

AllHandsOnDeck

Final Project Report

Caitlyn Chen - *Language Guru*

Tiffeny Chen - *System Architect*

Jang Hun Choi - *System Architect*

Mara Dimofte - *Manager*

Christi Kim - *Tester*

{ckc2143, tc2963, jc5112, md3713, cwk2109}@columbia.edu

Contents

1	Introduction	3
2	Language Tutorial	4
2.1	Getting Started	4
2.2	Building AHOD	4
2.3	Running a Simple AHOD Program	5
3	Language Reference Manual	6
3.1	Tokens	6
3.2	Comments	6
3.3	Identifiers	6
3.4	Keywords	7
3.5	Data Types	7
3.6	Operators	7
3.7	Syntax Notation	8
3.8	Types	8
3.9	Params	8
3.10	Args	8
3.11	Program Structure	9
3.12	Declarations	9
3.12.1	Global Declarations	9
3.12.2	Function Declarations	9
3.13	Statements	11
3.13.1	Expression statement	11
3.13.2	Conditional statement	11
3.13.3	While statement	11
3.13.4	For statement	11
3.13.5	Return statement	12
3.13.6	Series Push	12
3.13.7	Statement Block	12
3.14	Expressions	12
3.14.1	Identifiers	13
3.14.2	Literals	13
3.14.3	More on Series Literal	13
3.14.4	Negation	14
3.14.5	Operations	14
3.14.6	Comparison	14
3.14.7	Assignment	14

3.14.8	Function Call	15
3.14.9	Class Call	15
3.14.10	Attribute Call	15
3.15	Context-Free Grammar	15
4	Standard Library	18
4.1	Built-in classes	18
4.2	Built-in functions	18
4.3	Runtime Exceptions	18
5	Project Plan	20
5.1	Development Process	20
5.1.1	Planning	20
5.1.2	Specification	20
5.1.3	Development	20
5.1.4	Testing	20
5.2	Style Guide	20
5.3	Project Timeline	21
5.4	Roles and Responsibilities	21
5.5	Software Development Environment	21
5.6	Project Log	22
6	Architectural Design	36
6.1	Scanner (scanner.mll)	36
6.2	Parser (parse.mly, ast.ml)	36
6.3	Semantic Checking (semant.ml, sast.ml)	36
6.4	Code Generation (codegen.ml)	36
7	Test Plan	37
7.1	Testing Suite and Automation	37
7.2	Example Test Programs	37
7.2.1	AHOD Code 1	37
7.2.2	AHOD Code 2	46
8	Lessons Learned	63
8.1	Caitlyn Chen	63
8.2	Tiffeny Chen	63
8.3	Jang Hun Choi	63
8.4	Mara Dimofte	64
8.5	Christi Kim	64
9	Appendix	65

Chapter 1

Introduction

Card games come in many different forms: games with the standard 52-card deck such as War or Blackjack, and games relying on unique decks such as Apples to Apples, UNO, SET, etc. We drew inspiration from past proposals, which shared similar motivations of building out languages aimed to support card game development. We found that there was a shortcoming in how past languages focused on supporting standard 52-card deck based games. And though existing card game languages might be able to represent standard 52-card games reasonably, they fail to generalize to the full breadth of card games out there. Not only does our language allow the user to create any turn-based card game, but it also supports general-purpose programming. The goal of our object-oriented, Python, Ruby, and C++-inspired language is to enable programmers to easily code the gameplay and functionality of a turn-based card game with an emphasis on code readability and modularity.

Chapter 2

Language Tutorial

2.1 Getting Started

AHOD requires OCaml and LLVM libraries to build and execute. Run the following commands to install the necessary packages.

Mac OS Distributions:

```
$ brew install opam
$ brew install llvm
$ opam install llvm
```

Ubuntu:

```
$ sudo apt-get install opam
$ sudo apt-get install llvm
$ opam install llvm
```

Other Linux distributions:

```
$ brew install llvm
$ opam depect conf-llvm.6.0.0
$ opam install llvm
$ export PATH=/usr/local/opt/llvm@6/bin:$PATH
```

2.2 Building AHOD

First, clone the AHOD repository with the following command:

```
$ git clone https://github.com/AllHandsOnDeck-PLT/AHOD.git
```

Next, run make inside the AHOD directory to create the AHOD compiler.

```
$ make
opam config exec -- \
ocamlbuild -use-ocamlfind AHOD.native
```

2.3 Running a Simple AHOD Program

Once the compiler is created, we can begin writing our own AHOD programs. Refer to chapter 3 of the Language Reference Manual to learn more about writing programs in AHOD.

