrderSize
  %tmp1 = urem i32 %tmp, %playerOrderSize
  store i32 %tmp1, i32* @currentPlayerIndex ret void }

define i32 @traverse(i32 %src_x, i32 %src_y, i32 %dst_x, i32 %dst_y) { entry:
  %src_x1 = alloca i32
  store i32 %src_x, i32* %src_x1
  %src_y2 = alloca i32
  store i32 %src_y, i32* %src_y2
  %dst_x3 = alloca i32
  store i32 %dst_x, i32* %dst_x3
  %dst_y4 = alloca i32
```

```
  store i32 %dst_y, i32* %dst_y4
  %gx = alloca i32 sx = alloca i32 tx = alloca i32 gy = alloca i32 sy = alloca
  %i32 start_x = alloca i32 start_y = alloca i32 end_x = alloca i32 end_y =
  %alloca i32 iter = alloca %GenericPiece* repeat = alloca i32 dst_y5 = load
  %i32* %dst_y4 src_y6 = load i32* %src_y2 tmp = icmp eq i32 %dst_y5, %src_y6
  br i1 %tmp, label %then, label %else32

merge:                                            ; preds = %else32
  %src_x33 = load i32* %src_x1 dst_x34 = load i32* %dst_x3 tmp35 = icmp eq i32
  %%src_x33, %dst_x34
  br i1 %tmp35, label %then37, label %else67

then:                                             ; preds = %entry
  %src_x7 = load i32* %src_x1 dst_x8 = load i32* %dst_x3 tmp9 = icmp sgt i32
  %%src_x7, %dst_x8
  br i1 %tmp9, label %then11, label %else

merge10:                                          ; preds = %else, %then11
  %sx16 = load i32* %sx tmp17 = add i32 %sx16, 1
  store i32 %tmp17, i32* %tx br label %while

then11:                                           ; preds = %then
  %src_x12 = load i32* %src_x1
  store i32 %src_x12, i32* %gx
  %dst_x13 = load i32* %dst_x3
  store i32 %dst_x13, i32* %sx br label %merge10

else:                                             ; preds = %then
  %dst_x14 = load i32* %dst_x3
  store i32 %dst_x14, i32* %gx
  %src_x15 = load i32* %src_x1
  store i32 %src_x15, i32* %sx br label %merge10

while:                                            ; preds = %merge23, %merge10
  %tx28 = load i32* %tx gx29 = load i32* %gx tmp30 = icmp slt i32 %tx28, %gx29
  br i1 %tmp30, label %while_body, label %merge31

while_body:                                       ; preds = %while
  %tx18 = load i32* %tx src_y19 = load i32* %src_y2 name = getelementptr
  %inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32 %tx18, i32
  %%src_y19 name20 = load %GenericPiece** %name
  store %GenericPiece* %name20, %GenericPiece** %iter
  %iter21 = load %GenericPiece** %iter tmp22 = icmp ne %GenericPiece* %iter21,
  %null
  br i1 %tmp22, label %then24, label %else25

merge23:                                          ; preds = %else25
  %tx26 = load i32* %tx tmp27 = add i32 %tx26, 1
  store i32 %tmp27, i32* %tx br label %while

then24:                                           ; preds = %while_body ret i32
1

else25:                                           ; preds = %while_body br
```

```
label %merge23

merge31:                                          ; preds = %while ret i32 0

else32:                                           ; preds = %entry br label
%merge

merge36:                                          ; preds = %else67
  %src_x68 = load i32* %src_x1
  store i32 %src_x68, i32* %start_x
  %src_y69 = load i32* %src_y2
  store i32 %src_y69, i32* %start_y
  %dst_x70 = load i32* %dst_x3
  store i32 %dst_x70, i32* %end_x
  %dst_y71 = load i32* %dst_y4
  store i32 %dst_y71, i32* %end_y
  %dst_y72 = load i32* %dst_y4 src_y73 = load i32* %src_y2 tmp74 = sub i32
  %%dst_y72, %src_y73 dst_x75 = load i32* %dst_x3 src_x76 = load i32* %src_x1
  %tmp77 = sub i32 %dst_x75, %src_x76 tmp78 = sdiv i32 %tmp74, %tmp77 absl =
  %call i32 @abs(i32 %tmp78) tmp79 = icmp eq i32 %absl, 1
  br i1 %tmp79, label %then81, label %else206

then37:                                           ; preds = %merge
  %src_y38 = load i32* %src_y2 dst_y39 = load i32* %dst_y4 tmp40 = icmp sgt i32
  %%src_y38, %dst_y39
  br i1 %tmp40, label %then42, label %else45

merge41:                                          ; preds = %else45, %then42
  %sy48 = load i32* %sy tmp49 = add i32 %sy48, 1
  store i32 %tmp49, i32* %tx br label %while50

then42:                                           ; preds = %then37
  %src_y43 = load i32* %src_y2
  store i32 %src_y43, i32* %gy
  %dst_y44 = load i32* %dst_y4
  store i32 %dst_y44, i32* %sy br label %merge41

else45:                                           ; preds = %then37
  %dst_y46 = load i32* %dst_y4
  store i32 %dst_y46, i32* %gy
  %src_y47 = load i32* %src_y2
  store i32 %src_y47, i32* %sy br label %merge41

while50:                                          ; preds = %merge58, %merge41
  %tx63 = load i32* %tx gy64 = load i32* %gy tmp65 = icmp slt i32 %tx63, %gy64
  br i1 %tmp65, label %while_body51, label %merge66

while_body51:                                     ; preds = %while50
  %src_x52 = load i32* %src_x1 tx53 = load i32* %tx name54 = getelementptr
  %inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32 %src_x52, i32
  %%tx53 name55 = load %GenericPiece** %name54
  store %GenericPiece* %name55, %GenericPiece** %iter
  %iter56 = load %GenericPiece** %iter tmp57 = icmp ne %GenericPiece* %iter56,
  %null
```

```
  br i1 %tmp57, label %then59, label %else60

merge58:                                          ; preds = %else60
  %tx61 = load i32* %tx tmp62 = add i32 %tx61, 1
  store i32 %tmp62, i32* %tx br label %while50

then59:                                           ; preds = %while_body51 ret
i32 1

else60:                                           ; preds = %while_body51 br
label %merge58

merge66:                                          ; preds = %while50 ret i32 0

else67:                                           ; preds = %merge br label
%merge36

merge80:                                          ; preds = %else206, %merge182
ret i32 0

then81:                                           ; preds = %merge36
  %dst_x82 = load i32* %dst_x3 src_x83 = load i32* %src_x1 tmp84 = icmp sgt i32
  %%dst_x82, %src_x83 dst_y85 = load i32* %dst_y4 src_y86 = load i32* %src_y2
  %tmp87 = icmp sgt i32 %dst_y85, %src_y86 tmp88 = and i1 %tmp84, %tmp87
  br i1 %tmp88, label %then90, label %else112

merge89:                                          ; preds = %else112
  %dst_x113 = load i32* %dst_x3 src_x114 = load i32* %src_x1 tmp115 = icmp sgt
  %i32 %dst_x113, %src_x114 dst_y116 = load i32* %dst_y4 src_y117 = load i32*
  %%src_y2 tmp118 = icmp slt i32 %dst_y116, %src_y117 tmp119 = and i1 %tmp115,
  %%tmp118
  br i1 %tmp119, label %then121, label %else143

then90:                                           ; preds = %then81
  %start_x91 = load i32* %start_x tmp92 = add i32 %start_x91, 1
  store i32 %tmp92, i32* %tx br label %while93

while93:                                          ; preds = %merge103, %then90
  %tx108 = load i32* %tx end_x109 = load i32* %end_x tmp110 = icmp slt i32
  %%tx108, %end_x109
  br i1 %tmp110, label %while_body94, label %merge111

while_body94:                                     ; preds = %while93
  %start_y95 = load i32* %start_y tmp96 = add i32 %start_y95, 1
  store i32 %tmp96, i32* %start_y
  %tx97 = load i32* %tx start_y98 = load i32* %start_y name99 = getelementptr
  %inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32 %tx97, i32
  %%start_y98 name100 = load %GenericPiece** %name99
  store %GenericPiece* %name100, %GenericPiece** %iter
  %iter101 = load %GenericPiece** %iter tmp102 = icmp ne %GenericPiece*
  %%iter101, null
  br i1 %tmp102, label %then104, label %else105

merge103:                                         ; preds = %else105
```

```
  %tx106 = load i32* %tx tmp107 = add i32 %tx106, 1
  store i32 %tmp107, i32* %tx br label %while93

then104:                                          ; preds = %while_body94 ret
i32 1

else105:                                          ; preds = %while_body94 br
label %merge103

merge111:                                         ; preds = %while93 ret i32 0

else112:                                          ; preds = %then81 br label
%merge89

merge120:                                         ; preds = %else143
  %dst_x144 = load i32* %dst_x3 src_x145 = load i32* %src_x1 tmp146 = icmp slt
  %i32 %dst_x144, %src_x145 dst_y147 = load i32* %dst_y4 src_y148 = load i32*
  %%src_y2 tmp149 = icmp slt i32 %dst_y147, %src_y148 tmp150 = and i1 %tmp146,
  %%tmp149
  br i1 %tmp150, label %then152, label %else174

then121:                                          ; preds = %merge89
  %start_x122 = load i32* %start_x tmp123 = add i32 %start_x122, 1
  store i32 %tmp123, i32* %tx br label %while124

while124:                                         ; preds = %merge134, %then121
  %tx139 = load i32* %tx end_x140 = load i32* %end_x tmp141 = icmp slt i32
  %%tx139, %end_x140
  br i1 %tmp141, label %while_body125, label %merge142

while_body125:                                    ; preds = %while124
  %start_y126 = load i32* %start_y tmp127 = sub i32 %start_y126, 1
  store i32 %tmp127, i32* %start_y
  %tx128 = load i32* %tx start_y129 = load i32* %start_y name130 =
  %getelementptr inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32
  %%tx128, i32 %start_y129 name131 = load %GenericPiece** %name130
  store %GenericPiece* %name131, %GenericPiece** %iter
  %iter132 = load %GenericPiece** %iter tmp133 = icmp ne %GenericPiece*
  %%iter132, null
  br i1 %tmp133, label %then135, label %else136

merge134:                                         ; preds = %else136
  %tx137 = load i32* %tx tmp138 = add i32 %tx137, 1
  store i32 %tmp138, i32* %tx br label %while124

then135:                                          ; preds = %while_body125 ret
i32 1

else136:                                          ; preds = %while_body125 br
label %merge134

merge142:                                         ; preds = %while124 ret i32 0

else143:                                          ; preds = %merge89 br label
```

```
%merge120

merge151:                                         ; preds = %else174
  %dst_x175 = load i32* %dst_x3 src_x176 = load i32* %src_x1 tmp177 = icmp slt
  %i32 %dst_x175, %src_x176 dst_y178 = load i32* %dst_y4 src_y179 = load i32*
  %%src_y2 tmp180 = icmp sgt i32 %dst_y178, %src_y179 tmp181 = and i1 %tmp177,
  %%tmp180
   br i1 %tmp181, label %then183, label %else205

then152:                                          ; preds = %merge120
  %start_x153 = load i32* %start_x tmp154 = sub i32 %start_x153, 1
   store i32 %tmp154, i32* %tx br label %while155

while155:                                         ; preds = %merge165, %then152
  %tx170 = load i32* %tx end_x171 = load i32* %end_x tmp172 = icmp sgt i32
  %%tx170, %end_x171
   br i1 %tmp172, label %while_body156, label %merge173

while_body156:                                    ; preds = %while155
  %start_y157 = load i32* %start_y tmp158 = sub i32 %start_y157, 1
   store i32 %tmp158, i32* %start_y
  %tx159 = load i32* %tx start_y160 = load i32* %start_y name161 =
  %getelementptr inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32
  %%tx159, i32 %start_y160 name162 = load %GenericPiece** %name161
   store %GenericPiece* %name162, %GenericPiece** %iter
  %iter163 = load %GenericPiece** %iter tmp164 = icmp ne %GenericPiece*
  %%iter163, null
   br i1 %tmp164, label %then166, label %else167

merge165:                                         ; preds = %else167
  %tx168 = load i32* %tx tmp169 = sub i32 %tx168, 1
   store i32 %tmp169, i32* %tx br label %while155

then166:                                          ; preds = %while_body156 ret
i32 1

else167:                                          ; preds = %while_body156 br
label %merge165

merge173:                                         ; preds = %while155 ret i32 0

else174:                                          ; preds = %merge120 br label
%merge151

merge182:                                         ; preds = %else205 br label
%merge80

then183:                                          ; preds = %merge151
  %start_x184 = load i32* %start_x tmp185 = sub i32 %start_x184, 1
   store i32 %tmp185, i32* %tx br label %while186

while186:                                         ; preds = %merge196, %then183
  %tx201 = load i32* %tx end_x202 = load i32* %end_x tmp203 = icmp sgt i32
  %%tx201, %end_x202
```

```
  br i1 %tmp203, label %while_body187, label %merge204

while_body187:                                    ; preds = %while186
  %start_y188 = load i32* %start_y tmp189 = add i32 %start_y188, 1
  store i32 %tmp189, i32* %start_y
  %tx190 = load i32* %tx start_y191 = load i32* %start_y name192 =
  %getelementptr inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32
  %%tx190, i32 %start_y191 name193 = load %GenericPiece** %name192
  store %GenericPiece* %name193, %GenericPiece** %iter
  %iter194 = load %GenericPiece** %iter tmp195 = icmp ne %GenericPiece*
  %%iter194, null
  br i1 %tmp195, label %then197, label %else198

merge196:                                         ; preds = %else198
  %tx199 = load i32* %tx tmp200 = sub i32 %tx199, 1
  store i32 %tmp200, i32* %tx br label %while186

then197:                                          ; preds = %while_body187 ret
i32 1

else198:                                          ; preds = %while_body187 br
label %merge196

merge204:                                         ; preds = %while186 ret i32 0

else205:                                          ; preds = %merge151 br label
%merge182

else206:                                          ; preds = %merge36 br label
%merge80 }

define i32 @checkBound(i32 %x, i32 %y) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %repeat = alloca i32 x3 = load i32* %x1 tmp = icmp sgt i32 %x3, -1 x4 = load
  %i32* %x1 rows = load i32* @rows tmp5 = icmp sle i32 %x4, %rows tmp6 = and i1
  %%tmp, %tmp5 y7 = load i32* %y2 tmp8 = icmp sgt i32 %y7, -1 tmp9 = and i1
  %%tmp6, %tmp8 y10 = load i32* %y2 cols = load i32* @cols tmp11 = icmp sle i32
  %%y10, %cols tmp12 = and i1 %tmp9, %tmp11
  br i1 %tmp12, label %then, label %else

merge:                                            ; No predecessors!  ret i32 0

then:                                             ; preds = %entry ret i32 1

else:                                             ; preds = %entry ret i32 0 }

define %GenericPiece* @getPieceFromGrid(i8* %displayString) { entry:
  %displayString1 = alloca i8*
  store i8* %displayString, i8** %displayString1
  %iterator = alloca %GenericPiece* x = alloca i32 y = alloca i32 repeat =
  %alloca i32
```

```
  store i32 0, i32* %x br label %while

while:                                            ; preds = %merge23, %entry
  %x26 = load i32* %x rows = load i32* @rows tmp27 = icmp slt i32 %x26, %rows
  br i1 %tmp27, label %while_body, label %merge28

while_body:                                       ; preds = %while store i32 0,
i32* %y br label %while2

while2:                                           ; preds = %merge18,
%while_body
  %y21 = load i32* %y cols = load i32* @cols tmp22 = icmp slt i32 %y21, %cols
  br i1 %tmp22, label %while_body3, label %merge23

while_body3:                                      ; preds = %while2
  %x4 = load i32* %x y5 = load i32* %y name = getelementptr inbounds [6 x [6 x
  %%GenericPiece*]]* @GridData, i32 0, i32 %x4, i32 %y5 name6 = load
  %%GenericPiece** %name
  store %GenericPiece* %name6, %GenericPiece** %iterator br label %while7

while7:                                           ; preds = %merge,
%while_body3
  %iterator16 = load %GenericPiece** %iterator tmp17 = icmp ne %GenericPiece*
  %%iterator16, null
  br i1 %tmp17, label %while_body8, label %merge18

while_body8:                                      ; preds = %while7
  %iterator9 = load %GenericPiece** %iterator loaded_deref = load
  %%GenericPiece* %iterator9 dotop_terminal = getelementptr inbounds
  %%GenericPiece* %iterator9, i32 0, i32 2 loaded_dotop_terminal = load i8**
  %%dotop_terminal displayString10 = load i8** %displayString1 tmp = icmp eq
  %i8* %loaded_dotop_terminal, %displayString10
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else
  %iterator12 = load %GenericPiece** %iterator loaded_deref13 = load
  %%GenericPiece* %iterator12 dotop_terminal14 = getelementptr inbounds
  %%GenericPiece* %iterator12, i32 0, i32 3 loaded_dotop_terminal15 = load
  %%GenericPiece** %dotop_terminal14
  store %GenericPiece* %loaded_dotop_terminal15, %GenericPiece** %iterator br
label %while7

then:                                             ; preds = %while_body8
  %iterator11 = load %GenericPiece** %iterator
  ret %GenericPiece* %iterator11

else:                                             ; preds = %while_body8 br
label %merge

merge18:                                          ; preds = %while7
  %y19 = load i32* %y tmp20 = add i32 %y19, 1
  store i32 %tmp20, i32* %y br label %while2

merge23:                                          ; preds = %while2
```

```
  %x24 = load i32* %x tmp25 = add i32 %x24, 1
  store i32 %tmp25, i32* %x br label %while

merge28:                                          ; preds = %while store
%GenericPiece* null, %GenericPiece** %iterator
  %iterator29 = load %GenericPiece** %iterator
  ret %GenericPiece* %iterator29 }

define %GenericPiece* @getPieceAtLocation(i32 %x, i32 %y) { entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %head = alloca %GenericPiece* repeat = alloca i32 x3 = load i32* %x1 y4 =
  %load i32* %y2 name = getelementptr inbounds [6 x [6 x %GenericPiece*]]*
  %@GridData, i32 0, i32 %x3, i32 %y4 name5 = load %GenericPiece** %name
  store %GenericPiece* %name5, %GenericPiece** %head
  %head6 = load %GenericPiece** %head
  ret %GenericPiece* %head6 }

define i32 @moveOnGrid(%GenericPiece* %p_n, i32 %dst_x, i32 %dst_y) { entry:
  %p_n1 = alloca %GenericPiece*
  store %GenericPiece* %p_n, %GenericPiece** %p_n1
  %dst_x2 = alloca i32
  store i32 %dst_x, i32* %dst_x2
  %dst_y3 = alloca i32
  store i32 %dst_y, i32* %dst_y3
  %result = alloca i32 src_x = alloca i32 src_y = alloca i32 repeat = alloca
  %i32 p_n4 = load %GenericPiece** %p_n1 tmp = icmp ne %GenericPiece* %p_n4,
  %null
  br i1 %tmp, label %then, label %else32

merge:                                            ; preds = %merge20
  %result35 = load i32* %result
  ret i32 %result35

then:                                             ; preds = %entry
  %p_n5 = load %GenericPiece** %p_n1 loaded_deref = load %GenericPiece* %p_n5
  %dotop_terminal = getelementptr inbounds %GenericPiece* %p_n5, i32 0, i32 5
  %loaded_dotop_terminal = load i32* %dotop_terminal
  store i32 %loaded_dotop_terminal, i32* %src_x
  %p_n6 = load %GenericPiece** %p_n1 loaded_deref7 = load %GenericPiece* %p_n6
  %dotop_terminal8 = getelementptr inbounds %GenericPiece* %p_n6, i32 0, i32 4
  %loaded_dotop_terminal9 = load i32* %dotop_terminal8
  store i32 %loaded_dotop_terminal9, i32* %src_y
  %p_n10 = load %GenericPiece** %p_n1 loaded_deref11 = load %GenericPiece*
  %%p_n10 dotop_terminal12 = getelementptr inbounds %GenericPiece* %p_n10, i32
  %0, i32 1 loaded_dotop_terminal13 = load i8** %dotop_terminal12 dst_y14 =
  %load i32* %dst_y3 dst_x15 = load i32* %dst_x2 src_y16 = load i32* %src_y
  %src_x17 = load i32* %src_x triggerRule_result = call i32 @triggerRule(i32
  %%src_x17, i32 %src_y16, i32 %dst_x15, i32 %dst_y14, i8*
  %%loaded_dotop_terminal13)
  store i32 %triggerRule_result, i32* %result
  %result18 = load i32* %result tmp19 = icmp eq i32 %result18, 1
```