Below is a simple `hello_world` program in AHOD:

```
1 main:
2 {
3   do PRINT("Hello world!")
4   do PRINT(":)")
5 }
6 /* prints hello world */
```

Save this as a `hello_world.ah` file. The following commands will help compile and execute the code:

```
$ ./AHOD.native helloworld.ah > helloworld.ll
$ llc -relocation-model=pic helloworld.ll > helloworld.s
$ cc -o helloworld.exe helloworld.s playercall.o
$ ./helloworld.exe
Hello world!
:)
```

Chapter 3

Language Reference Manual

Lexical Conventions

3.1 Tokens

There are six kinds of tokens: identifiers, keywords, comments, strings, expression operators, and other separators. AllHandsOnDeck employs curly braces (like C) and uses whitespaces as separators, similar to Python.

3.2 Comments

AHOD only supports single-line comments. For comments, the character `/*` is inserted at the beginning of the line and is terminated by either the character `*/` or newline. The compiler ignores all content between a `/*` and a `*/` or newline.

```
1 series<string> deck
2 series<string> hand
3 main:
4 {
5     /* This is a comment */
6     deck = ["d", "e", "f", "g"]
7     hand = ["a", "b", "c"] /* this is another comment */
8     hand.push(deck.pop()) /* deck.pop() gives 'e' */
9     hand.push(deck.pop()) /* deck.pop() gives 'f' */
10
11     /* hand = ["a", "b", "c", "f", "g"] */
12     /* deck = ["d", "e"] */
13 }
```

3.3 Identifiers

Identifiers in AllHandsOnDeck are sequences of letters and digits, and underscores `'_'`, where the first character must be a letter. Uppercase and lowercase letters are considered different. There are two kinds of

identifiers: ACTIONID and id.

ACTIONID identifiers denote functions in the AllHandsOnDeck language and may consist of uppercase letters, digits, and underscores only.

ACTIONID:

`('A'-'Z') ('A'-'Z' | '0'-'9' | '_')*`

Identifiers for variables and helper functions (non-state mutating functions) are denoted by id and may consist of lowercase letters, digits, and underscores only.

id:

`('a'-'z') ('a'-'z' | '0'-'9' | '_')*`

3.4 Keywords

The following are reserved keywords in AllHandsOnDeck:

int, float, bool, string, void, true, false, not, when, do, if, else, for, while, return, main, PRINT, Player, Card, series, push, pop, size

3.5 Data Types

Data Types	Description
int	integers are positive or negative whole numbers without decimal points
float	floats represent real numbers written with a decimal point
string	strings are sequences of characters that handle textual data
boolean	boolean variables are defined by the true and false keywords
series	series literals are dynamically sized arrays
Player	built-in class representing a player with name and score attributes
Card	built-in class representing a card with type, faceup, and value attributes

3.6 Operators

Operator	Description
<code>+, -, *, /</code>	arithmetic operators
<code>==, !=, <, >, <=, >=</code>	comparison operators
<code>and, or, not</code>	logical operators

Grammar

3.7 Syntax Notation

In the syntax notation used in this manual, syntactic categories are indicated by typewriter font, characters are indicated as the character itself in quotation marks, and the NEWLINE and EOF tokens are capitalized. The context free grammar is written in regex for the purpose of clarity, with the standard use of |Pipe, ?Question Mark, *Asterisk, +Plus, -Hyphen, and ()Parentheses.

3.8 Types

AllHandsOnDeck supports two fundamental types: primitive types and built-in class and data types.

```
type:
  prim_type | series | Player | Card
```

Primitive types include integer, floating-point, boolean, string, and void.

```
prim_type:
  int | float | bool | string | void
```

A series is declared with a type specification for member elements.

```
series:
  Series '<' type '>'
```

3.9 Params

params_list consists of parameters and are used in class constructors and function definitions.

```
params_list:
  param (',' param)*
```

params consist of variable and function identifiers and are type enforced.

```
param:
  type id
```

3.10 Args

args_list consists of arguments and is used in specifying instances of classes and function calls.

```
args_list:
  arg (',' arg)*
```

args consist of the expressions used to provide arguments for a class or function call.

```
arg:
  expr
```

3.11 Program Structure

Program is the top-level node in the syntax tree. Since we parse bottom-up, all parsing must end here.

```
program:
  (global_decl | action_decl)* main_decl EOF
```

A program is made up of a main function, and any number of global variable declarations and ACTION functions. All programs written in AllHandsOnDeck must contain a main function.

3.12 Declarations

There are three different types of declarations that can be made: global declarations, ACTION functions, and the main function.

3.12.1 Global Declarations

Global variable declarations occur at the top of a program file before any ACTION function declarations and main function.

```
global_decl:
  type id NEWLINE
```

3.12.2 Function Declarations

The main function runs the gameplay of the program and the ACTION functions are functions that can mutate the gamestate and have return values.

The main function takes the form of the keyword `main` and `:` followed by a `stmt_wrap`. The `locals_list` is the list of local variable declarations within the function declaration and must be ordered first before any other statements in the function body. A `stmt_wrap` is an optional block of statements.

```
main_decl:
  main ':' NEWLINE '{' NEWLINE locals_list stmt_wrap '}' NEWLINE
```

```
locals_list:
  (type id NEWLINE)*
```

```
stmt_wrap:
  (stmt+)?
```

The main function is intended to be a high-level, readable representation of what the gameplay entails for any game programmed using AllHandsOnDeck.

Sample main function:

```

1  series<string> deck
2  when do void INIT():
3  {
4      deck = ["R0","R1","R2"]
5  }
6  main:
7  {
8      do PRINT("game setup")
9      do INIT()
10 }

```

ACTIONS are declared with the when...do ACTION structure and : followed by a stmt_wrap. Programmers can specify any params the ACTION should take in, with type specifications.

action_decl:

```

when do type ACTIONID '(' params_list? ')' ':' NEWLINE '{' NEWLINE locals_list
stmt_wrap '}' NEWLINE

```

An example of an ACTION function declaration can be seen in the following initialization of a deck of cards for a game.

```

1  Card card1
2  Card card2
3  Card card3
4  series<Card> deck
5  Card type1
6  int i
7  when do void CREATEDECK():
8  {
9      string red5
10     red5 = "R5"
11     card1 = Card(red5, true, 5)
12     card2 = Card("R6", false, 6)
13     card3 = Card("R7", true, 7)
14     deck = [card1, card2, card3]
15     for (i = 0; i < deck.size(); i = i + 1):
16     {
17         type1 = deck[i]
18         do PRINT(type1.type)
19         do PRINT(type1.faceup)
20         do PRINT(type1.value)
21     }
22 }
23 main:
24 {
25     do CREATEDECK()
26 }

```

3.13 Statements

Statements, unless noted otherwise, are executed in sequence.

```
stmt:
    expr NEWLINE | if_stmt | for_stmt | while_stmt | return_stmt | series_push | stmt_block
```

3.13.1 Expression statement

Most statements are expression statements, usually assignments or function calls, and take the form of an expression followed by a NEWLINE token.

```
expr NEWLINE
```

3.13.2 Conditional statement

The code within an `if...else if...else` block will be executed if the result of the test expression in the `if` statement evaluates to `True`. If the test expression is `False`, the `stmt_block` will not be executed. AllHandsOnDeck interprets `true` as 1, and `false` as 0.

```
if_stmt:
    if expr ':' stmt_block else_block?
```

```
else_block:
    else ':' stmt_block
```

3.13.3 While statement

The code within a `while` block will be executed repeatedly as long as the evaluation of the test expression in the `while` statement evaluates to `True`.

```
while_stmt:
    while expr ':' stmt_block
```

3.13.4 For statement

A `for` loop is used to repeat a specific statement block a known number of times by initializing a counter and incrementing the counter until a condition is true. `For` loops can be used to execute a set of statements, once for each item in a given sequence.

```
for_stmt:
    for '(' expr ';' expr ';' expr ')' ':' stmt_block
```

```
1 int i
2 for (i = 0 ; i < 10; i = i + 1):
3 {
4     do PRINT(i)
5 }
```

For loops can be nested:

```
1  int i
2  int j
3  for (i = 0 ; i < 5; i = i + 1):
4  {
5      for (j = 0 ; j < 5; j = j + 1):
6      {
7          do PRINT(i)
8          do PRINT(j)
9      }
10 }
```

3.13.5 Return statement

ACTIONS and helper functions return to their callers by means of the `return` statement, which either returns no value or returns the value of the specified expression to the caller of the function.

```
return_stmt:
    return expr? NEWLINE
```

3.13.6 Series Push

`SeriesPush` is the function used to add an element at the back index of a `Series` object.

```
Series_push:
    id '.' push '('')' NEWLINE
```

3.13.7 Statement Block

Statement Blocks consist of curly braces used to demarcate an ensuing list of statements that will be executed. Unlike other languages that have delimiter like semicolons. Our language uses new lines to indicate the end of a statement.

```
stmt_block:
    NEWLINE ''' NEWLINE stmt+ ''' NEWLINE
```

3.14 Expressions

Expressions are sequences of operands and operators and are meant to be evaluated.

```
expr:
    | id | neg_expr | iliteral | fliteral | sliteral | bliteral | Series_literal
    | Series_get | Series_size | Series_pop | binary_op | unary_op | logical_op
    | comparison | assignment | call_print | call_action | call_class | call_attr
```

3.14.1 Identifiers

Identifiers denote names of variables, functions, and classes in AllHandsOnDeck. Refer to section 2.3 of Chapter 2 for more details.

3.14.2 Literals

There are five kinds of literals in AllHandsOnDeck: integer literals, floating-point literals, string literals, boolean literals, and Series literals.