```
  br i1 %tmp19, label %then21, label %else

merge20:                                          ; preds = %else, %then21 br
label %merge

then21:                                           ; preds = %then
  %p_n22 = load %GenericPiece** %p_n1 loaded_deref23 = load %GenericPiece*
  %%p_n22 dotop_terminal24 = getelementptr inbounds %GenericPiece* %p_n22, i32
  %0, i32 2 loaded_dotop_terminal25 = load i8** %dotop_terminal24 src_y26 =
  %load i32* %src_y src_x27 = load i32* %src_x
   call void @deleteFromGrid(i32 %src_x27, i32 %src_y26, i8*
%loaded_dotop_terminal25)
  %p_n28 = load %GenericPiece** %p_n1 dst_y29 = load i32* %dst_y3 dst_x30 =
  %load i32* %dst_x2
   call void @addToGrid(i32 %dst_x30, i32 %dst_y29, %GenericPiece* %p_n28) br
label %merge20

else:                                             ; preds = %then
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp31 = sub i32
  %%currentPlayerIndex, 1
   store i32 %tmp31, i32* @currentPlayerIndex br label %merge20

else32:                                           ; preds = %entry
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt30, i32 0, i32 0), i8* getelementptr inbounds ([17 x i8]* @name32, i32
  %0, i32 0)) currentPlayerIndex33 = load i32* @currentPlayerIndex tmp34 = sub
  %i32 %currentPlayerIndex33, 1
   store i32 %tmp34, i32* @currentPlayerIndex ret i32 0 }

define i32 @printGrid() { entry:
  %x = alloca i32 y = alloca i32 i = alloca i32 k = alloca i32 width = alloca
  %i32 tempLen = alloca i32 flag = alloca i32 max_width = alloca i32 border_len
  %= alloca i32 printer = alloca i8* iterator = alloca %GenericPiece* repeat =
  %alloca i32
   store i32 0, i32* %width store i32 0, i32* %max_width store i32 0, i32*
%border_len store i32 0, i32* %x br label %while

while:                                            ; preds = %merge43, %entry
  %x46 = load i32* %x rows = load i32* @rows tmp47 = icmp slt i32 %x46, %rows
   br i1 %tmp47, label %while_body, label %merge48

while_body:                                       ; preds = %while store i32 0,
i32* %y br label %while1

while1:                                           ; preds = %merge35,
%while_body
  %y41 = load i32* %y cols = load i32* @cols tmp42 = icmp slt i32 %y41, %cols
   br i1 %tmp42, label %while_body2, label %merge43

while_body2:                                      ; preds = %while1
  %x3 = load i32* %x y4 = load i32* %y name = getelementptr inbounds [6 x [6 x
  %%GenericPiece*]]* @GridData, i32 0, i32 %x3, i32 %y4 name5 = load
  %%GenericPiece** %name
   store %GenericPiece* %name5, %GenericPiece** %iterator store i32 0, i32*
```

```
%width
  %iterator6 = load %GenericPiece** %iterator tmp = icmp ne %GenericPiece*
  %%iterator6, null
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then br
label %while14

then:                                             ; preds = %while_body2
  %width7 = load i32* %width iterator8 = load %GenericPiece** %iterator
  %loaded_deref = load %GenericPiece* %iterator8 dotop_terminal = getelementptr
  %inbounds %GenericPiece* %iterator8, i32 0, i32 2 loaded_dotop_terminal =
  %load i8** %dotop_terminal getLen = call i32 @getLen(i8*
  %%loaded_dotop_terminal) tmp9 = add i32 %width7, %getLen
  store i32 %tmp9, i32* %width
  %iterator10 = load %GenericPiece** %iterator loaded_deref11 = load
  %%GenericPiece* %iterator10 dotop_terminal12 = getelementptr inbounds
  %%GenericPiece* %iterator10, i32 0, i32 3 loaded_dotop_terminal13 = load
  %%GenericPiece** %dotop_terminal12
  store %GenericPiece* %loaded_dotop_terminal13, %GenericPiece** %iterator br
label %merge

else:                                             ; preds = %while_body2 br
label %merge

while14:                                          ; preds = %while_body15,
%merge
  %iterator29 = load %GenericPiece** %iterator tmp30 = icmp ne %GenericPiece*
  %%iterator29, null
  br i1 %tmp30, label %while_body15, label %merge31

while_body15:                                     ; preds = %while14
  %width16 = load i32* %width tmp17 = add i32 %width16, 2
  store i32 %tmp17, i32* %width
  %width18 = load i32* %width iterator19 = load %GenericPiece** %iterator
  %loaded_deref20 = load %GenericPiece* %iterator19 dotop_terminal21 =
  %getelementptr inbounds %GenericPiece* %iterator19, i32 0, i32 2
  %loaded_dotop_terminal22 = load i8** %dotop_terminal21 getLen23 = call i32
  %@getLen(i8* %loaded_dotop_terminal22) tmp24 = add i32 %width18, %getLen23
  store i32 %tmp24, i32* %width
  %iterator25 = load %GenericPiece** %iterator loaded_deref26 = load
  %%GenericPiece* %iterator25 dotop_terminal27 = getelementptr inbounds
  %%GenericPiece* %iterator25, i32 0, i32 3 loaded_dotop_terminal28 = load
  %%GenericPiece** %dotop_terminal27
  store %GenericPiece* %loaded_dotop_terminal28, %GenericPiece** %iterator br
label %while14

merge31:                                          ; preds = %while14
  %width32 = load i32* %width max_width33 = load i32* %max_width tmp34 = icmp
  %sgt i32 %width32, %max_width33
  br i1 %tmp34, label %then36, label %else38

merge35:                                          ; preds = %else38, %then36
  %y39 = load i32* %y tmp40 = add i32 %y39, 1
```

```
  store i32 %tmp40, i32* %y br label %while1

then36:                                           ; preds = %merge31
  %width37 = load i32* %width
  store i32 %width37, i32* %max_width br label %merge35

else38:                                           ; preds = %merge31 br label
%merge35

merge43:                                          ; preds = %while1
  %x44 = load i32* %x tmp45 = add i32 %x44, 1
  store i32 %tmp45, i32* %x br label %while

merge48:                                          ; preds = %while
  %max_width49 = load i32* %max_width cols50 = load i32* @cols tmp51 = mul i32
  %%max_width49, %cols50 cols52 = load i32* @cols tmp53 = add i32 %tmp51,
  %%cols52
  store i32 %tmp53, i32* %border_len store i32 0, i32* %i br label %while54

while54:                                          ; preds = %while_body55,
%merge48
  %i58 = load i32* %i border_len59 = load i32* %border_len tmp60 = icmp slt i32
  %%i58, %border_len59
  br i1 %tmp60, label %while_body55, label %merge61

while_body55:                                     ; preds = %while54
  %print_sameline = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name35, i32 0, i32 0)) i56 = load i32* %i tmp57 = add i32 %i56, 1
  store i32 %tmp57, i32* %i br label %while54

merge61:                                          ; preds = %while54
  %print_endline = call i32 @print_endline()
  store i32 0, i32* %i br label %while62

while62:                                          ; preds = %merge86, %merge61
  %i89 = load i32* %i cols90 = load i32* @cols tmp91 = icmp slt i32 %i89,
  %%cols90
  br i1 %tmp91, label %while_body63, label %merge92

while_body63:                                     ; preds = %while62 store i32
0, i32* %k br label %while64

while64:                                          ; preds = %while_body65,
%while_body63
  %k69 = load i32* %k max_width70 = load i32* %max_width tmp71 = sdiv i32
  %%max_width70, 2 tmp72 = icmp slt i32 %k69, %tmp71
  br i1 %tmp72, label %while_body65, label %merge73

while_body65:                                     ; preds = %while64
  %print_sameline66 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name36, i32 0, i32 0)) k67 = load i32* %k tmp68 = add i32 %k67, 1
  store i32 %tmp68, i32* %k br label %while64

merge73:                                          ; preds = %while64
```

```
  %i74 = load i32* %i print_int_sameline = call i32 @print_int_sameline(i32
  %%i74)
   store i32 0, i32* %k br label %while75

while75:                                          ; preds = %while_body76,
%merge73
  %k80 = load i32* %k max_width81 = load i32* %max_width max_width82 = load
  %i32* %max_width tmp83 = sdiv i32 %max_width82, 2 tmp84 = sub i32
  %%max_width81, %tmp83 tmp85 = icmp slt i32 %k80, %tmp84
   br i1 %tmp85, label %while_body76, label %merge86

while_body76:                                     ; preds = %while75
  %print_sameline77 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name37, i32 0, i32 0)) k78 = load i32* %k tmp79 = add i32 %k78, 1
   store i32 %tmp79, i32* %k br label %while75

merge86:                                          ; preds = %while75
  %i87 = load i32* %i tmp88 = add i32 %i87, 1
   store i32 %tmp88, i32* %i br label %while62

merge92:                                          ; preds = %while62
  %print_endline93 = call i32 @print_endline()
   store i32 0, i32* %x br label %while94

while94:                                          ; preds = %merge160, %merge92
  %x167 = load i32* %x rows168 = load i32* @rows tmp169 = icmp slt i32 %x167,
  %%rows168
   br i1 %tmp169, label %while_body95, label %merge170

while_body95:                                     ; preds = %while94 store i32
0, i32* %y br label %while96

while96:                                          ; preds = %merge154,
%while_body95
  %y157 = load i32* %y cols158 = load i32* @cols tmp159 = icmp slt i32 %y157,
  %%cols158
   br i1 %tmp159, label %while_body97, label %merge160

while_body97:                                     ; preds = %while96 store i32
0, i32* %tempLen
  %print_sameline98 = call i32 @print_sameline(i8* getelementptr inbounds ([2 x
  %i8]* @name38, i32 0, i32 0)) x99 = load i32* %x y100 = load i32* %y name101
  %= getelementptr inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32
  %%x99, i32 %y100 name102 = load %GenericPiece** %name101
   store %GenericPiece* %name102, %GenericPiece** %iterator
  %iterator103 = load %GenericPiece** %iterator tmp104 = icmp ne %GenericPiece*
  %%iterator103, null
   br i1 %tmp104, label %then106, label %else121

merge105:                                         ; preds = %else121, %then106
br label %while122

then106:                                          ; preds = %while_body97
  %iterator107 = load %GenericPiece** %iterator loaded_deref108 = load
```

```
  %%GenericPiece* %iterator107 dotop_terminal109 = getelementptr inbounds
  %%GenericPiece* %iterator107, i32 0, i32 2 loaded_dotop_terminal110 = load
  %i8** %dotop_terminal109
   store i8* %loaded_dotop_terminal110, i8** %printer
  %tempLen111 = load i32* %tempLen printer112 = load i8** %printer getLen113 =
  %call i32 @getLen(i8* %printer112) tmp114 = add i32 %tempLen111, %getLen113
   store i32 %tmp114, i32* %tempLen
  %printer115 = load i8** %printer print_sameline116 = call i32
  %@print_sameline(i8* %printer115) iterator117 = load %GenericPiece**
  %%iterator loaded_deref118 = load %GenericPiece* %iterator117
  %dotop_terminal119 = getelementptr inbounds %GenericPiece* %iterator117, i32
   %0, i32 3 loaded_dotop_terminal120 = load %GenericPiece** %dotop_terminal119
   store %GenericPiece* %loaded_dotop_terminal120, %GenericPiece** %iterator br
label %merge105

else121:                                          ; preds = %while_body97 br
label %merge105

while122:                                         ; preds = %while_body123,
%merge105
  %iterator141 = load %GenericPiece** %iterator tmp142 = icmp ne %GenericPiece*
  %%iterator141, null
   br i1 %tmp142, label %while_body123, label %merge143

while_body123:                                    ; preds = %while122
  %print_sameline124 = call i32 @print_sameline(i8* getelementptr inbounds ([3
  %x i8]* @name39, i32 0, i32 0)) iterator125 = load %GenericPiece** %iterator
  %loaded_deref126 = load %GenericPiece* %iterator125 dotop_terminal127 =
  %getelementptr inbounds %GenericPiece* %iterator125, i32 0, i32 2
  %loaded_dotop_terminal128 = load i8** %dotop_terminal127
   store i8* %loaded_dotop_terminal128, i8** %printer
  %tempLen129 = load i32* %tempLen printer130 = load i8** %printer getLen131 =
  %call i32 @getLen(i8* %printer130) tmp132 = add i32 %tempLen129, %getLen131
   store i32 %tmp132, i32* %tempLen
  %tempLen133 = load i32* %tempLen tmp134 = add i32 %tempLen133, 2
   store i32 %tmp134, i32* %tempLen
  %printer135 = load i8** %printer print_sameline136 = call i32
  %@print_sameline(i8* %printer135) iterator137 = load %GenericPiece**
  %%iterator loaded_deref138 = load %GenericPiece* %iterator137
  %dotop_terminal139 = getelementptr inbounds %GenericPiece* %iterator137, i32
   %0, i32 3 loaded_dotop_terminal140 = load %GenericPiece** %dotop_terminal139
   store %GenericPiece* %loaded_dotop_terminal140, %GenericPiece** %iterator br
label %while122

merge143:                                         ; preds = %while122 store i32
0, i32* %k br label %while144

while144:                                         ; preds = %while_body145,
%merge143
  %k149 = load i32* %k max_width150 = load i32* %max_width tempLen151 = load
  %i32* %tempLen tmp152 = sub i32 %max_width150, %tempLen151 tmp153 = icmp slt
  %i32 %k149, %tmp152
   br i1 %tmp153, label %while_body145, label %merge154
```

```
while_body145:                                    ; preds = %while144
  %print_sameline146 = call i32 @print_sameline(i8* getelementptr inbounds ([2
  %x i8]* @name40, i32 0, i32 0)) k147 = load i32* %k tmp148 = add i32 %k147, 1
   store i32 %tmp148, i32* %k br label %while144

merge154:                                         ; preds = %while144
  %y155 = load i32* %y tmp156 = add i32 %y155, 1
   store i32 %tmp156, i32* %y br label %while96

merge160:                                         ; preds = %while96
  %print_sameline161 = call i32 @print_sameline(i8* getelementptr inbounds ([2
  %x i8]* @name41, i32 0, i32 0)) x162 = load i32* %x print_int_sameline163 =
  %call i32 @print_int_sameline(i32 %x162) print_endline164 = call i32
  %@print_endline() x165 = load i32* %x tmp166 = add i32 %x165, 1
   store i32 %tmp166, i32* %x br label %while94

merge170:                                         ; preds = %while94 store i32
0, i32* %i br label %while171

while171:                                         ; preds = %while_body172,
%merge170
  %i176 = load i32* %i border_len177 = load i32* %border_len tmp178 = icmp slt
  %i32 %i176, %border_len177
   br i1 %tmp178, label %while_body172, label %merge179

while_body172:                                    ; preds = %while171
  %print_sameline173 = call i32 @print_sameline(i8* getelementptr inbounds ([2
  %x i8]* @name42, i32 0, i32 0)) i174 = load i32* %i tmp175 = add i32 %i174, 1
   store i32 %tmp175, i32* %i br label %while171

merge179:                                         ; preds = %while171
  %print_endline180 = call i32 @print_endline()
   ret i32 0 }

define void @deleteFromGrid(i32 %x, i32 %y, i8* %tag) { entry:
  %x1 = alloca i32
   store i32 %x, i32* %x1
  %y2 = alloca i32
   store i32 %y, i32* %y2
  %tag3 = alloca i8*
   store i8* %tag, i8** %tag3
  %iterator = alloca %GenericPiece* next_iterator = alloca %GenericPiece*
  %repeat = alloca i32 x4 = load i32* %x1 y5 = load i32* %y2 name =
  %getelementptr inbounds [6 x [6 x %GenericPiece*]]* @GridData, i32 0, i32
  %%x4, i32 %y5 name6 = load %GenericPiece** %name
   store %GenericPiece* %name6, %GenericPiece** %iterator
  %iterator7 = load %GenericPiece** %iterator loaded_deref = load
  %%GenericPiece* %iterator7 dotop_terminal = getelementptr inbounds
  %%GenericPiece* %iterator7, i32 0, i32 3 loaded_dotop_terminal = load
  %%GenericPiece** %dotop_terminal
   store %GenericPiece* %loaded_dotop_terminal, %GenericPiece** %next_iterator
  %iterator8 = load %GenericPiece** %iterator loaded_deref9 = load
  %%GenericPiece* %iterator8 dotop_terminal10 = getelementptr inbounds
  %%GenericPiece* %iterator8, i32 0, i32 2 loaded_dotop_terminal11 = load i8**
```

```
  %%dotop_terminal10 tag12 = load i8** %tag3 tmp = icmp eq i8*
  %%loaded_dotop_terminal11, %tag12
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else br label
%while

then:                                             ; preds = %entry
  %x13 = load i32* %x1 y14 = load i32* %y2 name15 = getelementptr inbounds [6 x
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x13, i32 %y14 iterator16 =
  %load %GenericPiece** %iterator loaded_deref17 = load %GenericPiece*
  %%iterator16 dotop_terminal18 = getelementptr inbounds %GenericPiece*
  %%iterator16, i32 0, i32 3 loaded_dotop_terminal19 = load %GenericPiece**
  %%dotop_terminal18
  store %GenericPiece* %loaded_dotop_terminal19, %GenericPiece** %name15
  %iterator20 = load %GenericPiece** %iterator loaded_deref21 = load
  %%GenericPiece* %iterator20 next = getelementptr inbounds %GenericPiece*
  %%iterator20, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next store %GenericPiece* null,
%GenericPiece** %iterator ret void

else:                                             ; preds = %entry br label
%merge

while:                                            ; preds = %merge28, %merge
  %iterator49 = load %GenericPiece** %iterator loaded_deref50 = load
  %%GenericPiece* %iterator49 dotop_terminal51 = getelementptr inbounds
  %%GenericPiece* %iterator49, i32 0, i32 3 loaded_dotop_terminal52 = load
  %%GenericPiece** %dotop_terminal51 tmp53 = icmp ne %GenericPiece*
  %%loaded_dotop_terminal52, null
  br i1 %tmp53, label %while_body, label %merge54

while_body:                                       ; preds = %while
  %next_iterator22 = load %GenericPiece** %next_iterator loaded_deref23 = load
  %%GenericPiece* %next_iterator22 dotop_terminal24 = getelementptr inbounds
  %%GenericPiece* %next_iterator22, i32 0, i32 2 loaded_dotop_terminal25 = load
  %i8** %dotop_terminal24 tag26 = load i8** %tag3 tmp27 = icmp eq i8*
  %%loaded_dotop_terminal25, %tag26
  br i1 %tmp27, label %then29, label %else40

merge28:                                          ; preds = %else40
  %next_iterator41 = load %GenericPiece** %next_iterator loaded_deref42 = load
  %%GenericPiece* %next_iterator41 dotop_terminal43 = getelementptr inbounds
  %%GenericPiece* %next_iterator41, i32 0, i32 3 loaded_dotop_terminal44 = load
  %%GenericPiece** %dotop_terminal43
  store %GenericPiece* %loaded_dotop_terminal44, %GenericPiece** %next_iterator
  %iterator45 = load %GenericPiece** %iterator loaded_deref46 = load
  %%GenericPiece* %iterator45 dotop_terminal47 = getelementptr inbounds
  %%GenericPiece* %iterator45, i32 0, i32 3 loaded_dotop_terminal48 = load
  %%GenericPiece** %dotop_terminal47
  store %GenericPiece* %loaded_dotop_terminal48, %GenericPiece** %iterator br
label %while

then29:                                           ; preds = %while_body
```

```
  %next_iterator30 = load %GenericPiece** %next_iterator loaded_deref31 = load
  %%GenericPiece* %next_iterator30 dotop_terminal32 = getelementptr inbounds
  %%GenericPiece* %next_iterator30, i32 0, i32 3 loaded_dotop_terminal33 = load
  %%GenericPiece** %dotop_terminal32 iterator34 = load %GenericPiece**
  %%iterator loaded_deref35 = load %GenericPiece* %iterator34 next36 =
  %getelementptr inbounds %GenericPiece* %iterator34, i32 0, i32 3
   store %GenericPiece* %loaded_dotop_terminal33, %GenericPiece** %next36
  %next_iterator37 = load %GenericPiece** %next_iterator loaded_deref38 = load
  %%GenericPiece* %next_iterator37 next39 = getelementptr inbounds
  %%GenericPiece* %next_iterator37, i32 0, i32 3
   store %GenericPiece* null, %GenericPiece** %next39 store %GenericPiece* null,
%GenericPiece** %next_iterator ret void

else40:                                           ; preds = %while_body br
label %merge28

merge54:                                          ; preds = %while
  %iterator55 = load %GenericPiece** %iterator loaded_deref56 = load
  %%GenericPiece* %iterator55 dotop_terminal57 = getelementptr inbounds
  %%GenericPiece* %iterator55, i32 0, i32 3 loaded_dotop_terminal58 = load
  %%GenericPiece** %dotop_terminal57 tmp59 = icmp eq %GenericPiece*
  %%loaded_dotop_terminal58, null
   br i1 %tmp59, label %then61, label %else62

merge60:                                          ; preds = %else62 ret void

then61:                                           ; preds = %merge54
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt44, i32 0, i32 0), i8* getelementptr inbounds ([30 x i8]* @name46, i32
  %0, i32 0))
   ret void

else62:                                           ; preds = %merge54 br label
%merge60 }

define void @addToGrid(i32 %x, i32 %y, %GenericPiece* %p_n) { entry:
  %x1 = alloca i32
   store i32 %x, i32* %x1
  %y2 = alloca i32
   store i32 %y, i32* %y2
  %p_n3 = alloca %GenericPiece*
   store %GenericPiece* %p_n, %GenericPiece** %p_n3
  %iterator = alloca %GenericPiece* repeat = alloca i32 x4 = load i32* %x1 p_n5
  %= load %GenericPiece** %p_n3 loaded_deref = load %GenericPiece* %p_n5 x6 =
  %getelementptr inbounds %GenericPiece* %p_n5, i32 0, i32 5
   store i32 %x4, i32* %x6
  %y7 = load i32* %y2 p_n8 = load %GenericPiece** %p_n3 loaded_deref9 = load
  %%GenericPiece* %p_n8 y10 = getelementptr inbounds %GenericPiece* %p_n8, i32
  %0, i32 4
   store i32 %y7, i32* %y10
  %x11 = load i32* %x1 y12 = load i32* %y2 name = getelementptr inbounds [6 x
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x11, i32 %y12 name13 = load
  %%GenericPiece** %name tmp = icmp eq %GenericPiece* %name13, null
   br i1 %tmp, label %then, label %else
```

```
merge:                                            ; preds = %merge69 ret void

then:                                             ; preds = %entry
  %x14 = load i32* %x1 y15 = load i32* %y2 name16 = getelementptr inbounds [6 x
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x14, i32 %y15 p_n17 = load
  %%GenericPiece** %p_n3
  store %GenericPiece* %p_n17, %GenericPiece** %name16
  %x18 = load i32* %x1 y19 = load i32* %y2 name20 = getelementptr inbounds [6 x
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x18, i32 %y19 name21 = load
  %%GenericPiece** %name20
  store %GenericPiece* %name21, %GenericPiece** %iterator
  %iterator22 = load %GenericPiece** %iterator loaded_deref23 = load
  %%GenericPiece* %iterator22 next = getelementptr inbounds %GenericPiece*
  %%iterator22, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next ret void

else:                                             ; preds = %entry
  %x24 = load i32* %x1 y25 = load i32* %y2 name26 = getelementptr inbounds [6 x
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x24, i32 %y25 name27 = load
  %%GenericPiece** %name26
  store %GenericPiece* %name27, %GenericPiece** %iterator br label %while

while:                                            ; preds = %while_body, %else
  %iterator30 = load %GenericPiece** %iterator loaded_deref31 = load
  %%GenericPiece* %iterator30 dotop_terminal32 = getelementptr inbounds
  %%GenericPiece* %iterator30, i32 0, i32 3 loaded_dotop_terminal33 = load
  %%GenericPiece** %dotop_terminal32 tmp34 = icmp ne %GenericPiece*
  %%loaded_dotop_terminal33, null
  br i1 %tmp34, label %while_body, label %merge35

while_body:                                       ; preds = %while
  %iterator28 = load %GenericPiece** %iterator loaded_deref29 = load
  %%GenericPiece* %iterator28 dotop_terminal = getelementptr inbounds
  %%GenericPiece* %iterator28, i32 0, i32 3 loaded_dotop_terminal = load
  %%GenericPiece** %dotop_terminal
  store %GenericPiece* %loaded_dotop_terminal, %GenericPiece** %iterator br
label %while

merge35:                                          ; preds = %while
  %p_n36 = load %GenericPiece** %p_n3 iterator37 = load %GenericPiece**
  %%iterator loaded_deref38 = load %GenericPiece* %iterator37 next39 =
  %getelementptr inbounds %GenericPiece* %iterator37, i32 0, i32 3
  store %GenericPiece* %p_n36, %GenericPiece** %next39
  %iterator40 = load %GenericPiece** %iterator loaded_deref41 = load
  %%GenericPiece* %iterator40 dotop_terminal42 = getelementptr inbounds
  %%GenericPiece* %iterator40, i32 0, i32 3 loaded_dotop_terminal43 = load
  %%GenericPiece** %dotop_terminal42
  store %GenericPiece* %loaded_dotop_terminal43, %GenericPiece** %iterator
  %iterator44 = load %GenericPiece** %iterator loaded_deref45 = load
  %%GenericPiece* %iterator44 next46 = getelementptr inbounds %GenericPiece*
  %%iterator44, i32 0, i32 3
  store %GenericPiece* null, %GenericPiece** %next46
  %x47 = load i32* %x1 y48 = load i32* %y2 name49 = getelementptr inbounds [6 x
```

```
  %[6 x %GenericPiece*]]* @GridData, i32 0, i32 %x47, i32 %y48 name50 = load
  %%GenericPiece** %name49
   store %GenericPiece* %name50, %GenericPiece** %iterator br label %while51

while51:                                          ; preds = %merge56, %merge35
  %iterator67 = load %GenericPiece** %iterator tmp68 = icmp ne %GenericPiece*
  %%iterator67, null
   br i1 %tmp68, label %while_body52, label %merge69

while_body52:                                     ; preds = %while51
  %iterator53 = load %GenericPiece** %iterator p_n54 = load %GenericPiece**
  %%p_n3 tmp55 = icmp ne %GenericPiece* %iterator53, %p_n54
   br i1 %tmp55, label %then57, label %else62

merge56:                                          ; preds = %else62, %then57
  %iterator63 = load %GenericPiece** %iterator loaded_deref64 = load
  %%GenericPiece* %iterator63 dotop_terminal65 = getelementptr inbounds
  %%GenericPiece* %iterator63, i32 0, i32 3 loaded_dotop_terminal66 = load
  %%GenericPiece** %dotop_terminal65
   store %GenericPiece* %loaded_dotop_terminal66, %GenericPiece** %iterator br
label %while51

then57:                                           ; preds = %while_body52
  %iterator58 = load %GenericPiece** %iterator p_n59 = load %GenericPiece**
  %%p_n3 y60 = load i32* %y2 x61 = load i32* %x1 colocation_result = call i32
  %@colocation(i32 %x61, i32 %y60, %GenericPiece* %p_n59, %GenericPiece*
  %%iterator58)
   br label %merge56

else62:                                           ; preds = %while_body52 br
label %merge56

merge69:                                          ; preds = %while51 br label
%merge }

define i32 @gameloop() { entry:
  %dice = alloca i32 repeat = alloca i32
   store i32 0, i32* %repeat store i32 0, i32* @currentPlayerIndex br label
%while

while:                                            ; preds = %while_body, %entry
  %repeat3 = load i32* %repeat tmp4 = icmp eq i32 %repeat3, 0
   br i1 %tmp4, label %while_body, label %merge

while_body:                                       ; preds = %while
  %printGrid_result = call i32 @printGrid() prompt = call i32 @prompt(i8*
  %getelementptr inbounds ([4 x i8]* @name51, i32 0, i32 0)) diceThrow = call
  %i32 @diceThrow()
   store i32 %diceThrow, i32* %dice
  %dice1 = load i32* %dice makeMove_result = call i32 @makeMove(i32 %dice1)
  %checkGameEnd_result = call i32 @checkGameEnd()
   store i32 %checkGameEnd_result, i32* %repeat
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp = add i32
  %%currentPlayerIndex, 1 playerOrderSize = load i32* @playerOrderSize tmp2 =
```

```
  %urem i32 %tmp, %playerOrderSize
  store i32 %tmp2, i32* @currentPlayerIndex br label %while

merge:                                            ; preds = %while ret i32 0 }

define i32 @makeMove(i32 %dice) { entry:
  %dice1 = alloca i32
  store i32 %dice, i32* %dice1
  %i = alloca i32 src_x = alloca i32 src_y = alloca i32 dst_x = alloca i32
  %dst_y = alloca i32 direction = alloca i32 t = alloca %Token* token = alloca
  %%GenericPiece* repeat = alloca i32 currentPlayerIndex = load i32*
  %@currentPlayerIndex tmp = icmp eq i32 %currentPlayerIndex, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %t3 = load %Token** %t loaded_deref = load %Token* %t3 dotop_terminal =
  %getelementptr inbounds %Token* %t3, i32 0, i32 1 loaded_dotop_terminal4 =
  %load i8** %dotop_terminal getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop_terminal4)
  store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %token
  %token5 = load %GenericPiece** %token loaded_deref6 = load %GenericPiece*
  %%token5 dotop_terminal7 = getelementptr inbounds %GenericPiece* %token5, i32
  %0, i32 5 loaded_dotop_terminal8 = load i32* %dotop_terminal7
  store i32 %loaded_dotop_terminal8, i32* %src_x
  %token9 = load %GenericPiece** %token loaded_deref10 = load %GenericPiece*
  %%token9 dotop_terminal11 = getelementptr inbounds %GenericPiece* %token9,
  %i32 0, i32 4 loaded_dotop_terminal12 = load i32* %dotop_terminal11
  store i32 %loaded_dotop_terminal12, i32* %src_y
  %src_x13 = load i32* %src_x
  store i32 %src_x13, i32* %dst_x
  %src_y14 = load i32* %src_y
  store i32 %src_y14, i32* %dst_y
  %token15 = load %GenericPiece** %token loaded_deref16 = load %GenericPiece*
  %%token15 dotop_terminal17 = getelementptr inbounds %GenericPiece* %token15,
  %i32 0, i32 5 loaded_dotop_terminal18 = load i32* %dotop_terminal17 tmp19 =
  %urem i32 %loaded_dotop_terminal18, 2 tmp20 = icmp eq i32 %tmp19, 0
  br i1 %tmp20, label %then22, label %else23

then:                                             ; preds = %entry
  %p1 = load %Player* @p1 loaded_dotop_terminal = load %Token* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 0)
  store %Token* getelementptr inbounds (%Player* @p1, i32 0, i32 0), %Token**
%t br label %merge

else:                                             ; preds = %entry
  %p2 = load %Player* @p2 loaded_dotop_terminal2 = load %Token* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 0)
  store %Token* getelementptr inbounds (%Player* @p2, i32 0, i32 0), %Token**
%t br label %merge

merge21:                                          ; preds = %else23, %then22
store i32 0, i32* %i br label %while

then22:                                           ; preds = %merge store i32 1,
```

```
i32* %direction br label %merge21

else23:                                           ; preds = %merge store i32
-1, i32* %direction br label %merge21

while:                                            ; preds = %merge35, %merge21
  %i46 = load i32* %i dice47 = load i32* %dice1 tmp48 = icmp slt i32 %i46,
  %%dice47
   br i1 %tmp48, label %while_body, label %merge49

while_body:                                       ; preds = %while
  %dst_y24 = load i32* %dst_y tmp25 = icmp eq i32 %dst_y24, 0 direction26 =
  %load i32* %direction tmp27 = icmp eq i32 %direction26, -1 tmp28 = and i1
  %%tmp25, %tmp27 dst_y29 = load i32* %dst_y tmp30 = icmp eq i32 %dst_y29, 5
  %direction31 = load i32* %direction tmp32 = icmp eq i32 %direction31, 1 tmp33
  %= and i1 %tmp30, %tmp32 tmp34 = or i1 %tmp28, %tmp33
   br i1 %tmp34, label %then36, label %else40

merge35:                                          ; preds = %else40, %then36
  %i44 = load i32* %i tmp45 = add i32 %i44, 1
   store i32 %tmp45, i32* %i br label %while

then36:                                           ; preds = %while_body
  %dst_x37 = load i32* %dst_x tmp38 = sub i32 %dst_x37, 1
   store i32 %tmp38, i32* %dst_x
  %direction39 = load i32* %direction Neg_op = sub i32 0, %direction39
   store i32 %Neg_op, i32* %direction br label %merge35

else40:                                           ; preds = %while_body
  %dst_y41 = load i32* %dst_y direction42 = load i32* %direction tmp43 = add
  %i32 %dst_y41, %direction42
   store i32 %tmp43, i32* %dst_y br label %merge35

merge49:                                          ; preds = %while
  %printf = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]*
  %@mmt53, i32 0, i32 0), i8* getelementptr inbounds ([10 x i8]* @name55, i32
  %0, i32 0)) dice50 = load i32* %dice1 dice51 = load i32* %dice1 printf52 =
  %call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt52,
  %i32 0, i32 0), i32 %dice51) dst_y53 = load i32* %dst_y dst_x54 = load i32*
  %%dst_x token55 = load %GenericPiece** %token moveOnGrid_result = call i32
  %@moveOnGrid(%GenericPiece* %token55, i32 %dst_x54, i32 %dst_y53)
   ret i32 0 }

define i32 @main() { entry:
  %ladder1 = alloca %Ladder ladder2 = alloca %Ladder ladder1_top = alloca
  %%LadderTop ladder2_top = alloca %LadderTop snake1 = alloca %Snake snake2 =
  %alloca %Snake snake1_tail = alloca %Snaketail snake2_tail = alloca
  %%Snaketail repeat = alloca i32 ladder11 = load %Ladder* %ladder1
  %displayString = getelementptr inbounds %Ladder* %ladder1, i32 0, i32 1
   store i8* getelementptr inbounds ([10 x i8]* @name58, i32 0, i32 0), i8**
%displayString
  %ladder22 = load %Ladder* %ladder2 displayString3 = getelementptr inbounds
  %%Ladder* %ladder2, i32 0, i32 1
   store i8* getelementptr inbounds ([10 x i8]* @name59, i32 0, i32 0), i8**
```

```
%displayString3
  %ladder1_top4 = load %LadderTop* %ladder1_top displayString5 = getelementptr
  %inbounds %LadderTop* %ladder1_top, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name60, i32 0, i32 0), i8**
%displayString5
  %ladder2_top6 = load %LadderTop* %ladder2_top displayString7 = getelementptr
  %inbounds %LadderTop* %ladder2_top, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name61, i32 0, i32 0), i8**
%displayString7
  %snake18 = load %Snake* %snake1 displayString9 = getelementptr inbounds
  %%Snake* %snake1, i32 0, i32 1
  store i8* getelementptr inbounds ([8 x i8]* @name62, i32 0, i32 0), i8**
%displayString9
  %snake1_tail10 = load %Snaketail* %snake1_tail displayString11 =
  %getelementptr inbounds %Snaketail* %snake1_tail, i32 0, i32 1
  store i8* getelementptr inbounds ([8 x i8]* @name63, i32 0, i32 0), i8**
%displayString11
  %snake212 = load %Snake* %snake2 displayString13 = getelementptr inbounds
  %%Snake* %snake2, i32 0, i32 1
  store i8* getelementptr inbounds ([8 x i8]* @name64, i32 0, i32 0), i8**
%displayString13
  %snake2_tail14 = load %Snaketail* %snake2_tail displayString15 =
  %getelementptr inbounds %Snaketail* %snake2_tail, i32 0, i32 1
  store i8* getelementptr inbounds ([8 x i8]* @name65, i32 0, i32 0), i8**
%displayString15
  %p1 = load %Player* @p1 loaded_dotop_terminal = load %Token* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 0) p116 = load %Player* @p1
  %loaded_dotop_terminal17 = load %Token* getelementptr inbounds (%Player* @p1,
  %i32 0, i32 0)
  store i8* getelementptr inbounds ([3 x i8]* @name66, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p1, i32 0, i32 0, i32 1)
  %p2 = load %Player* @p2 loaded_dotop_terminal18 = load %Token* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 0) p219 = load %Player* @p2
  %loaded_dotop_terminal20 = load %Token* getelementptr inbounds (%Player* @p2,
  %i32 0, i32 0)
  store i8* getelementptr inbounds ([3 x i8]* @name67, i32 0, i32 0), i8**
getelementptr inbounds (%Player* @p2, i32 0, i32 0, i32 1)
  %snake121 = load %Snake* %snake1 newNode = alloca %GenericPiece newNode22 =
  %load %GenericPiece* %newNode owner = getelementptr inbounds %GenericPiece*
  %%newNode, i32 0, i32 0
  store %Player* null, %Player** %owner
  %snake123 = load %Snake* %snake1 newNode24 = load %GenericPiece* %newNode
  %Snake_node = getelementptr inbounds %GenericPiece* %newNode, i32 0, i32 8
  store %Snake* %snake1, %Snake** %Snake_node
  %snake125 = load %Snake* %snake1 dotop_terminal = getelementptr inbounds
  %%Snake* %snake1, i32 0, i32 1 loaded_dotop_terminal26 = load i8**
  %%dotop_terminal newNode27 = load %GenericPiece* %newNode nametag =
  %getelementptr inbounds %GenericPiece* %newNode, i32 0, i32 2
  store i8* %loaded_dotop_terminal26, i8** %nametag
  %newNode28 = load %GenericPiece* %newNode typetag = getelementptr inbounds
  %%GenericPiece* %newNode, i32 0, i32 1
  store i8* getelementptr inbounds ([6 x i8]* @name68, i32 0, i32 0), i8**
%typetag
  %newNode29 = load %GenericPiece* %newNode
```