Integer and floating-point literals are immutable.

```
iliteral:  
  ('0'-'9')*
```

```
fliteral:  
  ('0'-'9')* '.' ('0'-'9')* (('e' | 'E')('+ | '-'))? ('0'-'9')*
```

String literals are sequences of characters surrounded by single quotes or double quotes.

```
sliteral:  
  '"' (' '-'!' ' #'-'&' ' ('-'[' ' ]'-' ' ' 'a'-'z' ' ' 'A'-'Z' '0'-'9')*
```

A boolean literal can have either the `true` or `false` value.

```
bliteral:  
  true | false
```

3.14.3 More on Series Literal

A `Series` literal is a representation of a `Series` in AllHandsOnDeck, a dynamic array that supports `size`, `get`, `push`, and `pop` methods.

```
Series_literal:  
  '[' (expr (',' expr)*)? '']'
```

Indexing can be done following the structure `id[expr]`.

```
Series_get:  
  id '[' expr '']'
```

To get the size of a series, one follows the structure `id.size()`.

```
Series_size:  
  id '.' size '(')'
```

To use the `pop` method series, one follows the structure `id.pop()`.

```
Series_pop:
```

```
id '.' pop '()'
```

3.14.4 Negation

The `not` keyword is a logical operator and the return value will be 1 if the statements are not true, and will be 0 otherwise.

```
neg_expr:  
  not expr
```

3.14.5 Operations

The following binary operations are supported by AllHandsOnDeck:

```
binary_op:  
  expr ("+" | "-" | "*" | "/" ) expr
```

The following unary operations are supported by AllHandsOnDeck:

```
unary_op:  
  ("-" | "not" ) expr
```

The following logical operations are supported by AllHandsOnDeck:

```
logical_op:  
  expr ("and" | "or" ) expr
```

3.14.6 Comparison

Comparisons yield 1 or 0 and can be chained arbitrarily. All comparison operators have the same priority, which is lower than that of any arithmetic operation.

```
comparison:  
  expr ("==" | "!=" | "<" | "<=" | ">" | ">=" ) expr
```

3.14.7 Assignment

An assignment expression assigns an expression to an identifier, while also returning the value of the expression.

```
assignment:  
  id "=" expr
```


3.14.8 Function Call

Calls to ACTIONS have the following syntax in AllHandsOnDeck.

ACTION functions are called following the structure of do ACTION.

```
call_action:
  expr? do ACTION ('(' args_list ')')?
```

3.14.9 Class Call

Calls can be made to AHOD's built-in classes Player and Card and take the structure of the class name followed by the appropriate arguments to the class's constructor.

```
call_class:
  Card '(' args_list? ')' | Player '(' args_list? ')'
```

3.14.10 Attribute Call

Calls can be made to access specific attributes of object instantiations of AHOD's built-in classes Player and Card.

```
call_attr:
  id '.' id
```

3.15 Context-Free Grammar

```
program:
  (global_decl | action_decl)* main_decl EOF

global_decl:
  type id NEWLINE

action_decl:
  when do type ACTIONID '(' params_list? ')' ':' NEWLINE '{' NEWLINE locals_list
  stmt_wrap '}' NEWLINE

main_decl:
  main ':' NEWLINE '{' NEWLINE locals_list stmt_wrap '}' NEWLINE

locals_list:
  (type id NEWLINE)*

stmt_wrap:
  (stmt+)?
```

```

type:
  prim_type | series | Player | Card

prim_type:
  int | float | bool | string | void

series:
  Series '<' type '>'

params_list:
  param (',' param)*

args_list:
  arg (',' arg)*

stmt:
  expr NEWLINE | if_stmt | for_stmt | while_stmt | return_stmt | series_push | stmt_block

if_stmt:
  if expr ':' stmt_block else_block?

else_block:
  else ':' stmt_block

while_stmt:
  while expr ':' stmt_block

for_stmt:
  for '(' expr ';' expr ';' expr ')' ':' stmt_block

return_stmt:
  return expr? NEWLINE

Series_push:
  id '.' push '('')' NEWLINE

stmt_block:
  NEWLINE '' NEWLINE stmt+ '' NEWLINE

expr:
  | id | neg_expr | iliteral | fliteral | sliteral | bliteral | Series_literal
  | Series_get | Series_size | Series_pop | binary_op | unary_op | logical_op
  | comparison | assignment | call_print | call_action | call_class | call_attr

iliteral:
  ('0'-'9')*

```

```

fliteral:
    ('0'-'9')* '.' ('0'-'9')* (('e' | 'E')('+ | '-' )? ('0'-'9')*)?

sliteral:
    '"' (' '-'!' ' #-'&' ' ('-'[' ' ]'-' ' ' a'-'z' ' ' A'-'Z' '0'-'9')*

bliteral:
    true | false

Series_literal:
    '[' (expr (',' expr)*)? ']'

Series_get:
    id '[' expr ']'

Series_size:
    id '.' size '(' ')'

Series_pop:
    id '.' pop '(' ')'

neg_expr:
    not expr

binary_op:
    expr ("+" | "-" | "*" | "/" ) expr

unary_op:
    ("-" | "not" ) expr

logical_op:
    expr ("and" | "or" ) expr

comparison:
    expr ("==" | "!=" | "<" | "<=" | ">" | ">=" ) expr

assignment:
    id "=" expr

call_action:
    expr? do ACTION ('(' args_list ')')?

call_class:
    Card '(' args_list? ') ' | Player '(' args_list? ') '

call_attr:
    id '.' id

```

Chapter 4

Standard Library

4.1 Built-in classes

Player and Card entities are predefined classes in AllHandsOnDeck that a programmer can use in coding up card games.