```
  call void @addToGrid(i32 4, i32 2, %GenericPiece* %newNode)
  %snake1_tail30 = load %Snaketail* %snake1_tail newNode31 = alloca
  %%GenericPiece newNode32 = load %GenericPiece* %newNode31 owner33 =
  %getelementptr inbounds %GenericPiece* %newNode31, i32 0, i32 0
  store %Player* null, %Player** %owner33
  %snake1_tail34 = load %Snaketail* %snake1_tail newNode35 = load
  %%GenericPiece* %newNode31 Snaketail_node = getelementptr inbounds
  %%GenericPiece* %newNode31, i32 0, i32 7
  store %Snaketail* %snake1_tail, %Snaketail** %Snaketail_node
  %snake1_tail36 = load %Snaketail* %snake1_tail dotop_terminal37 =
  %getelementptr inbounds %Snaketail* %snake1_tail, i32 0, i32 1
  %loaded_dotop_terminal38 = load i8** %dotop_terminal37 newNode39 = load
  %%GenericPiece* %newNode31 nametag40 = getelementptr inbounds %GenericPiece*
  %%newNode31, i32 0, i32 2
  store i8* %loaded_dotop_terminal38, i8** %nametag40
  %newNode41 = load %GenericPiece* %newNode31 typetag42 = getelementptr
  %inbounds %GenericPiece* %newNode31, i32 0, i32 1
  store i8* getelementptr inbounds ([10 x i8]* @name69, i32 0, i32 0), i8**
%typetag42
  %newNode43 = load %GenericPiece* %newNode31
  call void @addToGrid(i32 5, i32 4, %GenericPiece* %newNode31)
  %snake244 = load %Snake* %snake2 newNode45 = alloca %GenericPiece newNode46 =
  %load %GenericPiece* %newNode45 owner47 = getelementptr inbounds
  %%GenericPiece* %newNode45, i32 0, i32 0
  store %Player* null, %Player** %owner47
  %snake248 = load %Snake* %snake2 newNode49 = load %GenericPiece* %newNode45
  %Snake_node50 = getelementptr inbounds %GenericPiece* %newNode45, i32 0, i32
  %8
  store %Snake* %snake2, %Snake** %Snake_node50
  %snake251 = load %Snake* %snake2 dotop_terminal52 = getelementptr inbounds
  %%Snake* %snake2, i32 0, i32 1 loaded_dotop_terminal53 = load i8**
  %%dotop_terminal52 newNode54 = load %GenericPiece* %newNode45 nametag55 =
  %getelementptr inbounds %GenericPiece* %newNode45, i32 0, i32 2
  store i8* %loaded_dotop_terminal53, i8** %nametag55
  %newNode56 = load %GenericPiece* %newNode45 typetag57 = getelementptr
  %inbounds %GenericPiece* %newNode45, i32 0, i32 1
  store i8* getelementptr inbounds ([6 x i8]* @name70, i32 0, i32 0), i8**
%typetag57
  %newNode58 = load %GenericPiece* %newNode45
  call void @addToGrid(i32 0, i32 4, %GenericPiece* %newNode45)
  %snake2_tail59 = load %Snaketail* %snake2_tail newNode60 = alloca
  %%GenericPiece newNode61 = load %GenericPiece* %newNode60 owner62 =
  %getelementptr inbounds %GenericPiece* %newNode60, i32 0, i32 0
  store %Player* null, %Player** %owner62
  %snake2_tail63 = load %Snaketail* %snake2_tail newNode64 = load
  %%GenericPiece* %newNode60 Snaketail_node65 = getelementptr inbounds
  %%GenericPiece* %newNode60, i32 0, i32 7
  store %Snaketail* %snake2_tail, %Snaketail** %Snaketail_node65
  %snake2_tail66 = load %Snaketail* %snake2_tail dotop_terminal67 =
  %getelementptr inbounds %Snaketail* %snake2_tail, i32 0, i32 1
  %loaded_dotop_terminal68 = load i8** %dotop_terminal67 newNode69 = load
  %%GenericPiece* %newNode60 nametag70 = getelementptr inbounds %GenericPiece*
  %%newNode60, i32 0, i32 2
  store i8* %loaded_dotop_terminal68, i8** %nametag70
```

```
  %newNode71 = load %GenericPiece* %newNode60 typetag72 = getelementptr
  %inbounds %GenericPiece* %newNode60, i32 0, i32 1
   store i8* getelementptr inbounds ([10 x i8]* @name71, i32 0, i32 0), i8**
%typetag72
  %newNode73 = load %GenericPiece* %newNode60
   call void @addToGrid(i32 5, i32 1, %GenericPiece* %newNode60)
  %ladder174 = load %Ladder* %ladder1 newNode75 = alloca %GenericPiece
  %newNode76 = load %GenericPiece* %newNode75 owner77 = getelementptr inbounds
  %%GenericPiece* %newNode75, i32 0, i32 0
   store %Player* null, %Player** %owner77
  %ladder178 = load %Ladder* %ladder1 newNode79 = load %GenericPiece*
  %%newNode75 Ladder_node = getelementptr inbounds %GenericPiece* %newNode75,
  %i32 0, i32 10
   store %Ladder* %ladder1, %Ladder** %Ladder_node
  %ladder180 = load %Ladder* %ladder1 dotop_terminal81 = getelementptr inbounds
  %%Ladder* %ladder1, i32 0, i32 1 loaded_dotop_terminal82 = load i8**
  %%dotop_terminal81 newNode83 = load %GenericPiece* %newNode75 nametag84 =
  %getelementptr inbounds %GenericPiece* %newNode75, i32 0, i32 2
   store i8* %loaded_dotop_terminal82, i8** %nametag84
  %newNode85 = load %GenericPiece* %newNode75 typetag86 = getelementptr
  %inbounds %GenericPiece* %newNode75, i32 0, i32 1
   store i8* getelementptr inbounds ([7 x i8]* @name72, i32 0, i32 0), i8**
%typetag86
  %newNode87 = load %GenericPiece* %newNode75
   call void @addToGrid(i32 2, i32 3, %GenericPiece* %newNode75)
  %ladder1_top88 = load %LadderTop* %ladder1_top newNode89 = alloca
  %%GenericPiece newNode90 = load %GenericPiece* %newNode89 owner91 =
  %getelementptr inbounds %GenericPiece* %newNode89, i32 0, i32 0
   store %Player* null, %Player** %owner91
  %ladder1_top92 = load %LadderTop* %ladder1_top newNode93 = load
  %%GenericPiece* %newNode89 LadderTop_node = getelementptr inbounds
  %%GenericPiece* %newNode89, i32 0, i32 9
   store %LadderTop* %ladder1_top, %LadderTop** %LadderTop_node
  %ladder1_top94 = load %LadderTop* %ladder1_top dotop_terminal95 =
  %getelementptr inbounds %LadderTop* %ladder1_top, i32 0, i32 1
  %loaded_dotop_terminal96 = load i8** %dotop_terminal95 newNode97 = load
  %%GenericPiece* %newNode89 nametag98 = getelementptr inbounds %GenericPiece*
  %%newNode89, i32 0, i32 2
   store i8* %loaded_dotop_terminal96, i8** %nametag98
  %newNode99 = load %GenericPiece* %newNode89 typetag100 = getelementptr
  %inbounds %GenericPiece* %newNode89, i32 0, i32 1
   store i8* getelementptr inbounds ([10 x i8]* @name73, i32 0, i32 0), i8**
%typetag100
  %newNode101 = load %GenericPiece* %newNode89
   call void @addToGrid(i32 0, i32 1, %GenericPiece* %newNode89)
  %ladder2102 = load %Ladder* %ladder2 newNode103 = alloca %GenericPiece
  %newNode104 = load %GenericPiece* %newNode103 owner105 = getelementptr
  %inbounds %GenericPiece* %newNode103, i32 0, i32 0
   store %Player* null, %Player** %owner105
  %ladder2106 = load %Ladder* %ladder2 newNode107 = load %GenericPiece*
  %%newNode103 Ladder_node108 = getelementptr inbounds %GenericPiece*
  %%newNode103, i32 0, i32 10
   store %Ladder* %ladder2, %Ladder** %Ladder_node108
  %ladder2109 = load %Ladder* %ladder2 dotop_terminal110 = getelementptr
```

```
  %inbounds %Ladder* %ladder2, i32 0, i32 1 loaded_dotop_terminal111 = load
  %i8** %dotop_terminal110 newNode112 = load %GenericPiece* %newNode103
  %nametag113 = getelementptr inbounds %GenericPiece* %newNode103, i32 0, i32 2
  store i8* %loaded_dotop_terminal111, i8** %nametag113
  %newNode114 = load %GenericPiece* %newNode103 typetag115 = getelementptr
  %inbounds %GenericPiece* %newNode103, i32 0, i32 1
  store i8* getelementptr inbounds ([7 x i8]* @name74, i32 0, i32 0), i8**
%typetag115
  %newNode116 = load %GenericPiece* %newNode103
  call void @addToGrid(i32 4, i32 0, %GenericPiece* %newNode103)
  %ladder2_top117 = load %LadderTop* %ladder2_top newNode118 = alloca
  %%GenericPiece newNode119 = load %GenericPiece* %newNode118 owner120 =
  %getelementptr inbounds %GenericPiece* %newNode118, i32 0, i32 0
  store %Player* null, %Player** %owner120
  %ladder2_top121 = load %LadderTop* %ladder2_top newNode122 = load
  %%GenericPiece* %newNode118 LadderTop_node123 = getelementptr inbounds
  %%GenericPiece* %newNode118, i32 0, i32 9
  store %LadderTop* %ladder2_top, %LadderTop** %LadderTop_node123
  %ladder2_top124 = load %LadderTop* %ladder2_top dotop_terminal125 =
  %getelementptr inbounds %LadderTop* %ladder2_top, i32 0, i32 1
  %loaded_dotop_terminal126 = load i8** %dotop_terminal125 newNode127 = load
  %%GenericPiece* %newNode118 nametag128 = getelementptr inbounds
  %%GenericPiece* %newNode118, i32 0, i32 2
  store i8* %loaded_dotop_terminal126, i8** %nametag128
  %newNode129 = load %GenericPiece* %newNode118 typetag130 = getelementptr
  %inbounds %GenericPiece* %newNode118, i32 0, i32 1
  store i8* getelementptr inbounds ([10 x i8]* @name75, i32 0, i32 0), i8**
%typetag130
  %newNode131 = load %GenericPiece* %newNode118
  call void @addToGrid(i32 2, i32 0, %GenericPiece* %newNode118)
  %p1132 = load %Player* @p1 loaded_dotop_terminal133 = load %Token*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 0) newNode134 = alloca
  %%GenericPiece p1135 = load %Player* @p1 newNode136 = load %GenericPiece*
  %%newNode134 owner137 = getelementptr inbounds %GenericPiece* %newNode134,
  %i32 0, i32 0
  store %Player* @p1, %Player** %owner137
  %p1138 = load %Player* @p1 loaded_dotop_terminal139 = load %Token*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 0) newNode140 = load
  %%GenericPiece* %newNode134 Token_node = getelementptr inbounds
  %%GenericPiece* %newNode134, i32 0, i32 6
  store %Token* getelementptr inbounds (%Player* @p1, i32 0, i32 0), %Token**
%Token_node
  %p1141 = load %Player* @p1 loaded_dotop_terminal142 = load %Token*
  %getelementptr inbounds (%Player* @p1, i32 0, i32 0) p1143 = load %Player*
  %@p1 loaded_dotop_terminal144 = load %Token* getelementptr inbounds (%Player*
  %@p1, i32 0, i32 0) loaded_dotop = load i8** getelementptr inbounds (%Player*
  %@p1, i32 0, i32 0, i32 1) newNode145 = load %GenericPiece* %newNode134
  %nametag146 = getelementptr inbounds %GenericPiece* %newNode134, i32 0, i32 2
  store i8* %loaded_dotop, i8** %nametag146
  %newNode147 = load %GenericPiece* %newNode134 typetag148 = getelementptr
  %inbounds %GenericPiece* %newNode134, i32 0, i32 1
  store i8* getelementptr inbounds ([6 x i8]* @name76, i32 0, i32 0), i8**
%typetag148
  %newNode149 = load %GenericPiece* %newNode134
```

```
  call void @addToGrid(i32 5, i32 5, %GenericPiece* %newNode134)
  %p2150 = load %Player* @p2 loaded_dotop_terminal151 = load %Token*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 0) newNode152 = alloca
  %%GenericPiece p2153 = load %Player* @p2 newNode154 = load %GenericPiece*
  %%newNode152 owner155 = getelementptr inbounds %GenericPiece* %newNode152,
  %i32 0, i32 0
  store %Player* @p2, %Player** %owner155
  %p2156 = load %Player* @p2 loaded_dotop_terminal157 = load %Token*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 0) newNode158 = load
  %%GenericPiece* %newNode152 Token_node159 = getelementptr inbounds
  %%GenericPiece* %newNode152, i32 0, i32 6
  store %Token* getelementptr inbounds (%Player* @p2, i32 0, i32 0), %Token**
%Token_node159
  %p2160 = load %Player* @p2 loaded_dotop_terminal161 = load %Token*
  %getelementptr inbounds (%Player* @p2, i32 0, i32 0) p2162 = load %Player*
  %@p2 loaded_dotop_terminal163 = load %Token* getelementptr inbounds (%Player*
  %@p2, i32 0, i32 0) loaded_dotop164 = load i8** getelementptr inbounds
  %(%Player* @p2, i32 0, i32 0, i32 1) newNode165 = load %GenericPiece*
  %%newNode152 nametag166 = getelementptr inbounds %GenericPiece* %newNode152,
  %i32 0, i32 2
  store i8* %loaded_dotop164, i8** %nametag166
  %newNode167 = load %GenericPiece* %newNode152 typetag168 = getelementptr
  %inbounds %GenericPiece* %newNode152, i32 0, i32 1
  store i8* getelementptr inbounds ([6 x i8]* @name77, i32 0, i32 0), i8**
%typetag168
  %newNode169 = load %GenericPiece* %newNode152
  call void @addToGrid(i32 5, i32 5, %GenericPiece* %newNode152) store i32 2,
i32* @playerOrderSize
  %gameloop_result = call i32 @gameloop()
  ret i32 0 }

define i32 @colocation(i32 %x, i32 %y, %GenericPiece* %i1, %GenericPiece* %i2)
{ entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %y2 = alloca i32
  store i32 %y, i32* %y2
  %i13 = alloca %GenericPiece*
  store %GenericPiece* %i1, %GenericPiece** %i13
  %i24 = alloca %GenericPiece*
  store %GenericPiece* %i2, %GenericPiece** %i24
  %l = alloca %GenericPiece* dst_x = alloca i32 dst_y = alloca i32 repeat =
  %alloca i32 i15 = load %GenericPiece** %i13 loaded_deref = load
  %%GenericPiece* %i15 dotop_terminal = getelementptr inbounds %GenericPiece*
  %%i15, i32 0, i32 1 loaded_dotop_terminal = load i8** %dotop_terminal tmp =
  %icmp eq i8* %loaded_dotop_terminal, getelementptr inbounds ([6 x i8]*
  %@name80, i32 0, i32 0) i26 = load %GenericPiece** %i24 loaded_deref7 = load
  %%GenericPiece* %i26 dotop_terminal8 = getelementptr inbounds %GenericPiece*
  %%i26, i32 0, i32 1 loaded_dotop_terminal9 = load i8** %dotop_terminal8 tmp10
  %= icmp eq i8* %loaded_dotop_terminal9, getelementptr inbounds ([6 x i8]*
  %@name81, i32 0, i32 0) tmp11 = and i1 %tmp, %tmp10
  br i1 %tmp11, label %then, label %else39

merge:                                            ; preds = %else39, %merge24
```

```
  %i140 = load %GenericPiece** %i13 loaded_deref41 = load %GenericPiece* %i140
  %dotop_terminal42 = getelementptr inbounds %GenericPiece* %i140, i32 0, i32 1
  %loaded_dotop_terminal43 = load i8** %dotop_terminal42 tmp44 = icmp eq i8*
  %%loaded_dotop_terminal43, getelementptr inbounds ([6 x i8]* @name88, i32 0,
  %i32 0) i245 = load %GenericPiece** %i24 loaded_deref46 = load %GenericPiece*
  %%i245 dotop_terminal47 = getelementptr inbounds %GenericPiece* %i245, i32 0,
  %i32 1 loaded_dotop_terminal48 = load i8** %dotop_terminal47 tmp49 = icmp eq
  %i8* %loaded_dotop_terminal48, getelementptr inbounds ([7 x i8]* @name89, i32
  %0, i32 0) tmp50 = and i1 %tmp44, %tmp49
  br i1 %tmp50, label %then52, label %else85

then:                                             ; preds = %entry
  %printGrid_result = call i32 @printGrid() printf = call i32 (i8*, ...)*
  %@printf(i8* getelementptr inbounds ([4 x i8]* @mmt79, i32 0, i32 0), i8*
  %getelementptr inbounds ([9 x i8]* @name83, i32 0, i32 0)) i212 = load
  %%GenericPiece** %i24 loaded_deref13 = load %GenericPiece* %i212
  %dotop_terminal14 = getelementptr inbounds %GenericPiece* %i212, i32 0, i32 2
  %loaded_dotop_terminal15 = load i8** %dotop_terminal14 tmp16 = icmp eq i8*
  %%loaded_dotop_terminal15, getelementptr inbounds ([8 x i8]* @name84, i32 0,
  %i32 0)
  br i1 %tmp16, label %then18, label %else

merge17:                                          ; preds = %else, %then18
  %i219 = load %GenericPiece** %i24 loaded_deref20 = load %GenericPiece* %i219
  %dotop_terminal21 = getelementptr inbounds %GenericPiece* %i219, i32 0, i32 2
  %loaded_dotop_terminal22 = load i8** %dotop_terminal21 tmp23 = icmp eq i8*
  %%loaded_dotop_terminal22, getelementptr inbounds ([8 x i8]* @name86, i32 0,
  %i32 0)
  br i1 %tmp23, label %then25, label %else27

then18:                                           ; preds = %then
  %getPieceFromGrid_result = call %GenericPiece* @getPieceFromGrid(i8*
  %getelementptr inbounds ([8 x i8]* @name85, i32 0, i32 0))
  store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %l br label
%merge17

else:                                             ; preds = %then br label
%merge17

merge24:                                          ; preds = %else27, %then25
  %l28 = load %GenericPiece** %l loaded_deref29 = load %GenericPiece* %l28
  %dotop_terminal30 = getelementptr inbounds %GenericPiece* %l28, i32 0, i32 5
  %loaded_dotop_terminal31 = load i32* %dotop_terminal30
  store i32 %loaded_dotop_terminal31, i32* %dst_x
  %l32 = load %GenericPiece** %l loaded_deref33 = load %GenericPiece* %l32
  %dotop_terminal34 = getelementptr inbounds %GenericPiece* %l32, i32 0, i32 4
  %loaded_dotop_terminal35 = load i32* %dotop_terminal34
  store i32 %loaded_dotop_terminal35, i32* %dst_y
  %dst_y36 = load i32* %dst_y dst_x37 = load i32* %dst_x i138 = load
  %%GenericPiece** %i13 moveOnGrid_result = call i32 @moveOnGrid(%GenericPiece*
  %%i138, i32 %dst_x37, i32 %dst_y36)
  br label %merge

then25:                                           ; preds = %merge17
```

```
  %getPieceFromGrid_result26 = call %GenericPiece* @getPieceFromGrid(i8*
  %getelementptr inbounds ([8 x i8]* @name87, i32 0, i32 0))
   store %GenericPiece* %getPieceFromGrid_result26, %GenericPiece** %l br label
%merge24

else27:                                           ; preds = %merge17 br label
%merge24

else39:                                           ; preds = %entry br label
%merge

merge51:                                          ; preds = %else85, %merge69
ret i32 0

then52:                                           ; preds = %merge
  %printGrid_result53 = call i32 @printGrid() printf54 = call i32 (i8*, ...)*
  %@printf(i8* getelementptr inbounds ([4 x i8]* @mmt79, i32 0, i32 0), i8*
  %getelementptr inbounds ([14 x i8]* @name91, i32 0, i32 0)) i255 = load
  %%GenericPiece** %i24 loaded_deref56 = load %GenericPiece* %i255
  %dotop_terminal57 = getelementptr inbounds %GenericPiece* %i255, i32 0, i32 2
  %loaded_dotop_terminal58 = load i8** %dotop_terminal57 tmp59 = icmp eq i8*
  %%loaded_dotop_terminal58, getelementptr inbounds ([10 x i8]* @name92, i32 0,
  %i32 0)
   br i1 %tmp59, label %then61, label %else63

merge60:                                          ; preds = %else63, %then61
  %i264 = load %GenericPiece** %i24 loaded_deref65 = load %GenericPiece* %i264
  %dotop_terminal66 = getelementptr inbounds %GenericPiece* %i264, i32 0, i32 2
  %loaded_dotop_terminal67 = load i8** %dotop_terminal66 tmp68 = icmp eq i8*
  %%loaded_dotop_terminal67, getelementptr inbounds ([10 x i8]* @name94, i32 0,
  %i32 0)
   br i1 %tmp68, label %then70, label %else72

then61:                                           ; preds = %then52
  %getPieceFromGrid_result62 = call %GenericPiece* @getPieceFromGrid(i8*
  %getelementptr inbounds ([7 x i8]* @name93, i32 0, i32 0))
   store %GenericPiece* %getPieceFromGrid_result62, %GenericPiece** %l br label
%merge60

else63:                                           ; preds = %then52 br label
%merge60

merge69:                                          ; preds = %else72, %then70
  %l73 = load %GenericPiece** %l loaded_deref74 = load %GenericPiece* %l73
  %dotop_terminal75 = getelementptr inbounds %GenericPiece* %l73, i32 0, i32 5
  %loaded_dotop_terminal76 = load i32* %dotop_terminal75
   store i32 %loaded_dotop_terminal76, i32* %dst_x
  %l77 = load %GenericPiece** %l loaded_deref78 = load %GenericPiece* %l77
  %dotop_terminal79 = getelementptr inbounds %GenericPiece* %l77, i32 0, i32 4
  %loaded_dotop_terminal80 = load i32* %dotop_terminal79
   store i32 %loaded_dotop_terminal80, i32* %dst_y
  %dst_y81 = load i32* %dst_y dst_x82 = load i32* %dst_x i183 = load
  %%GenericPiece** %i13 moveOnGrid_result84 = call i32
  %@moveOnGrid(%GenericPiece* %i183, i32 %dst_x82, i32 %dst_y81)
```

```
  br label %merge51

then70:                                           ; preds = %merge60
  %getPieceFromGrid_result71 = call %GenericPiece* @getPieceFromGrid(i8*
  %getelementptr inbounds ([7 x i8]* @name95, i32 0, i32 0))
  store %GenericPiece* %getPieceFromGrid_result71, %GenericPiece** %l br label
%merge69

else72:                                           ; preds = %merge60 br label
%merge69

else85:                                           ; preds = %merge br label
%merge51 }

define i32 @checkGameEnd() { entry:
  %t = alloca %Token* token = alloca %GenericPiece* repeat = alloca i32
  %currentPlayerIndex = load i32* @currentPlayerIndex tmp = icmp eq i32
  %%currentPlayerIndex, 0
  br i1 %tmp, label %then, label %else

merge:                                            ; preds = %else, %then
  %t2 = load %Token** %t loaded_deref = load %Token* %t2 dotop_terminal =
  %getelementptr inbounds %Token* %t2, i32 0, i32 1 loaded_dotop_terminal3 =
  %load i8** %dotop_terminal getPieceFromGrid_result = call %GenericPiece*
  %@getPieceFromGrid(i8* %loaded_dotop_terminal3)
  store %GenericPiece* %getPieceFromGrid_result, %GenericPiece** %token
  %token4 = load %GenericPiece** %token loaded_deref5 = load %GenericPiece*
  %%token4 dotop_terminal6 = getelementptr inbounds %GenericPiece* %token4, i32
  %0, i32 5 loaded_dotop_terminal7 = load i32* %dotop_terminal6 tmp8 = icmp eq
  %i32 %loaded_dotop_terminal7, 0 token9 = load %GenericPiece** %token
  %loaded_deref10 = load %GenericPiece* %token9 dotop_terminal11 =
  %getelementptr inbounds %GenericPiece* %token9, i32 0, i32 4
  %loaded_dotop_terminal12 = load i32* %dotop_terminal11 tmp13 = icmp eq i32
  %%loaded_dotop_terminal12, 5 tmp14 = and i1 %tmp8, %tmp13
  br i1 %tmp14, label %then16, label %else26

then:                                             ; preds = %entry
  %p1 = load %Player* @p1 loaded_dotop_terminal = load %Token* getelementptr
  %inbounds (%Player* @p1, i32 0, i32 0)
  store %Token* getelementptr inbounds (%Player* @p1, i32 0, i32 0), %Token**
%t br label %merge

else:                                             ; preds = %entry
  %p2 = load %Player* @p2 loaded_dotop_terminal1 = load %Token* getelementptr
  %inbounds (%Player* @p2, i32 0, i32 0)
  store %Token* getelementptr inbounds (%Player* @p2, i32 0, i32 0), %Token**
%t br label %merge

merge15:                                          ; preds = %else26 ret i32 0

then16:                                           ; preds = %merge
  %printGrid_result = call i32 @printGrid() printf = call i32 (i8*, ...)*
  %@printf(i8* getelementptr inbounds ([4 x i8]* @mmt97, i32 0, i32 0), i8*
  %getelementptr inbounds ([12 x i8]* @name99, i32 0, i32 0)) t17 = load
```

```
  %%Token** %t loaded_deref18 = load %Token* %t17 dotop_terminal19 =
  %getelementptr inbounds %Token* %t17, i32 0, i32 1 loaded_dotop_terminal20 =
  %load i8** %dotop_terminal19 t21 = load %Token** %t loaded_deref22 = load
  %%Token* %t21 dotop_terminal23 = getelementptr inbounds %Token* %t21, i32 0,
  %i32 1 loaded_dotop_terminal24 = load i8** %dotop_terminal23 printf25 = call
  %i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @mmt97, i32 0,
  %i32 0), i8* %loaded_dotop_terminal24)
  ret i32 1

else26:                                           ; preds = %merge br label
%merge15 }
```

6.2 Test Suites

Output of our test suite is below. OK indicates that the test case shows expected behavior.

```
$ ./testall.sh
test-access-1d-array .........OK
test-access-2d-array .........OK
test-add-ints .........OK
test-arith .........OK
test-array2dinit .........OK
test-file-import .........OK
test-init-1d-array .........OK
test-move-pawn .........OK
test-nestedstruct .........OK
test-print-1d-array .........OK
test-print-boolean .........OK
test-print-int-var .........OK
test-print-int .........OK
test-print-string-literal .........OK
test-print-string-var .........OK
test-struct-1Darray .........OK
test-struct-2Darray .........OK
test-struct-pointer .........OK
test-struct .........OK
test-struct1D-pointer .........OK
test-while .........OK
fail-assign1 .........OK
fail-assign2 .........OK
fail-assign3 .........OK
fail-dead1 .........OK
fail-dead2 .........OK
fail-expr1 .........OK
fail-expr2 .........OK
fail-for1 .........OK
fail-for2 .........OK
fail-for3 .........OK
fail-for4 .........OK
fail-for5 .........OK
fail-func1 .........OK
fail-func2 .........OK
fail-func3 .........OK
fail-func4 .........OK
```

```
fail-func5 .........OK
fail-func6 .........OK
fail-func7 .........OK
fail-func8 .........OK
fail-func9 .........OK
fail-global1 .........OK
fail-player-displaystring .........OK
fail-semant-check-print .........OK
fail-semant-illegal-struct-assignment .........OK
fail-semant-illegal-struct-element-access .........OK
fail-semant-undeclared-struct .........OK
```

6.3 Selection of Test Cases

The test cases were chosen so that every primitive datatype and built-in function was covered, along with a few tests (such as test-move-pawn) that integration-tested multiple features together. The error test cases were chosen so that a wide range of errors such as illegal assignment and out-of-scope variables were covered.

6.4 Test Automation

We used a script testall.sh that took as an input all the files(with ".grid" extension) in the /tests folder and compiled them to ".ll" format. The test programs that were supposed to pass started with "test_" and their expected output was matched to content of "test_" file with the same name as the source program and ending with ".golden" extension. Similarly, for test programs that were supposed to fail started with "fail_" and their expected output was matched to content of "fail_" file with the same name as the source program and ending with ".err" extension.

6.5 Testing Roles

Akshay wrote the test cases for arrays and failing test cases for all components. Dhruv wrote the passing test cases for structs. Parth wrote the passing test cases for semantic checking.

7 Lessons Learned

7.1 Dhruv

- Team matters. Choose members who are hungry to learn something new each day. I was very lucky to find such a team.
- When times get tough, move inch by inch, into the light.
- Two heads are always better than one when debugging code
- If an idea fails, keep your heads high, make some change, try until it fails again. Repeat.

7.2 Sagar

- While it is important to be flexible over the course of taking on a project of this scope, it is essential to have a relatively clear plan of all your core language features from the beginning. Know the What and the Why. The How, you will figure out along the way. If you have some particularly challenging features, try taking those on first. If you fail quickly, you can modify your plan early enough so that it doesn't derail you.

7.3 Parth

- I realized the importance of pair programming through this project. When you are stuck debugging for hours, a fresh pair of eyes helps.
- Understanding team members who you tune in with is very important. That enables correct and streamlined implementation of the project with any design choice conflicts being resolved via healthy discussions.

7.4 Akshay

- Too much division of work can lead to delay as integrating individual components also takes a lot more time than you may think.
- Starting early helped us a lot.
- Pair programming was the most powerful tool against fixing silly bugs in code.
- Try to make a small subset of your ambitious language plans before committing completely to a domain.

8 Appendix

Note: Since all of the members had a contribution to all the files, we have not individually attributed any of the files. **grid.ml**

```
type action =  LLVM_IR | Compile

let _ =
  let action = if Array.length Sys.argv > 1 then
    List.assoc Sys.argv.(1) [
            ("-l", LLVM_IR);  (* Generate LLVM, don't check *)
            ("-c", Compile) ] (* Generate, check LLVM IR *)
  else Compile in
  let lexbuf = Lexing.from_string (Preprocess.process_files Sys.argv.(2) ) in
  let ast = Parser.program Scanner.token lexbuf in
  Semant.check ast;
    match action with
  LLVM_IR -> print_string (Llvm.string_of_llmodule (Codegen.translate ast))
  | Compile -> let m = Codegen.translate ast in
    Llvm_analysis.assert_valid_module m;
print_string (Llvm.string_of_llmodule m)
```

ast.ml

```
type typ =
        Int
        | Bool
        | Void
        | String
        | Array1DType of typ * int  (* int[m] *)
        | PlayerType
        | Array2DType of typ * int * int  (* int[m][n] *)
        | StructType of string
        | PointerType of typ
        | GridType of int * int

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

type uop = Neg | Not | Deref | Ref

type bind = typ * string

type expr =
```

```
    Literal of int
  | Null of string
  | BoolLit of bool
  | Array1DAccess of string * expr
  | Array2DAccess of string * expr * expr
  | Call of string * expr list
  | Id of string
  | Binop of expr * op * expr
  | Dotop of expr * string
  | Unop of uop * expr
  | GridAssign of expr * expr * expr
  | DeletePiece of expr * expr * expr
  | Assign of expr * expr
  | Array1DAssign of string * expr * expr
  | Array2DAssign of string * expr * expr * expr
  | String_Lit of string
  | ArrayLiteral of expr list
  | Noexpr


type stmt =
    Block of stmt list
  | Expr of expr
  | For of expr * expr * expr * stmt
  | If of expr * stmt * stmt
  | While of expr * stmt
  | Return of expr

type func_decl = {
    typ : typ;
    fname : string;
    formals : bind list;
    locals : bind list;
    body : stmt list;
  }

type struct_decl = {
    sname: string;
    sformals: bind list;
    sfunc: func_decl;
}

type program = bind list * func_decl list * struct_decl list

(* Pretty-printing functions *)

let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
```

```
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "&&"
  | Or -> "||"
  | Modulo -> "%"

let string_of_uop = function
    Neg -> "-"
  | Not -> "!"
  | Deref -> "Deref"
  | Ref -> "&"

let rec string_of_expr = function
    Literal(l) -> string_of_int l
  | BoolLit(true) -> "true"
  | BoolLit(false) -> "false"
  | Null s -> "Null " ^ s
  | GridAssign(e1, e2, e3) ->
    "GridAssign" ^string_of_expr e1 ^ string_of_expr e2 ^ string_of_expr e3
  | DeletePiece(e1, e2, e3) ->
    "DeletePiece" ^string_of_expr e1 ^ string_of_expr e2 ^ string_of_expr e3
  | Array1DAccess(s, e) -> s ^ "[" ^ string_of_expr e ^ "]"
  | Array2DAccess(s, e1, e2) ->
    s ^ "[" ^ string_of_expr e1 ^ "]" ^ "[" ^ string_of_expr e2 ^ "]"
  | ArrayLiteral(e) ->
    "ArrayLiteral[" ^ String.concat "," (List.map string_of_expr e) ^ "]"
  | Id(s) -> s
  | String_Lit(s) -> s
  | Dotop(e, s) -> string_of_expr e ^ "." ^ s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e
  | Array1DAssign(s, e1, e2) ->
    s ^ "[" ^string_of_expr e1 ^"] =" ^ string_of_expr e2
  | Array2DAssign(s, e1, e2, e3) ->
      s ^ "[" ^ string_of_expr e1 ^ "," ^string_of_expr e2 ^ "] =" ^ string_of_expr e3
  | Call(f, el) ->
    f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""


let rec string_of_stmt = function
    Block(stmts) ->
      "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
      string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s
```

```
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s

let rec string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Void -> "void"
  | String -> "string"
  | Array1DType(typ, e) ->
    string_of_typ typ ^ " array [" ^ string_of_int e ^ "]"
  | Array2DType(typ, e1, e2) ->
    string_of_typ typ ^ " array [" ^ string_of_int e1 ^ "][" ^ string_of_int e2 ^ "]"
  | StructType(s) -> "struct " ^ s
  | PointerType(typ) -> string_of_typ typ ^ " pointer"
  | PlayerType -> "Player"
  | GridType(i,j) -> "Grid " ^ string_of_int i ^ " , "^ string_of_int j

let string_of_vdecl (t, id) = string_of_typ t ^ " " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  string_of_typ fdecl.typ ^ " " ^
  fdecl.fname ^ "(" ^ String.concat ", " (List.map snd fdecl.formals) ^
  ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

scanner.mll

```
(* Ocamllex scanner for Grid compiler *)

{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['      { LARRAY }
| ']'      { RARRAY }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { TIMES }
| '/'      { DIVIDE }
| '='      { ASSIGN }
| '&'      { REF }
| '.'      { DOT }
| '<'      { LT }
```

```
| "<--"     { INARROW }
| "-->"     { OUTARROW }
| "=="      { EQ }
| "!="      { NEQ }
| "<="      { LEQ }
| ">"       { GT }
| ">="      { GEQ }
| "&&"      { AND }
| "||"      { OR }
| "!"       { NOT }
| "%"       { MODULO }
| "if"      { IF }
| "else"    { ELSE }
| "for"     { FOR }
| "while"   { WHILE }
| "return"  { RETURN }
| "int"     { INT }
| "string"  { STRING }
| "bool"    { BOOL }
| "None"    { NULL }
| "void"    { VOID }
| "true"    { TRUE }
| "false"   { FALSE }
| "Grid_Init"    { GRIDINIT }
| "Grid"    { GRID }
| "Player"  { PLAYER }
| "Piece"    { PIECE }
| ['0'-'9']+ as lxm { LITERAL(int_of_string lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| '"'([^'"']* as lxm)'"' { STRING_LIT(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

and comment = parse
  "*/" { token lexbuf }
| _    { comment lexbuf }
```

parser.mly

```
/* Ocamlyacc parser for Grid compiler */

%{
open Ast

let first (a,_,_) = a;;
let second (_,b,_) = b;;
let third (_,_,c) = c;;
%}


%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA LARRAY RARRAY
%token PLUS MINUS TIMES DIVIDE INARROW OUTARROW ASSIGN NOT DOT PERCENT DEREF REF MODULO
%token EQ NEQ LT LEQ GT GEQ TRUE FALSE AND OR GRIDINIT GRID NULL
%token RETURN IF ELSE FOR WHILE INT BOOL VOID STRING PLAYER PIECE
%token <int> LITERAL
```

```
%token <string> ID
%token <string> STRING_LIT
%token EOF

%nonassoc NOASSIGN
%nonassoc NOELSE
%nonassoc ELSE
%nonassoc NOLARRAY
%nonassoc POINTER
%right ASSIGN
%right INARROW
%right OUTARROW
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS MODULO
%left TIMES DIVIDE
%right NOT NEG DEREF REF
%left DOT

%start program
%type <Ast.program> program

%%

program:
  decls EOF { $1 }

decls:
   /* nothing */ { [], [], [] }
 | decls vdecl { (List.append $2 (first $1)), second $1, third $1 }
 | decls fdecl { first $1, ($2 :: second $1), third $1 }
 | decls sdecl { first $1, second $1, ($2 :: third $1) }

fdecl:
   typ ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
     { { typ = $1;
   fname = $2;
   formals = $4;
   locals = List.rev $7;
   body = List.rev $8 } }

formals_opt:
    /* nothing */ { [] }
  | formal_list   { List.rev $1 }

formal_list:
    typ ID                   { [($1,$2)] }
  | formal_list COMMA typ ID { ($3,$4) :: $1 }

typ:
    INT { Int }
  | BOOL { Bool }
```

```
  | VOID { Void }
  | STRING { String }
  | array1d_type { $1 }   /* int[4] */
  | array2d_type { $1 }   /* int[4][3] */
  | PIECE ID { StructType ($2) }
  | PLAYER { PlayerType }
  | typ TIMES %prec POINTER { PointerType ($1) }
  | GRIDINIT LT LITERAL COMMA LITERAL GT { GridType ($3, $5) }

array1d_type:
    typ LARRAY LITERAL RARRAY %prec NOLARRAY { Array1DType($1,$3) }  /* int[4] */

array2d_type:
    typ LARRAY LITERAL COMMA LITERAL RARRAY { Array2DType($1,$3,$5) } /* int[4][3] */

arr_literal:
  expr   {[$1]}
  | arr_literal COMMA expr {$3::$1}

vdecl_list:
    /* nothing */    { [] }
  | vdecl_list vdecl { List.append $2 $1 }

multi_vdecl:
     ID {[$1]}
   | multi_vdecl COMMA ID {$3::$1}

vdecl:
   typ multi_vdecl SEMI { List.map (fun x -> ($1,x)) $2 }

sdecl:
    PIECE ID LBRACE vdecl_list fdecl RBRACE
      { { sname = $2;
      sformals = $4;
      sfunc = $5;
      } }
    | PLAYER LBRACE vdecl_list fdecl RBRACE
      { { sname = "Player";
      sformals = $3;
      sfunc = $4;
      } }

stmt_list:
    /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
    expr SEMI { Expr $1 }
  | RETURN SEMI { Return Noexpr }
  | RETURN expr SEMI { Return $2 }
  | LBRACE stmt_list RBRACE { Block(List.rev $2) }
  | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
  | IF LPAREN expr RPAREN stmt ELSE stmt    { If($3, $5, $7) }
  | FOR LPAREN expr_opt SEMI expr SEMI expr_opt RPAREN stmt
```

```
    { For($3, $5, $7, $9) }
  | WHILE LPAREN expr RPAREN stmt { While($3, $5) }

expr_opt:
    /* nothing */ { Noexpr }
  | expr          { $1 }

expr:
    LITERAL          { Literal($1) }
  | NULL             { Null("GenericPiece") }
  | TRUE             { BoolLit(true) }
  | FALSE            { BoolLit(false) }
  | GRID LT expr COMMA expr GT INARROW expr { GridAssign($3,$5,$8) }
  | GRID LT expr COMMA expr GT OUTARROW expr { DeletePiece($3,$5,$8) }
  | expr ASSIGN expr { Assign($1,$3) }
  | ID               { Id($1) }
  | STRING_LIT        { String_Lit($1) }
  | ID LARRAY expr COMMA expr RARRAY ASSIGN expr { Array2DAssign($1,$3,$5,$8)}
  | ID LARRAY expr RARRAY ASSIGN expr { Array1DAssign($1, $3, $6) }
  | ID LARRAY expr COMMA expr RARRAY %prec NOASSIGN { Array2DAccess ($1,$3,$5) }
  | ID LARRAY expr RARRAY %prec NOLARRAY{Array1DAccess($1,$3)}
  | expr PLUS   expr { Binop($1, Add,   $3) }
  | expr MINUS  expr { Binop($1, Sub,   $3) }
  | expr TIMES  expr { Binop($1, Mult,  $3) }
  | expr DIVIDE expr { Binop($1, Div,   $3) }
  | expr EQ     expr { Binop($1, Equal, $3) }
  | expr NEQ    expr { Binop($1, Neq,   $3) }
  | expr LT     expr { Binop($1, Less,  $3) }
  | expr LEQ    expr { Binop($1, Leq,   $3) }
  | expr GT     expr { Binop($1, Greater, $3) }
  | expr GEQ    expr { Binop($1, Geq,   $3) }
  | expr AND    expr { Binop($1, And,   $3) }
  | expr OR     expr { Binop($1, Or,    $3) }
  | expr MODULO expr { Binop($1, Modulo, $3) }
  | expr DOT    ID   { Dotop($1, $3) }
  | MINUS expr %prec NEG { Unop(Neg, $2) }
  | TIMES expr %prec DEREF { Unop(Deref, $2) }
  | REF expr { Unop(Ref, $2) }
  | NOT expr         { Unop(Not, $2) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }
  | LARRAY arr_literal RARRAY {ArrayLiteral(List.rev $2)}

actuals_opt:
    /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
    expr                    { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }
```