A Player class has string name and int score attributes and a Card class has string type, bool faceup, and int value attributes.

A Series can be thought of as a list data structure where the front element is index 0 and the back element is index -1 and can be used to represent a player's hand. The built-in methods for a Series include indexing and helper functions like `size()`, `push()`, and `pop()`.

4.2 Built-in functions

- `print()` prints the specified object to the screen after first converting it to a string
- `<series>.size()` returns the number of elements in the series
- `<series>.pop()` pops the back element of a Series and returns the element
- `<series>.push(element)`: pushes 1 element to the back of a Series

4.3 Runtime Exceptions

Runtime exceptions occur on types that are not inferrable. AHOD checks for invalid assigns, argument types, initialization, and list bounds. For these exceptions, we simply throw a "Fatal error: exception Failure(...)".

For example, the following program returns a "Fatal error: exception Failure("undeclared identifier j")" error:

```
1  when do void PLAY():
2  {
3      int i
4      for (i = 0; i < 10 ; i = j + 1): /*j not defined*/
5      {
6      do PRINT("uh oh failure")
7      }
8  }
9  main:
10 {
11     do PLAY()
12 }
```

Chapter 5

Project Plan

5.1 Development Process

5.1.1 Planning

In order to complete AHOD in a timely fashion, we had biweekly meetings scheduled on Google Meets. During our meetings we discussed action items, shared development concerns and assigned tasks. One of the meetings was on Tuesday, and the other one on Friday, when we would continue our discussion after the weekly meeting with our TA, Xijiao Li. We scheduled any additional meetings, based on demand.

We had a master database created on Notion that contained any relevant notes, meetings, code, links, and external links to google drive. We used the Notion page to organize our project and keep a consolidated list of deadlines, task assignments and action items.

Our day to day communication happened on Facebook Messenger.

5.1.2 Specification

The goal of AHOD was to be a python-like card language that allows users to easily implement card based games that require both traditional and non-traditional type of cards.

5.1.3 Development

Our language development was based on the agile methodology. We prioritized understanding our development requirements, setting small goals to be achieved in sprints, and incrementally building the compiler functionality in a bottom up fashion. We scheduled multiple sprints based on the timeline of the material learned in lecture to be able to accomplish the scope of our project in a timely fashion.

5.1.4 Testing

We tested features for both failure and success before merging all implemented features together. Our test suite contains all tests used during development.

5.2 Style Guide

The style guide followed by AHOD:

- Variables formatted with snake case

- Parser types upper Caml case
- Parser braces properly aligned on a vertical axis
- Match case vertical bars aligned vertically
- 2-character indentations
- 80 characters per line

5.3 Project Timeline

Date	Milestone
Jan 15	Brainstorming
Jan 22	Individual Language Proposals
Jan 29	Proposal First Draft
Feb 3	Proposal
Feb 14	First Parser Attempt
Feb 20	Test Suite First Attempt
Feb 21 - Feb 23	Parser Sprint
Feb 24	LRM, Parser
Mar 14	MVP Hello World
Mar 24	Hello World
Apr 10	Binary Operators
Apr 12	Expressions
Apr 12	Statements
Apr 14	Control Flow
Apr 20	String Literals
Apr 22	Action Declarations
Apr 23	Objects
Apr 23 - Apr 26	Master Sprint

5.4 Roles and Responsibilities

Team Member	Role
Caitlyn Chen	Language Guru
Tiffeny Chen	System Architect
Jang Hun Choi	System Architect
Mara Dimofte	Manager
Christi Kim	Tester

5.5 Software Development Environment

We used the following programming and development environments:

- **Libraries and Languages:** OCaml Version 4.12.0 including Ocamllyacc and OCamllex and LLVM Version 11.1.0

- **Software:** VSCode, Sublime Text, Vim
- **OS:** OSX 11.0.1, Windows 10
- **Version Control:** Git Repository hosted on Github

5.6 Project Log

commit 9d0d913d59d1fb5a7d873c1e7b876ecdf9239f29 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>

Date: Mon Apr 26 21:51:23 2021 -0400

cleaned up parser

commit 6b2ed9fd88addc83c218f707269d0b046153bf04 Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 21:42:53 2021 -0400

not needed files

commit acf1d1c49c18e40c58725dac14c1f180efb05ec9 Merge: 0fa2e7b 69a7239 Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 21:35:44 2021 -0400

merge

commit 0fa2e7b86bfab0c8740d05fecddcce3a864b214b Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 21:34:57 2021 -0400

added fail cases

commit 69a7239a268596300aa745513f274a58fe6f0393 Merge: 6c51f3d ef1fd99 Author: Caitlyn Chen

<caitlyn.chen8@gmail.com> Date: Mon Apr 26 20:11:43 2021 -0400

grammar fixes

commit 6c51f3d00a1ebd7588bf117b84249278e6f23830 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>

Date: Mon Apr 26 20:10:12 2021 -0400

cleaned up grammar

commit ef1fd99d34044a6b6c833454726116419c3a7863 Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 20:07:19 2021 -0400

added more tests

commit 0b8af009cbc50ae1ca022b599826928bbe18a03b Merge: d2b657c 70eb46f Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 19:56:44 2021 -0400

resolved merge conflicts

commit d2b657c47522092d61c4acd6972aa0c3ed48b6cf Author: junebug <47231340+junebug111@users.noreply.github.com>

Date: Mon Apr 26 19:52:57 2021 -0400

added more test/fail cases

commit 70eb46f1b712348b6b9bb5da610afb8ecda9a045 Author: Mara Dimofte <mara.dimofte123@gmail.com>

Date: Mon Apr 26 19:48:52 2021 -0400

added tests for classes

commit f5a47d2b66973500f38445609e7769d326682037 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>

Date: Mon Apr 26 19:05:18 2021 -0400

series5 testcase for final report

commit 2408cd07718f1ee708cb3ed3f07425f6cbd8dfcb Author: tiffenychen <tc2963@columbia.edu>

Date: Mon Apr 26 18:14:37 2021 -0400

added test demo

commit ba95daa15393c7b038d90ef4a276a589a8077546 Merge: d708d00 0c69218 Author: Caitlyn Chen

<caitlyn.chen8@gmail.com> Date: Mon Apr 26 17:07:50 2021 -0400

Merge branch 'true_master' of <https://github.com/AllHandsOnDeck-PLT/AHODintotrue_master>

commit d708d0099e71de572df50082a3d67095f773afd5 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>

Date: Mon Apr 26 17:07:44 2021 -0400

semant fix and fail class tests

commit 0c6921884bb83182aa3227bb6c6bbd9e18c69b13 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 16:51:34 2021 -0400
 cleaned up code

commit 93116a3ccf6168e731e2f8de18b648aa13b824cc Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 16:35:22 2021 -0400
 demo uno

commit d58fa06f53a82f0b9ccd35065c8661135eb3432a Merge: a6361eb 316b60e Author: Caitlyn Chen
 <caitlyn.chen8@gmail.com> Date: Mon Apr 26 16:33:34 2021 -0400
 merge

commit a6361ebddc8f231c01d047eba79db782119692c9 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 16:26:40 2021 -0400
 demo2

commit 316b60ee0a4c85ad0017ce653d02c5522679d958 Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 16:23:46 2021 -0400
 Added int value to card type and bjsim demo

commit 320f7ed1c30000b54de7a0cf0adcd74a702c5294 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 15:17:23 2021 -0400
 added demo and presentation tests

commit 4c3681e33c3a4eb03167b5e9bc83de1942fea8ce Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 14:39:54 2021 -0400
 changed testclass4

commit 7bef248027dde61612ae58baa7ce6d24bf6310ae Merge: f2252d1 0fd8396 Author: tiffenychen
 <tc2963@columbia.edu> Date: Mon Apr 26 14:24:17 2021 -0400
 resolving merge isuses, comment out attrassign

commit f2252d101437a9287d0041dfa6df557dc16c552c Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 14:18:45 2021 -0400
 commented out attrassign, warnings resolved renamed class5 to demo

commit 0fd839637a63925a4707ede66e4f3e7bb8dc0364 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 13:59:20 2021 -0400
 fixed void instruction error

commit bed2376af0f9e6be638eeb0165b4202dda77a3e7 Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 13:57:06 2021 -0400
 Added to test suite

commit ad754e5765d42459c99a83867c2d5cb92e91e73d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 12:47:19 2021 -0400
 fixed compilation error for AttrAssign but still not working properly

commit eecc4101a156d1912a203f2448260b3d40112e62 Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 06:01:41 2021 -0400
 Fixed overshadow error by changing None to Void

commit b492b67e35f6d766b6ee4f557f9cf276d7960117 Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 05:46:33 2021 -0400
 pre-classes and none warnings fixed

commit 4c75c226677cf4d8e6eab10a892d385becdf975 Merge: 4fb1690 9956ebc Author: tiffenychen
 <tc2963@columbia.edu> Date: Mon Apr 26 05:24:49 2021 -0400
 Merged Merge remote-tracking branch 'refs/remotes/origin/true_master' into true_master

commit 4fb16901ec5b87b0a4ff1bdbbec6b310a978f80d Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 05:24:44 2021 -0400
 added tests for actions and handled warnings aside from None