codegen.ml

```
module L = Llvm
module A = Ast
```

```
module StringMap = Map.Make(String)
module S = String

let internal_if_flag = ref 0

let translate (globals, functions, structs) =
  let context = L.global_context () in
  let the_module = L.create_module context "GridLang"
  and i32_t  = L.i32_type  context
  and i8_t   = L.i8_type   context
  and i1_t   = L.i1_type   context
  and void_t = L.void_type context
  and array_t   = L.array_type in
  let str_t = L.pointer_type i8_t in

  let new_global_builder = ref (L.builder context) in
  (*add all struct names to a hashtable*)
  let struct_types:(string, L.lltype) Hashtbl.t = Hashtbl.create 1000 in
  let struct_names:(L.lltype,string) Hashtbl.t = Hashtbl.create 1000 in

  let add_empty_named_struct_types sdecl =
    let struct_t = L.named_struct_type context sdecl.A.sname in
    let _ = Hashtbl.add struct_types sdecl.A.sname struct_t in
    Hashtbl.add struct_names struct_t sdecl.A.sname
  in

  let _ =  List.map add_empty_named_struct_types structs in

  let rec ltype_of_typ = function
     A.Int -> i32_t
    | A.Bool -> i1_t
    | A.Void -> void_t
    | A.StructType s ->  Hashtbl.find struct_types s
    | A.PlayerType -> ltype_of_typ((A.StructType "Player"))
    | A.String -> str_t
    | A.Array1DType (typ,size) -> array_t (ltype_of_typ typ) size
    | A.PointerType t -> L.pointer_type (ltype_of_typ t)
    | A.Array2DType (typ,size1,size2) ->
      array_t (array_t (ltype_of_typ typ) size2) size1
    | A.GridType (rows, cols) ->
      ltype_of_typ (A.Array2DType ((A.StructType("GenericPiece")), rows, cols))
  in


  let vars_global:(string, L.llvalue) Hashtbl.t = Hashtbl.create 1000 in

  let rec create_rep_list this_list llvm_val count =
    (match count with
      0 -> this_list
      |_ -> create_rep_list (llvm_val::this_list) llvm_val (count-1))
    in
  let createThisGrid rows cols =
    let str_typ = ltype_of_typ (A.PointerType(A.StructType("GenericPiece"))) in
```

```
  let cell_init = L.const_pointer_null str_typ in
  let each_col_init =
    L.const_array str_typ (Array.of_list (create_rep_list [] cell_init cols))
  in
  let ty_each_col = array_t str_typ cols in
  let init = L.const_array ty_each_col
  (Array.of_list (create_rep_list [] each_col_init rows)) in
  let grid_val = L.define_global "GridData" init the_module in
  let _ = Hashtbl.add vars_global "GridData" grid_val in
  grid_val
in

let other_global_vars (t,n) =
  let global_val =
  (match t with
   A.Int -> L.define_global n
          (L.const_int (ltype_of_typ (A.Int)) 0) the_module
  | A.Bool -> L.define_global n
            (L.const_int (ltype_of_typ (A.Bool)) 0) the_module
  | A.String -> L.define_global
                n (L.const_string context "") the_module
  | A.PlayerType -> let init = L.const_null (ltype_of_typ (A.PlayerType)) in
                    L.define_global n init the_module
  | A.PointerType t1 -> let init = L.const_pointer_null
              (ltype_of_typ (A.PointerType(t1))) in
              L.define_global n init the_module
  | A.StructType (s)-> let init = L.const_null (ltype_of_typ (A.StructType(s))) in
                    L.define_global n init the_module
  | A.Array1DType (typ,size) ->
    let each_cell = L.const_null (ltype_of_typ typ) in
    let init = L.const_array (ltype_of_typ typ)
    (Array.of_list (create_rep_list [] each_cell size)) in
    L.define_global n init the_module
  | _ -> raise(Failure("Invalid type of global declaration")))
  in Hashtbl.add vars_global n global_val;global_val
in

let global_var_func (t, n) =
    match t with
      A.GridType (rows, cols) ->
        (*Allocate int rows and int cols in global context*)
        let row_val = L.define_global "rows"
        (L.const_int (ltype_of_typ (A.Int)) rows) the_module in
        let _ = Hashtbl.add vars_global "rows" row_val in
        let col_val = L.define_global "cols"
        (L.const_int (ltype_of_typ (A.Int)) cols) the_module in
        let _ = Hashtbl.add vars_global "cols" col_val in
        createThisGrid rows cols
      | _ -> other_global_vars (t,n)
in
ignore(List.map global_var_func globals);

let global_val = L.define_global "currentPlayerIndex"
(L.const_int (ltype_of_typ (A.Int)) 0) the_module in
```

```
Hashtbl.add vars_global "currentPlayerIndex" global_val;

let populate_struct_type sdecl =
  let struct_t = Hashtbl.find struct_types sdecl.A.sname in
  let type_list = List.map (fun(t, _) -> ltype_of_typ t) sdecl.A.sformals in
  let type_list = Array.of_list(type_list) in
  L.struct_set_body struct_t type_list true
in
  ignore(List.map populate_struct_type structs);
let string_option_to_string = function
  None -> ""
  | Some(s) -> s
in

(*
  struct_field_index is a map where key is struct name and value is another map
  this second map, the key is the field name and the value is the index number
  basically index every field of struct so that they can be accessed later on
*)

let struct_field_index_list =
  let handle_list m individual_struct =
    let struct_field_name_list =
    List.map snd individual_struct.A.sformals in
    let increment n = n + 1 in
    (*add each field and index to second map called struct_field_map*)
    let add_field_and_index (m, i) field_name =
    (StringMap.add field_name (increment i) m, increment i) in
    let struct_field_map =
    List.fold_left add_field_and_index
    (StringMap.empty, -1) struct_field_name_list in
    (*add struct_field_map to the main map*)
    StringMap.add individual_struct.A.sname (fst struct_field_map) m
  in
  List.fold_left handle_list StringMap.empty structs
in

(* Declare printf(), which the print built-in function will call *)
let printf_t = L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
let printf_func = L.declare_function "printf" printf_t the_module in

(* Declare built-in prompt() function *)
let prompt_t = L.function_type i32_t [| str_t |] in
let prompt_func = L.declare_function "prompt" prompt_t the_module in

(* Declare built-in abs() function *)
let abs_t = L.function_type i32_t [|i32_t|] in
let abs_func = L.declare_function "abs" abs_t the_module in

(* Declare built-in print_endline() function *)
let print_endline_t = L.function_type i32_t [||] in
let print_endline_func = L.declare_function
"print_endline" print_endline_t the_module in
```

```
(* Declare built-in print_sameline() function *)
let print_sameline_t = L.function_type i32_t [| str_t |] in
let print_sameline_func = L.declare_function
"print_sameline" print_sameline_t the_module in

(* Declare built-in diceThrow() function *)
let diceThrow_num_gen_t = L.function_type i32_t [||] in
let diceThrow_num_gen_func = L.declare_function
"diceThrow" diceThrow_num_gen_t the_module in

(* Declare built-in getLen() function *)
let getLen_t = L.function_type i32_t [| str_t |] in
let getLen_func = L.declare_function
"getLen" getLen_t the_module in

(* Declare built-in print_int_sameline() function*)
let print_int_sameline_t = L.function_type i32_t [| i32_t |] in
let print_int_sameline_func = L.declare_function
"print_int_sameline" print_int_sameline_t the_module in

let main_func_map = StringMap.add "setup" "main" StringMap.empty in

(* Define each struct function (arguments and return type) so we can call it*)
let struct_function_decls =
  let function_decl m sdecl =
    let fdecl = sdecl.A.sfunc in
      let name = sdecl.A.sname ^ fdecl.A.fname and
      formal_types = Array.of_list
      (List.map (fun (t,_) -> ltype_of_typ t) fdecl.A.formals) in
    let ftype = L.function_type
    (ltype_of_typ fdecl.A.typ) formal_types in
    StringMap.add name (L.define_function name ftype the_module, fdecl) m in
  List.fold_left function_decl StringMap.empty structs in

let function_decls =
  let function_decl m fdecl =
    let get_formal_types formal =
      let (t, _) = formal in
      match t with
        A.Array1DType (_,_) -> L.pointer_type (ltype_of_typ t)
      | A.Array2DType (_, _, _) -> L.pointer_type (ltype_of_typ t)
      | A.StructType _ -> L.pointer_type (ltype_of_typ t)
      | _ -> ltype_of_typ t
    in
    let name = try StringMap.find fdecl.A.fname main_func_map
    with Not_found -> fdecl.A.fname
    and formal_types = Array.of_list
    (List.map get_formal_types fdecl.A.formals) in
    let ftype = L.function_type
    (ltype_of_typ fdecl.A.typ) formal_types in
    StringMap.add name (L.define_function name ftype the_module, fdecl) m
  in
  List.fold_left function_decl StringMap.empty functions in
```

```
(* Fill in the body of the given function *)
let build_function_body the_function fdecl =


  let builder = L.builder_at_end context (L.entry_block the_function) in

  let int_format_str = L.build_global_stringptr "%d\n" "fmt" builder in
  let str_format_str = L.build_global_stringptr "%s\n" "mmt" builder in

  let vars_local:(string, L.llvalue) Hashtbl.t = Hashtbl.create 1000 in

  (* Construct the function's "locals": formal arguments and locally
     declared variables.  Allocate each on the stack, initialize their
     value, if appropriate, and remember their values in the "locals" map *)
  let local_vars =
    let add_formal (t, n) p = L.set_value_name n p;
      match t with
        A.Array1DType (_, _) -> Hashtbl.add vars_local n p
        | A.Array2DType (_, _, _) -> Hashtbl.add vars_local n p
        | A.StructType _ -> Hashtbl.add vars_local n p
        | _ -> let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore (L.build_store p local builder);Hashtbl.add vars_local n local
    in

    let add_local (t, n) =
      let local_var = L.build_alloca (ltype_of_typ t) n builder
    in
    Hashtbl.add vars_local n local_var
  in

  let _ = List.iter2 add_formal
  fdecl.A.formals (Array.to_list (L.params the_function)) in
  List.map add_local fdecl.A.locals
  in
  let local = L.build_alloca i32_t "repeat" builder in
  let _ = Hashtbl.add vars_local "repeat" local in

  (* Return the value for a variable or formal argument *)
  let lookup n = try Hashtbl.find vars_local n
  with Not_found -> try Hashtbl.find vars_global n
  with Not_found -> raise(Failure("Undeclared identifier "^n)) in

  let lookup_at_index s index builder=
     L.build_in_bounds_gep (lookup s)
     (Array.of_list [L.const_int i32_t 0; index]) "name" builder in

  let lookup_at_2d_index s index1 index2 builder=
    L.build_in_bounds_gep (lookup s)
    (Array.of_list [L.const_int i32_t 0; index1; index2]) "name" builder in

  let add_terminal builder f =
    match L.block_terminator (L.insertion_block builder) with
```

```
    Some _ -> ()
    | None -> ignore (f builder) in

let rec llvalue_expr_getter builder = function
   A.Id s -> lookup s
  | A.Array1DAccess (s, e) ->
  let index = expr builder e in lookup_at_index s index builder
  | A.Array2DAccess(s,e1,e2) ->
  let index1 = expr builder e1 and index2 = expr builder e2
  in lookup_at_2d_index s index1 index2 builder

  | A.Dotop(e1, field) ->
    (match e1 with
      A.Id s -> let etype = fst(
          try List.find (fun t -> snd(t) = s) fdecl.A.locals
          with Not_found ->
          try List.find (fun t -> snd(t) = s) fdecl.A.formals
          with Not_found ->
          try let sval = lookup s in
          let llvm_type = L.type_of sval in
          let player_type = ltype_of_typ (A.StructType "Player") in
          (match llvm_type with
            player_type -> (A.PlayerType,s)
            | i32_t -> (A.Int, s)
            | _ -> let struct_name = Hashtbl.find struct_names llvm_type in
                   (A.StructType(struct_name),s))
          with Not_found ->
          raise(Failure("unable to find" ^ s ^ "in structure assignment"))
      )
      in

      (match etype with
        A.StructType t->
          let index_number_list = StringMap.find t struct_field_index_list in
          let index_number = StringMap.find field index_number_list in
          let struct_llvalue = lookup s in
          let access_llvalue = L.build_struct_gep
          struct_llvalue index_number "struct_lvalue" builder in
          access_llvalue
        | A.PointerType _-> let e' = expr builder e1 in
          let e_loaded = L.build_load e' "ptr_deref" builder in
          let e1'_lltype = L.type_of e_loaded in
          let e1'_struct_name_string_option = L.struct_name e1'_lltype in
          let e1'_struct_name_string =
          string_option_to_string e1'_struct_name_string_option in
          let index_number_list =
          StringMap.find e1'_struct_name_string struct_field_index_list in
          let index_number = StringMap.find field index_number_list in
          let access_llvalue = L.build_struct_gep
          e' index_number "struct_lvalue" builder in
          let loaded_access =
          L.build_load access_llvalue "struct_ptr_lvalue" builder in
          loaded_access
        | A.PlayerType -> let t = "Player" in
```

```
            let index_number_list =
            StringMap.find t struct_field_index_list in
            let index_number =
            StringMap.find field index_number_list in
            let struct_llvalue = lookup s in
            let access_llvalue =
            L.build_struct_gep
            struct_llvalue index_number "player_lvalue" builder in
            access_llvalue
          | _ -> raise (Failure
            ("Couldn't match type of " ^s ^ " in structure assignment"))
        )
        | _ as e1_expr ->  (*Handles nested structs*)
          let e1'_llvalue = llvalue_expr_getter builder e1_expr in
          let loaded_e1' = expr builder e1_expr in
          let e1'_lltype = L.type_of loaded_e1' in
          let e1'_struct_name_string_option = L.struct_name e1'_lltype in
          let e1'_struct_name_string =
          string_option_to_string e1'_struct_name_string_option in
          let index_number_list =
          StringMap.find e1'_struct_name_string struct_field_index_list in
          let index_number = StringMap.find field index_number_list in
          let access_llvalue =
          L.build_struct_gep
          e1'_llvalue index_number "nested_struct_lvalue" builder
        in
          access_llvalue
      )

    | A.Unop(op, e)  ->
      (match op with
        A.Deref ->
          let e_llvalue = (llvalue_expr_getter builder e) in
          let e_loaded = L.build_load e_llvalue "ptr_deref" builder in
          e_loaded
        |_ -> raise (Failure("Invalid Unop in llvalue_expr_getter"))
      )
    |_ -> raise (Failure ("Invalid param to llvalue_expr_getter"))

  and expr builder = function
    A.Literal i -> L.const_int i32_t i
    | A.Null t -> L.const_pointer_null
    (ltype_of_typ(A.PointerType(A.StructType(t))))
    | A.BoolLit b -> L.const_int i1_t (if b then 1 else 0)
    | A.Id s -> L.build_load (lookup s) s builder
    | A.GridAssign (e1, e2, e3) ->
      let struct_llvalue = expr builder e3 in
      let struct_type = L.type_of struct_llvalue in
      let struct_name = Hashtbl.find struct_names struct_type in
      (*Create a list node*)
      let struct_listnode_type = Hashtbl.find struct_types "GenericPiece" in
      let struct_val = L.build_alloca (struct_listnode_type) "newNode" builder in
      let _ = Hashtbl.add vars_local "newNode" struct_val in
      (*Assign newNode.owner as the left side of e3*)
```

```
  let owner_val_expr =
  (match e3 with
    A.Dotop(e1, _) ->
                          (*Do a lookup of e1*)
                            A.Unop(A.Ref,e1)
    | A.Id (_) -> A.Null("Player")
    | _ -> raise(Failure("Unknown type for GridAssign")))
  in
  let dotoperator = A.Dotop(A.Id("newNode"), "owner") in
  let _ = expr builder (A.Assign(dotoperator, owner_val_expr)) in

  (*Fill in the appropriate structure inside listNode*)
  let correct_struct =
    let get_good_struct sdecl=
      sdecl.A.sname = "GenericPiece"
    in
    List.filter get_good_struct structs
  in
  let actual_listnode_struct = List.hd correct_struct in

  let name_type_pair_list =
    let is_correct_name struct_pair =
      let current_type = fst(struct_pair) in
      match current_type with
        | A.PointerType t ->
            (match t with
              A.StructType s -> s = struct_name
              | _ -> false)
        | _ -> false
    in
  List.filter is_correct_name actual_listnode_struct.sformals in
  let name_type_pair = List.hd name_type_pair_list in
  let var_name = (snd(name_type_pair)) in
  let dotoperator = A.Dotop(A.Id("newNode"), var_name) in
  let _ = expr builder (A.Assign(dotoperator, A.Unop(A.Ref,e3))) in

  (*Next thing is to assign the tagtype that was filled in*)
  let dotoperator = A.Dotop(A.Id("newNode"), "nametag") in
  let displayString = A.Dotop(e3, "displayString") in
  let _ = expr builder (A.Assign(dotoperator, displayString)) in
  let dotoperator = A.Dotop(A.Id("newNode"), "typetag") in
  let _ = expr builder (A.Assign(dotoperator, A.String_Lit(struct_name))) in
  expr builder
  (A.Call("addToGrid", [e1; e2; A.Unop(A.Ref, (A.Id("newNode")));]))

| A.DeletePiece (e1, e2, e3) ->
  let tag_to_send =
  (match e3 with
  A.Id s -> s
  | A.Dotop (e1, field) -> let left_of_dot =
    (match e1 with
      A.Id(s) -> s
      | _ -> raise(Failure("Invalid dot operation in Delete Player"))) in
    left_of_dot ^ "." ^ field
```

```
    | _ -> raise(Failure("Invalid attempt to delete from grid"))
        )
  in
  expr builder
  (A.Call("deleteFromGrid", [e1; e2; A.String_Lit(tag_to_send)]));

| A.Dotop(e1, field) ->
let e' = expr builder e1 in
  (match e1 with
    A.Id s ->
      let etype =
      fst(
        try List.find (fun t -> snd(t) = s) fdecl.A.locals
        with Not_found ->
        try List.find (fun t -> snd(t) = s) fdecl.A.formals
        with Not_found -> try let sval = lookup s in  (*Check in globals*)
          let llvm_type = L.type_of sval in
          let player_type = ltype_of_typ (A.StructType "Player") in
          (match llvm_type with
            player_type -> (A.PlayerType,s)
            | _ -> let struct_name = Hashtbl.find struct_names llvm_type in
                   (A.StructType(struct_name),s))
        with Not_found ->
          raise(Failure("Unable to find" ^ s ^ "in Dotop expr"))
        )
        in
        (try match etype with
          A.StructType t->
            let index_number_list =
            StringMap.find t struct_field_index_list in
            let index_number = StringMap.find field index_number_list in
            let struct_llvalue = lookup s in
            let access_llvalue =
            L.build_struct_gep struct_llvalue
            index_number "dotop_terminal" builder in
            let loaded_access =
            L.build_load access_llvalue "loaded_dotop_terminal" builder in
            loaded_access
          | A.PointerType _->
            let e_loaded = L.build_load e' "loaded_deref" builder in
            let e1'_lltype = L.type_of e_loaded in
            let e1'_struct_name_string_option = L.struct_name e1'_lltype in
            let e1'_struct_name_string =
            string_option_to_string e1'_struct_name_string_option in
            let index_number_list =
            StringMap.find e1'_struct_name_string struct_field_index_list in
            let index_number = StringMap.find field index_number_list in
            let access_llvalue =
            L.build_struct_gep e' index_number "dotop_terminal" builder in
            let loaded_access =
            L.build_load access_llvalue "loaded_dotop_terminal" builder in
            loaded_access
          | A.PlayerType -> let t = "Player" in
            let index_number_list =
```

```
            StringMap.find t struct_field_index_list in
            let index_number = StringMap.find field index_number_list in
            let struct_llvalue = lookup s in
            let access_llvalue =
            L.build_struct_gep
            struct_llvalue index_number "dotop_terminal" builder in
            let loaded_access =
            L.build_load access_llvalue "loaded_dotop_terminal" builder in
            loaded_access
          | _ -> raise (Failure("No structype."))
          with Not_found -> raise (Failure("unable to find" ^ s))
        )
      | _ as e1_expr ->
      let e1'_llvalue = llvalue_expr_getter builder e1_expr in
      let loaded_e1' = expr builder e1_expr in
      let e1'_lltype = L.type_of loaded_e1' in
      let e1'_struct_name_string_option = L.struct_name e1'_lltype in
      let e1'_struct_name_string =
      string_option_to_string e1'_struct_name_string_option in
      let index_number_list =
      StringMap.find e1'_struct_name_string struct_field_index_list in
      let index_number = StringMap.find field index_number_list in
      let access_llvalue =
      L.build_struct_gep e1'_llvalue index_number "gep_in_dotop" builder in
      L.build_load access_llvalue "loaded_dotop" builder
    )

| A.Unop(op, e) ->
  let e' = expr builder e in
  (match op with
    A.Neg -> L.build_neg e' "Neg_op" builder
    | A.Not     -> L.build_not e' "Not_op" builder
    | A.Deref -> L.build_load e' "Deref_op" builder
    | A.Ref -> (llvalue_expr_getter builder e)
    | _ -> raise(Failure("Invalid unary operation"))
  )
| A.Assign (lhs, e2) ->
  let e2' = expr builder e2 in
  (match lhs with
    | A.Array1DAssign (array_name, i, v) ->
      let addr = (let index = expr builder i in
      lookup_at_index array_name index builder)
      and value = expr builder v in
      ignore(L.build_store value addr builder); value

    | A.Array2DAssign(array_name, i,j,v) ->
      let addr = (let index1 = expr builder i
      and index2 = expr builder j in
      lookup_at_2d_index array_name index1 index2 builder)
      and value = expr builder v in
      ignore(L.build_store value addr builder); value

    |A.Id s ->ignore (L.build_store e2' (lookup s) builder); e2'
```

```
|A.Dotop (e1, field) ->
  let e' = expr builder e1 in
    (match e1 with
      A.Id s ->
        let e1typ =
        (match s with
        "newNode" ->  A.StructType ("GenericPiece")
        | _ ->
          fst(
            try List.find (fun t -> snd(t) = s) fdecl.A.locals
            with Not_found ->
            try List.find (fun t -> snd(t) = s) fdecl.A.formals
            with Not_found ->
            try let sval = lookup s in
            let llvm_type = L.type_of sval in
            let player_type = ltype_of_typ (A.StructType "Player") in
            (match llvm_type with
              player_type -> (A.PlayerType,s)
              | _ -> let struct_name =
              Hashtbl.find struct_names llvm_type in
                      (A.StructType(struct_name),s))
            with Not_found ->
            raise(Failure
            ("unable to find" ^ s ^ "in structure assignment"))
        ))
        in
      (match e1typ with
      A.StructType t -> (try
        let index_number_list =
        StringMap.find t struct_field_index_list in
        let index_number = StringMap.find field index_number_list in
        let struct_llvalue = lookup s in
        let access_llvalue =
        L.build_struct_gep struct_llvalue index_number field builder in
        (try (ignore(L.build_store e2' access_llvalue builder);e2')
          with Not_found -> raise (Failure("unable to store " ^ t ))
        )
        with Not_found -> raise (Failure("unable to find" ^ s)) )

      | A.PointerType _ ->
      let e_loaded = L.build_load e' "loaded_deref" builder in
      let e1'_lltype = L.type_of e_loaded in
      let e1'_struct_name_string_option = L.struct_name e1'_lltype in
      let e1'_struct_name_string =
      string_option_to_string e1'_struct_name_string_option in
      let index_number_list =
      StringMap.find e1'_struct_name_string struct_field_index_list in
      let index_number = StringMap.find field index_number_list in
      let access_llvalue =
      L.build_struct_gep e' index_number field builder in
      (try (ignore(L.build_store e2' access_llvalue builder);e2')
        with Not_found -> raise (Failure("unable to store error" ))
      )
      | A.PlayerType ->
```

```
          let t = "Player" in
                          (try
            let index_number_list =
            StringMap.find t struct_field_index_list in
            let index_number = StringMap.find field index_number_list in
            let struct_llvalue = lookup s in
            let access_llvalue =
            L.build_struct_gep struct_llvalue index_number field builder in
            (try (ignore(L.build_store e2' access_llvalue builder);e2')
              with Not_found -> raise (Failure("unable to store " ^ t ))
            )
            with Not_found -> raise (Failure("unable to find" ^ s)) )

          | _ -> raise (Failure("StructType not found."))
        )
        |_ as e1_expr ->
        let e1'_llvalue = llvalue_expr_getter builder e1_expr in
        let loaded_e1' = expr builder e1_expr in
        let e1'_lltype = L.type_of loaded_e1' in
        let e1'_struct_name_string_option = L.struct_name e1'_lltype in
        let e1'_struct_name_string =
        string_option_to_string e1'_struct_name_string_option in
        let index_number_list =
        StringMap.find e1'_struct_name_string struct_field_index_list in
        let index_number =
        StringMap.find field index_number_list in
        let access_llvalue =
        L.build_struct_gep
        e1'_llvalue index_number "gep_in_Sassign" builder in
        let _ = L.build_store e2' access_llvalue builder in
        e2'
      )
    |A.Unop(op, e) ->
      (match op with
        A.Deref ->
          let e_llvalue = (llvalue_expr_getter builder e) in
          let e_loaded = L.build_load e_llvalue "loaded_deref" builder in
          let _ = L.build_store e2' e_loaded builder in
          e2'
        |_ -> raise (Failure("Invalid unary operation"))
      )
    |_ -> raise (Failure("can't match in assign"))
  )
| A.Array1DAssign (array_name, i, v) ->
  let addr = (let index = expr builder i in
  lookup_at_index array_name index builder)
  and value = expr builder v in
  ignore(L.build_store value addr builder); value
| A.Array2DAssign(array_name, i,j,v) ->
  let addr = (let index1 = expr builder i and
  index2 = expr builder j in
  lookup_at_2d_index array_name index1 index2 builder)
  and value = expr builder v in ignore(L.build_store value addr builder);
  value
```

```
| A.String_Lit(s) -> L.build_global_stringptr s "name" builder

| A.Array1DAccess (s, e) ->
let index = expr builder e in L.build_load (lookup_at_index s index builder)
"name" builder
| A.Array2DAccess(s,e1,e2) ->
let index1 = expr builder e1 and index2 = expr builder e2
in L.build_load(lookup_at_2d_index s index1 index2 builder) "name" builder
| A.ArrayLiteral (params) ->
let val_zero = expr builder (List.hd params) in
let val_type = L.type_of val_zero in
L.const_array val_type (Array.of_list (List.map (expr builder) params))
| A.Binop (e1, op, e2) ->
  (* Construct code for an expression; return its value *)
    let e1' = expr builder e1
    and e2' = expr builder e2 in
    (match op with
        A.Add     -> L.build_add
      | A.Sub     -> L.build_sub
      | A.Mult    -> L.build_mul
      | A.Div     -> L.build_sdiv
      | A.And     -> L.build_and
      | A.Or      -> L.build_or
      | A.Modulo  -> L.build_urem
      | A.Equal   -> L.build_icmp L.Icmp.Eq
      | A.Neq     -> L.build_icmp L.Icmp.Ne
      | A.Less    -> L.build_icmp L.Icmp.Slt
      | A.Leq     -> L.build_icmp L.Icmp.Sle
      | A.Greater -> L.build_icmp L.Icmp.Sgt
      | A.Geq     -> L.build_icmp L.Icmp.Sge
    ) e1' e2' "tmp" builder
  | A.Unop(op, e) ->
    let e' = expr builder e in
    (match op with
      A.Neg     -> L.build_neg e' "tmp" builder
      | A.Not     -> L.build_not e' "tmp" builder
      | A.Ref -> llvalue_expr_getter builder e
      | A.Deref -> L.build_load e' "tmp" builder )
| A.Call ("print", [e]) ->
      let e' = expr builder e in
        if (L.type_of e' = i32_t || L.type_of e' = i1_t) then
          L.build_call printf_func [| int_format_str ; (expr builder e) |]
          "printf" builder
      else
        L.build_call printf_func [| str_format_str ; (expr builder e) |]
        "printf" builder

| A.Call ("prompt", [e]) ->
    L.build_call prompt_func [|expr builder e|] "prompt" builder

| A.Call ("abs", [e]) ->
    L.build_call abs_func [|expr builder e|] "absl" builder

| A.Call ("print_endline", []) ->
```

```
    L.build_call print_endline_func [||] "print_endline" builder

| A.Call ("print_sameline", [e]) ->
L.build_call print_sameline_func
[| expr builder e |] "print_sameline" builder

| A.Call ("diceThrow", []) ->
    L.build_call diceThrow_num_gen_func [||] "diceThrow" builder

| A.Call("getLen",[e])  ->
    L.build_call getLen_func [|expr builder e|] "getLen" builder

| A.Call("print_int_sameline",[e])  ->
  L.build_call print_int_sameline_func [|expr builder e|]
  "print_int_sameline" builder

| A.Call (f, act) ->
  let map_arguments actual =
    (match actual with
    A.Id s ->
      (match s with
        "newNode" -> expr builder actual
        |_ ->
          let etype =
            fst(
            try List.find (fun t->snd(t)=s) fdecl.A.locals with
            |Not_found -> List.find (fun t->snd(t)=s) fdecl.A.formals
            |Not_found -> try let sval = lookup s in
            let llvm_type = L.type_of sval in
            let player_type = ltype_of_typ (A.StructType "Player") in
            (match llvm_type with
              player_type -> (A.PlayerType,s)
              | i32_t -> (A.Int, s)
              | _ -> let struct_name = Hashtbl.find struct_names llvm_type in
                     (A.StructType(struct_name),s))
          with Not_found -> raise
          (Failure("Unable to find" ^ s ^ "in map_arguments ID")))

        in
        (match etype with
          A.Array1DType (_,_)-> llvalue_expr_getter builder (actual)
          | A.Array2DType (_, _, _) -> llvalue_expr_getter builder (actual)
          | A.StructType _ -> llvalue_expr_getter builder (actual)
          | _ -> expr builder actual))
      | _ -> expr builder actual)
      in
      (match f with

      |_ -> let (fdef, fdecl_called) =
      try StringMap.find f function_decls
      with Not_found -> StringMap.find f struct_function_decls in
      let actuals = List.rev (List.map map_arguments (List.rev act)) in
      let result = (match fdecl_called.A.typ with
        A.Void -> ""
```

```
        | _ -> f ^ "_result")
      in L.build_call fdef (Array.of_list actuals) result builder)
| _ -> raise(Failure("Expr builder failed"))

(* Build the code for the given statement; return the builder for
   the statement's successor *)
and stmt builder arg_to_match =
let builder =
if !internal_if_flag = 1 then
  !new_global_builder
else builder
in
ignore(internal_if_flag := 0);
match arg_to_match with
  A.Block sl -> List.fold_left stmt builder sl
| A.Expr e -> ignore (expr builder e); builder
| A.If (predicate, then_stmt, else_stmt) ->
  let bool_val = expr builder predicate in
  let merge_bb = L.append_block context "merge" the_function in
  let then_bb = L.append_block context "then" the_function in
  add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
  (L.build_br merge_bb);
   let else_bb = L.append_block context "else" the_function in
   add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
   (L.build_br merge_bb);
   ignore (L.build_cond_br bool_val then_bb else_bb builder);
   L.builder_at_end context merge_bb

| A.While (predicate, body) ->
  let pred_bb = L.append_block context "while" the_function in
  ignore (L.build_br pred_bb builder);
  let body_bb = L.append_block context "while_body" the_function in
  add_terminal (stmt (L.builder_at_end context body_bb) body) (L.build_br pred_bb);
  let pred_builder = L.builder_at_end context pred_bb in
  let bool_val = expr pred_builder predicate in
  let merge_bb = L.append_block context "merge" the_function in
  ignore (L.build_cond_br bool_val body_bb merge_bb pred_builder);
  L.builder_at_end context merge_bb
| A.For (e1, e2, e3, body) -> stmt builder
  ( A.Block [A.Expr e1 ; A.While (e2, A.Block [body ; A.Expr e3]) ] )
| A.Return e -> if (fdecl.A.fname = "gameloop") then
  ignore(expr builder
  (A.Assign(A.Id("repeat"),A.Call ("checkGameEnd", [])))))
  else
  ignore (match fdecl.A.typ with
    A.Void -> L.build_ret_void builder
    | _ -> L.build_ret (expr builder e) builder); builder
in


(* Build the code for each statement in the function *)
let builder = if (fdecl.A.fname = "gameloop") then
  let _ = ignore(expr builder (A.Assign(A.Id("repeat"),A.Literal(0)))) in
  let _ = ignore (expr builder
```

```
        ( A.Assign(A.Id("currentPlayerIndex"), A.Literal(0)))) in
      let add_op = A.Binop(A.Id("currentPlayerIndex"),A.Add,A.Literal(1)) in
      let assign_this = A.Binop(add_op, A.Modulo, A.Id("playerOrderSize")) in
      let bodyWithIndex =  fdecl.A.body @
      [ A.Expr ( A.Assign(A.Id("currentPlayerIndex"), assign_this)) ] in
      stmt builder
      (A.While(A.Binop(A.Id("repeat"),A.Equal,A.Literal(0)),
       A.Block bodyWithIndex))
      else stmt builder (A.Block fdecl.A.body)
    in

    (* Add a return if the last block falls off the end *)
    add_terminal builder (match fdecl.A.typ with
                A.Void -> L.build_ret_void
                | t -> L.build_ret (L.const_int (ltype_of_typ t) 0))
  in


(*Build the function bodies for struct functions*)
let the_struct_function_list =
  let get_struct_func_decls sdecl =
    let fdecl = sdecl.A.sfunc in
      let (the_function, _) =
      StringMap.find (sdecl.A.sname ^ fdecl.A.fname) struct_function_decls
    in the_function
  in List.map get_struct_func_decls structs
in
List.iter2 build_function_body
the_struct_function_list (List.map (fun (sdecl) -> sdecl.A.sfunc) structs);
(*Build the function bodies*)
let the_function_list =
  let get_func_decls fdecl =
    let (the_function, _) =
      try StringMap.find fdecl.A.fname function_decls
    with Not_found -> StringMap.find "main" function_decls
    in the_function
  in
  List.map get_func_decls functions
in
List.iter2 build_function_body the_function_list functions;


the_module
```

gridBasics.grid

```
void addToGrid(int x, int y, Piece GenericPiece* p_n){
        /*Set location of item p_n somewhere here*/
        /*Having trouble setting a nested struct with a pointer on the left-most side. Lea
        Piece GenericPiece* iterator;
        p_n.x = x;
        p_n.y = y;
  if(GridData[x,y] == None)
  {
                GridData[x,y] = p_n;
```

```
                iterator = GridData[x,y];
                iterator.next = None;
                return;
        }
        else
        {
                iterator = GridData[x,y];
          while(iterator.next != None)
          {
                        iterator = iterator.next;
                }
                iterator.next = p_n;
                iterator = iterator.next;
                iterator.next = None;

                /*Go through list to run colocation*/
                iterator = GridData[x,y];
          while(iterator != None)
          {
                        if (iterator != p_n)
                        {
                                colocation(x, y, p_n, iterator);
                        }
                        iterator = iterator.next;
                }

        }

}


void deleteFromGrid(int x, int y, string tag){

        Piece GenericPiece* iterator;
        Piece GenericPiece* next_iterator;
        iterator = GridData[x,y];
        next_iterator = iterator.next;

        if(iterator.nametag == tag){
                GridData[x,y] = iterator.next;
                iterator.next = None;
                iterator = None;
                return;
        }

        while(iterator.next != None ){
                if(next_iterator.nametag == tag){
                        iterator.next = next_iterator.next;
                        next_iterator.next = None;
                        next_iterator = None;
                        return;
                }
                next_iterator = next_iterator.next;
                iterator = iterator.next;
```

```
        }

        if(iterator.next == None){
                print("Not found on given coordinate");
                return;
        }

}

int printGrid(){
        int x;
        int y;
        int i;
        int k;
        int width;
        int tempLen;
        int flag;
        int max_width;
        int border_len;
        string printer;
        Piece GenericPiece* iterator;
        width = 0;
        max_width = 0;
        border_len = 0;

        for(x = 0; x < rows; x = x+1){
                for(y = 0; y < cols; y = y+1){
                        iterator = GridData[x,y];
                        width = 0;
                        if(iterator!=None){
                                width = width + getLen(iterator.nametag);
                                iterator = iterator.next;
                        }
                        while(iterator != None ){
                                width = width + 2;
                                width = width + getLen(iterator.nametag);
                                iterator = iterator.next;
                        }
                        if(width>max_width){
                                max_width = width;
                        }
                }
        }

        border_len = max_width * cols + (cols);

        for (i = 0; i < border_len; i=i+1)
        {
                print_sameline("-");
        }
        print_endline();

        /* printing column numbers*/
```

```
        for(i=0;i<cols;i=i+1){
                for(k=0; k < max_width/2; k=k+1){
                        print_sameline("_");
                }
                print_int_sameline(i);
                for(k=0; k < max_width-(max_width/2); k=k+1){
                        print_sameline("_");
                }

        }

        print_endline();

        for(x = 0; x < rows; x = x+1)
        {
                for(y = 0; y < cols; y = y+1)
                {
                        tempLen = 0;
                        print_sameline("|");
                        iterator = GridData[x,y];
                        if(iterator!=None){
                                printer = iterator.nametag;
                                tempLen = tempLen + getLen(printer);
                                print_sameline(printer);
                                iterator = iterator.next;
                        }
                        while(iterator !=None)
                        {
                                print_sameline(", ");
                                printer = iterator.nametag;
                                tempLen = tempLen + getLen(printer);
                                tempLen = tempLen +2;
                                print_sameline(printer);
                                iterator = iterator.next;
                        }
                        for(k=0;k<max_width-tempLen;k=k+1){
                                print_sameline(" ");
                        }
                }
                print_sameline("|");
                print_int_sameline(x);
                print_endline();
        }

        for (i = 0; i < border_len; i=i+1)
        {
                print_sameline("-");
        }
        print_endline();
        return 0;
}

int moveOnGrid(Piece GenericPiece* p_n, int dst_x, int dst_y)
{
```

```
        int result, src_x, src_y;
        if (p_n != None)
        {
                src_x = p_n.x;
                src_y = p_n.y;
                result = triggerRule(src_x, src_y, dst_x, dst_y, p_n.typetag);
                if (result == 1)
                {
                        deleteFromGrid(src_x, src_y, p_n.nametag);
                        addToGrid(dst_x, dst_y, p_n);
                }
                else
                {
                        currentPlayerIndex = currentPlayerIndex - 1;
                }
        }
        else
        {
                print("No piece on cell");
                currentPlayerIndex = currentPlayerIndex - 1;
                return 0;
        }
        return result;
}

Piece GenericPiece* getPieceAtLocation(int x, int y){
        Piece GenericPiece* head;
        head = GridData[x,y];
        return head;

}

Piece GenericPiece* getPieceFromGrid(string displayString)
{
        Piece GenericPiece* iterator;
        int x;
        int y;
        for(x = 0; x < rows; x = x+1)
        {
                for(y = 0; y < cols; y = y+1)
                {
                        iterator = GridData[x,y];
                        while(iterator != None )
                        {
                                if (iterator.nametag == displayString)
                                {
                                        return iterator;
                                }
                                iterator = iterator.next;
                        }
                }
        }
        iterator = None;
        return iterator;
```

```
}

int checkBound(int x,int y){
        if(x > -1 && x <= rows && y>-1 && y <= cols){
                return 1;
        }
        else{
                return 0;
        }
}

/*Currently assumes only x is different (column-wise)*/

int traverse(int src_x, int src_y, int dst_x, int dst_y)
{
        int gx;
        int sx;
        int tx;
        int gy;
        int sy;
        int start_x;
        int start_y;
        int end_x;
        int end_y;
        Piece GenericPiece* iter;

        if(dst_y == src_y){
                if (src_x > dst_x){
                        gx = src_x;
                        sx = dst_x;
                }
                else{
                        gx = dst_x;
                        sx = src_x;
                }

                for(tx = sx+1; tx < gx; tx = tx+1){
                        iter = GridData[tx,src_y];
                        if (iter != None){
                                return 1;
                        }
                }
                return 0;
        }

        if(src_x == dst_x){
                if (src_y > dst_y){
                        gy = src_y;
                        sy = dst_y;
                }
                else{
                        gy = dst_y;
                        sy = src_y;
                }
```

```
        for(tx = sy+1; tx < gy; tx = tx+1){
                iter = GridData[src_x,tx];
                if (iter != None){
                        return 1;
                }
        }
        return 0;
}

start_x = src_x;
start_y = src_y;
end_x = dst_x;
end_y = dst_y;
if(abs((dst_y - src_y)/(dst_x - src_x)) == 1){
        if((dst_x > src_x) && (dst_y > src_y)){
                for(tx = start_x + 1; tx < end_x; tx = tx+1){
                        start_y = start_y + 1;
                        iter = GridData[tx,start_y];
                        if (iter != None){
                                return 1;
                        }
                }
                return 0;

        }
        if((dst_x > src_x) && (dst_y < src_y)){
                for(tx = start_x + 1; tx < end_x; tx = tx+1){
                        start_y = start_y - 1;
                        iter = GridData[tx,start_y];
                        if (iter != None){
                                return 1;
                        }
                }
                return 0;
        }
        if((dst_x < src_x) && (dst_y < src_y)){
                for(tx = start_x - 1; tx > end_x; tx = tx-1){
                        start_y = start_y - 1;
                        iter = GridData[tx,start_y];
                        if (iter != None){
                                return 1;
                        }
                }
                return 0;
        }
        if((dst_x < src_x) && (dst_y > src_y)){
                for(tx = start_x - 1; tx > end_x; tx = tx - 1){
                        start_y = start_y + 1;
                        iter = GridData[tx,start_y];
                        if (iter != None){
                                return 1;
                        }
                }
```

```
                        return 0;
                }

        }

        return 0;
}

void nextPlayer()
{
        currentPlayerIndex = (currentPlayerIndex + 1) % playerOrderSize;
}
```