commit 9956ebc8fbb39d90c80d2a3f43d43ed06a8d7a05 Merge: 53236e1 1dbaac7 Author: Caitlyn Chen
 <caitlyn.chen8@gmail.com> Date: Mon Apr 26 03:53:12 2021 -0400

merged
 commit 53236e183b5b053a314379770b273aa2eba4436b Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 03:50:03 2021 -0400
 series card support attempt at AttrAssign
 commit 1dbaac77ed14f6e8c7878d797333e21b539ff2a3 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 03:23:28 2021 -0400
 added more test cases for stmt
 commit fae90eff997fddb95aacaf494d87937754ee4d1c Merge: 3d3ccb2 9ce9a03 Author: janggg <janghc39@gmail.com>
 Date: Mon Apr 26 02:46:30 2021 -0400
 "adding fail tests for for" Merge branch 'fail-cases' into true_{master}
 commit 9ce9a03c6a59a1266e612ade3d450436e357134f Author: janggg <janghc39@gmail.com> Date:
 Mon Apr 26 02:41:42 2021 -0400
 fail tests for for
 commit 3d3ccb249477a05e54a6a9c1bd549264fad8373e Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 02:35:18 2021 -0400
 cleaning up warnings
 commit 407d5e6bef1c736b0cb3dc3ed29339f1a25a531c Merge: 18387d5 d3ab559 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 02:20:30 2021 -0400
 added comments
 commit 18387d54c654669ec28e2ff861ad84d1a269d54e Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 02:13:13 2021 -0400
 added working comments
 commit d3ab5591d5c3a1f464fa5c014eee507fd6a8d9f9 Merge: e74eb10 908b11e Author: tiffenychen
 <tc2963@columbia.edu> Date: Mon Apr 26 02:07:06 2021 -0400
 Merged Merge remote-tracking branch 'refs/remotes/origin/true_{master}' into true_{master}
 commit e74eb10ebc4567e5ad66d46b3c2254d59de51808 Author: tiffenychen <tc2963@columbia.edu>
 Date: Mon Apr 26 02:06:41 2021 -0400
 Working if/else in stmt returns and tests for return and action params
 commit 908b11e851e402f26e892a805ff094812b6d10f7 Merge: 9be31d3 45e7a7f Author: Caitlyn Chen
 <caitlyn.chen8@gmail.com> Date: Mon Apr 26 01:58:06 2021 -0400
 merged classes_{objects_{new}}
 commit 45e7a7f925a85689ad0ccf1a92163c72154c30e1 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 01:00:42 2021 -0400
 class2 test passing
 commit 9be31d3ce0cc5bbae5301e07c65d76977519464c Merge: e0c877f 3e72d7b Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Mon Apr 26 00:36:46 2021 -0400
 merged master_{andlocal/actionstuff}
 commit 94b041a6fa0944d91a34e0b11943feccb0475107 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Mon Apr 26 00:26:27 2021 -0400
 built-in class support for Card class
 commit e0c877fd7c6d487d3de277d2ed020cbfa0bfc5ce Merge: c5f470d b080b47 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 25 23:51:23 2021 -0400
 merged to have print string
 commit 61554db09bba22e638451705725270ccf218ae4c Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 23:48:09 2021 -0400
 access attribute working
 commit c5f470d3c1c68f6ad04026680c67368a46e3ede8 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 25 23:20:26 2021 -0400
 working locals, kind of working action stuff

commit 868cf0f846a12bfcc09015529e9623aaef12e113 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 22:28:33 2021 -0400
 fixed seg fault, incorrect args in build, *all forget attribute function calls*

commit a86102b00bf73d8a1359c082f06013af13fb674d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 19:12:13 2021 -0400
 changes for attribute access

commit b080b474f24e3d7afec0aa2e57d0137384049e34 Author: tiffenychen <tc2963@columbia.edu>
 Date: Sun Apr 25 18:56:39 2021 -0400
 Added in func param into stmt does not have terminator error

commit b53107cd08e805d030d31c1556fb346309f6e06c Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 18:52:15 2021 -0400
 added get attribute functions

commit c347ee26f0c748bd613c75668d01c7274437228c Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 18:51:25 2021 -0400
 added accessing attribute stuff

commit 88a75c7239a7e23d93b7c44575dd66971b330c63 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 17:47:25 2021 -0400
 added grammar for attrcall

commit f9883f5afe7491d81f85ec33cd62c0d3dde9f17d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 17:15:15 2021 -0400
 fixed playercall

commit 25ee3546b894428b540fd53536d6a8fe1f50ebbd Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 17:05:43 2021 -0400
 playercall function linked correctly

commit 20380808e70c6906869bc46c9f3221305390cdd6 Merge: e4a0012 6df957e Author: Caitlyn Chen
 <caitlyn.chen8@gmail.com> Date: Sun Apr 25 16:19:17 2021 -0400
 merge