bindings.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>

// get length of a string
int getLen(char* x){
   return strlen(x);
}

// get user input
int prompt(char* msg)
{
   // static char str[50];
   printf("%s",msg);
   static int x;
   scanf("%d", &x);
   return x;
}

// print to console with a newline
int print_endline(){
   printf("\n");
   return 0;
}

// print string value of integer
int print_int_sameline(int x){
   printf("%d",x);
   return 0;
}

// print to console without a newline
int print_sameline(char *s){
   printf("%s", s);
   return 0;
}

// random number generator
int diceThrow()
```

```
{
   time_t t;
   srand((unsigned) time(&t));
   return((rand() % 6)+1);
}

// return absolute value of an integer
int absl(int x){
        return abs(x);
}
```

fail-assign1.grid

```
int setup(){
  int i;
  bool b;
  i = 42;
  b = true;
  i = false; /* Fail: assigning a bool to an integer */
  return 0;
}
```

fail-assign2.grid

```
int setup()
{
  int i;
  bool b;

  b = 48; /* Fail: assigning an integer to a bool */
}
```

fail-assign3.grid

```
void myvoid()
{
  return;
}

int setup()
{
  int i;

  i = myvoid(); /* Fail: assigning a void to an integer */
}
```

fail-dead1.grid

```
int setup()
{
  int i;

  i = 15;
  return i;
  i = 32; /* Error: code after a return */
}
```

fail-dead2.grid

```
int setup()
{
  int i;

  {
    i = 15;
    return i;
  }
  i = 32; /* Error: code after a return */
}
```

fail-expr1.grid

```
int a;
bool b;

void foo(int c, bool d)
{
  int dd;
  bool e;
  a + c;
  c - a;
  a * 3;
  c / 2;
  d + a; /* Error: bool + int */
}

int setup()
{
  return 0;
}
```

fail-expr2.grid

```
int a;
bool b;

void foo(int c, bool d)
{
  int d;
  bool e;
  b + a; /* Error: bool + int */
}

int setup()
{
  return 0;
}
```

fail-for1.grid

```
int setup()
{
  int i;
  for ( ; true ; ) {} /* OK: Forever */

  for (i = 0 ; i < 10 ; i = i + 1) {
```

```
    if (i == 3) return 42;
  }

  for (j = 0; i < 10 ; i = i + 1) {} /* j undefined */

  return 0;
}
```

fail-for2.grid

```
int setup()
{
  int i;

  for (i = 0; j < 10 ; i = i + 1) {} /* j undefined */

  return 0;
}
```

fail-for3.grid

```
int setup()
{
  int i;

  for (i = 0; i ; i = i + 1) {} /* i is an integer, not Boolean */

  return 0;
}
```

fail-for4.grid

```
int setup()
{
  int i;

  for (i = 0; i < 10 ; i = j + 1) {} /* j undefined */

  return 0;
}
```

fail-for5.grid

```
int setup()
{
  int i;

  for (i = 0; i < 10 ; i = j + 1) {} /* j undefined */

  return 0;
}
```

fail-for6.grid

```
int setup()
{
  int i;

  for (i = 0; i < 10 ; i = i + 1) {
```

```
    foo(); /* Error: no function foo */
  }

  return 0;
}
```

fail-func1.grid

```
int foo() {}

int bar() {}

int baz() {}

void bar() {} /* Error: duplicate function bar */

int setup()
{
  return 0;
}
```

fail-func2.grid

```
int foo(int a, bool b, int c) { }

void bar(int a, bool b, int a) {} /* Error: duplicate formal a in bar */

int setup()
{
  return 0;
}
```

fail-func3.grid

```
int foo(int a, bool b, int c) { }

void bar(int a, void b, int c) {} /* Error: illegal void formal b */

int setup()
{
  return 0;
}
```

fail-func4.grid

```
int foo() {}

void bar() {}

int print() {} /* Should not be able to define print */

void baz() {}

int setup()
{
  return 0;
}
```

fail-func5.grid

```
int foo() {}

int bar() {
  int a;
  void b; /* Error: illegal void local b */
  bool c;

  return 0;
}

int setup()
{
  return 0;
}
```

fail-func6.grid

```
void foo(int a, bool b)
{
}

int setup()
{
  foo(42, true);
  foo(42); /* Wrong number of arguments */
}
```

fail-func7.grid

```
void foo(int a, bool b)
{
}

int setup()
{
  foo(42, true);
  foo(42, true, false); /* Wrong number of arguments */
}
```

fail-func8.grid

```
void foo(int a, bool b)
{
}

void bar()
{
}

int setup()
{
  foo(42, true);
  foo(42, bar()); /* int and void, not int and bool */
}
```

fail-func9.grid

```
void foo(int a, bool b)
```

```
{
}

int setup()
{
  foo(42, true);
  foo(42, 42); /* Fail: int, not bool */
}
```

fail-global1.grid

```
int c;
bool b;
void a; /* global variables should not be void */


int setup()
{
  return 0;
}
```

fail-player-displaystring.grid

```
import gridBasics.grid;

Player
{
        int x,y;
        string s;
        Player* next;
}

Grid_Init<5,4>;

int setup()
{
        Player p1,p2;
        p1.displayString = "true";
        print(p1.displayString);
        p2.x = 3;
        print(p2.x);
        return 0;
}
```

fail-semant-check-print.grid

```
Player{
        int x;
        Player *next;
}

int setup()
{
        Player* p4;
        print(p4);
        return 0;
}
```

fail-semant-check-print.grid

```
import gridBasics.grid;
Player
{
        int x,y;
        string s;
        Player* next;
}

Grid_Init<5,4>;

int setup()
{
        Player p1, p2;
        p2.z = 3;
        return 0;
}
```

fail-semant-undeclared-struct.grid

```
Player
{
        int x,y;
        string s;
        Player* next;
}

Grid_Init<5,4>;

int setup()
{
        Player p1;
        p2.x = 3;
        print(p2.x);
        return 0;

}
```

test-access-1d-array.grid

```
int setup(){
    int[4] x;
    x = [1,2,3,4];
    x[0] = 5;
    print(x[0]);
    return 0;
}
```

test-access-2d-array.grid

```
int setup(){
        int[2,2] y;
    y[0,0] = 0;
    y[0,1] = 1;
    y[1,0] = 2;
    y[1,1] = 3;
```

```
    print(y[0,1]);
    return 0;
}
```

test-add-ints.grid

```
int setup(){
  int i,j,k;
  i = 3;
  j = 4;
  k = i+j;
  print(k);
  return 0;
}
```

test-arith.grid

```
int setup(){
  int i,j,k,l,a;
  i = 3;
  j = 4;
  k = -1;
  l = 2;
  a = i*j/l-k;
  print(a);
  return 0;
}
```

test-array2dinit.grid

```
int setup(){
    int[4,4] y;
    print("array2d init success");
    return 0;
}
```

test-file-import.grid

```
import gridBasics.grid;

Grid_Init<0,0>;
int setup(){
        print("Import success");
        return 0;
}

int setup(){
    int[4] x;
    x = [1,2,3,4];
    print("Success.1d array intialized.");
    return 0;
}

import gridBasics.grid

int checkGameEnd()
{
  return 1;
```

```
}

Piece Pawn
{
  int something;
  string displayString;
}

Player
{
  Piece Pawn pawn;
  string color;
}

int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
  print("Inside colocation");
  return 0;
}

Grid_Init<5,4>;
Player p1,p2;
Player[2] playerOrder;
int count;

int setup(){
  p1.color = "White";
  p2.color = "Black";
  playerOrder[0] = p1;
  playerOrder[1] = p2;
  playerOrderSize = 2;
  p1.pawn.displayString = "pawn1";
  p2.pawn.displayString = "pawn2";
  Grid <1,2> <-- p1.pawn;
  Grid <3,2> <-- p2.pawn;
  return 0;
}

int gameloop(){
  int src_x,src_y,dst_x,dst_y;
  Player cur;
  Piece GenericPiece* l;
  printGrid();
  cur = playerOrder[currentPlayerIndex];
  print(cur.color);
  src_x = 1;
  src_y = 2;
  dst_x = 2;
  dst_y = 2;
  l = getPieceAtLocation(src_x, src_y);
  moveOnGrid(l, dst_x,dst_y);
  printGrid();
  return 0;
}
```

```
import gridBasics.grid

int checkGameEnd()
 {
  return 1;
 }

Player
{
int x;
int y;
string s;
Piece horse h1;
}

Piece horse
{
int x;
int y;
string s;

}

int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
  return 0;
}

Grid_Init<0,0>;

int setup(){
  return 0;
}

int gameloop()
{
Player p1;
p1.x = 10;
p1.s = "nested structs work";
p1.h1.x = 100;
p1.h1.y = 200;
p1.h1.s = "nested horse works";

print(p1.x);
print(p1.s);
print(p1.h1.x);
print(p1.h1.y);
print(p1.h1.s);

return 0;
}

import gridBasics.grid
```

```
int checkGameEnd()
 {
  return 1;
 }

Player
{
  string name;
}

int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
  return 0;
}

Grid_Init<0,0>;

int setup(){
  return 0;
}

int gameloop(){
    int j;
    int[4] x;
    x = [1,2,3,4];
    for(j=0;j<4;j=j+1){
        print_int_sameline(x[j]);
    }
    return 0;
}
```

test-print-boolean.grid

```
int setup(){
  bool x;
  x = true;
  print(x);
  return 0;
}
```

test-print-int-var.grid

```
int setup(){
  int x;
  x = 6;
  print(x);
  return 0;
}
```

test-print-int.grid

```
int setup(){
  print(6);
  return 0;
}
```

test-print-string-literal.grid

```
int setup()
{
  print("Hello World");
  return 0;
}
```

test-print-string-var.grid

```
int setup(){
  string s;
  s = "hello";
  print(s);
  return 0;
}
```

test-struct-1Darray.grid

```
Player{
        int x,y;
        string s;
}

int setup()
{

Player p1;
Player[2] parray;
p1.x = 3;
p1.y = 4;
p1.s = "1D array of structs work";
parray[0] = p1;
print(parray[0].x);
print(parray[0].y);
print(parray[0].s);

return 0;
}
```

test-struct-2Darray.grid

```
Player{
        int x,y;
        string s;
}

int setup()
{
Player p1;
Player[2,2] parray;
p1.x = 3;
p1.y = 4;
p1.s = "2D array of structs work";
parray[0,0] = p1;
print(parray[0,0].x);
print(parray[0,0].y);
```

```
print(parray[0,0].s);
return 0;
}
```

test-struct-pointer.grid

```
Player{
        int x,y;
        string s;
        Player *next;
}



int setup()
{
Player p1,p2,p3;

p2.x = 10;
p2.y = 20;
p2.s = "struct pointer works";

p1.next = &p2;
p3 = *(p1.next);

print(p3.x);
print(p3.y);
print(p3.s);

return 0;
}
```

test-struct.grid

```
Player{
        int x,y;
        string s;
}

int setup()
{
  Player p1;
  p1.x = 10;
  p1.y = 20;
  p1.s = "struct works";
  print(p1.x);
  print(p1.y);
  print(p1.s);
  return 0;
}
```

test-struct1D-pointer.grid

```
Player{
        int x;
        Player *next;
}
```

```
int setup()
{
Player p1,p2,p3;
Player* p4;
Player[2] parray;
p2.x = 3;

parray[0] = p1;
parray[0].next = &p2;
p4 = parray[0].next;
p3 = *(parray[0].next);
print(p3.x);
return 0;
}
```

test-while.grid

```
int setup(){
    int i;
    int j;
    i=5;
    j=0;
    while(j<i){
        print(1);
        j = j+1;
    }
    return 0;
}
```

gridrun.sh

```
#!/bin/bash
# If the terminal shows "can't find command", please run "chmod +x gridrun.sh" in terminal
LLI="lli"

# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

GRID_NATIVE="./grid.native"

DEFAULT_PATH="."

Run() {
    eval $*
}

Usage(){
    echo "Usage: ./gridrun.sh [options] [.grid file]"
    echo "[options (one at a time)]"
    echo "-c    Compile file"
    echo "-r    Run file"
    echo "-h    Print this help"
```

```
    exit 1
}

CompileFile(){
    IFS='.' read -ra SPLIT_ARRAY <<< "$1"
    basename=`echo ${SPLIT_ARRAY[0]}`
    echo "# compiling ${basename}.grid"
    Run "$GRID_NATIVE " "-c " $1 " > " "${DEFAULT_PATH}/${basename}.ll"
    echo "      ${basename}.ll ... Done"
    Run "$LLC" "${DEFAULT_PATH}/${basename}.ll" ">" "${DEFAULT_PATH}/${basename}.s"
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "bindings.o"
    Run "./${basename}.exe" ">" "${DEFAULT_PATH}/${basename}.out"
    echo "      ${basename}.out ... Done"
}

RunProgram(){
    IFS='.' read -ra SPLIT_ARRAY <<< "$1"
    basename=`echo ${SPLIT_ARRAY[0]}`
    Run "$GRID_NATIVE " "-c " $1 " > " "${DEFAULT_PATH}/${basename}.ll"
    echo "# Executing ${basename}.grid ..."
    Run "$LLC" "${basename}.ll" ">" "${basename}.s"
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "bindings.o"
    Run "./${basename}.exe"

}
MODE="Help";
while getopts crh x; do
    case $x in
    c) # Compile
        MODE="Compile"
        ;;
    r) # Run
        MODE="Run"
        ;;
    h) # Help
        Usage
        ;;
    *) #Help if no opt given
        Usage
        ;;
    esac
done
#shift `expr $OPTIND - 1`
file=`echo $2`

case $MODE in
Compile)
    CompileFile $file
    ;;
Run)
    RunProgram $file
    ;;
Help)
    Usage
```

```
    ;;
esac
```