commit 2897375497c783536ba626495cd8458851dcebec Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 25 16:15:51 2021 -0400
 made vars opt

commit 121449d8fefdbed4770539affae55c75d2279561 Author: tiffenychen <tc2963@columbia.edu>
 Date: Sun Apr 25 16:12:12 2021 -0400
 Print for multiple types supported by adding call, *rintexpression*

commit e4a001264cb123971a57acbe3301c370bdacbcac Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 15:59:02 2021 -0400
 modified test

commit 18254f39d992ddc226564710a4947f2e6fba787d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Sun Apr 25 15:58:09 2021 -0400
 ("printb", Bool); ("printf", Float);

commit 3556ad3b12bb2524b7f8d8a1fe2229c5272da4cd Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 25 15:01:04 2021 -0400
 working local, but terrible org of code

commit 25137f68102e83953b29008683e84566e9f2a2d8 Author: tiffenychen <tc2963@columbia.edu>
 Date: Sun Apr 25 15:00:53 2021 -0400
 commented out stuff

commit 2432cd6283c518c2cfb5a9f1b0f06fb988555c6d Author: tiffenychen <tc2963@columbia.edu>
 Date: Sun Apr 25 14:15:58 2021 -0400
 experimented with treating as a tuple

commit 6df957e0cf179f3e99656fe028115cbf753bf2bf Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Sun Apr 25 13:55:32 2021 -0400

fixed mismatch
commit 2932436956640cab2890f49fb779fe42fab42f84 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Sun Apr 25 10:55:37 2021 -0400

progress in fixing type mismatch
commit 9e8241d60aeeac549dfd9cbf5392da0239dbbf71 Author: tiffenychen <tc2963@columbia.edu>
Date: Sun Apr 25 02:57:50 2021 -0400

errors with tuple arg for stmt
commit 09c64d938472f5baf2bf439f4e0f9b5b1759a44e Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sun Apr 25 01:58:40 2021 -0400

not working local due to scope issues
commit e767a054a66c046346371afbc49e332f965930a0 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sun Apr 25 00:43:05 2021 -0400

fixed class stuff
commit b522f2a928da46e892e01ec5a0f3d5816fd46071 Author: tiffenychen <tc2963@columbia.edu>
Date: Sat Apr 24 19:19:18 2021 -0400

commented out main *funcinstatementspassingactioncasemovedbuilderorder*
commit a8fb65f13436f6840005868dec6fcab8316542c3 Merge: cf288bf9e968e7 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sat Apr 24 18:24:42 2021 -0400

merge conflict resolved
commit eab1e0f44bf0950cfaa545b99bca95c5ae6bd975 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Sat Apr 24 18:23:27 2021 -0400

commented out class expr
commit cf288bf32ffb2423a248b24f9eb9785efa5898b0 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sat Apr 24 18:23:05 2021 -0400

syntax err resolved, trying to get main working
commit 2fa379ceae048e898b093cdd5585b39ee51271f2 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Sat Apr 24 18:11:22 2021 -0400

compiling version
commit 9e968e70cd67c13f9f3f007334f87c323ffc5630 Author: tiffenychen <tc2963@columbia.edu>
Date: Sat Apr 24 15:03:05 2021 -0400

added in let/in error with the in
commit 3e72d7b855478d0e150343b884e1236ad8716e41 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sat Apr 24 14:33:09 2021 -0400

refining lists
commit 08e7e24a45d9bf9d9bc8e56354eb21ef8e7d1023 Author: tiffenychen <tc2963@columbia.edu>
Date: Sat Apr 24 14:30:58 2021 -0400

changed mbuilder to builder
commit bf1986667dec7b4878a61a28ea642ec57866296e Author: tiffenychen <tc2963@columbia.edu>
Date: Sat Apr 24 14:12:36 2021 -0400

error on let and made more readable
commit 322fb3fa5eca8ad8b586dbc3df8ea0ce2a416de8 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sat Apr 24 13:30:21 2021 -0400

added unop
commit fd1002c0cd4934e1df462e691fe82ffd0b266ac6 Author: tiffenychen <tc2963@columbia.edu>
Date: Sat Apr 24 03:08:47 2021 -0400

working on builder for main in CodeGen
commit b855ae25ece2bf856adb09947b264df9ce3fae2e Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sat Apr 24 01:25:54 2021 -0400

comments working

commit 0f22d58dc56a2034e512e48f0ec233ae848f361e Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Apr 23 23:30:17 2021 -0400
working on action decl
commit d8077cb607e41c4189f6630e90b6b32c6d1feab8 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Apr 23 22:14:33 2021 -0400
trying to do comment
commit a88e793235969fafa7340d24e18925729d81f4de Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Apr 23 22:06:32 2021 -0400
trying to fix comment
commit 5aa6d53083ab139862849b9909fa93dcd371153d Author: janggg <janghc39@gmail.com> Date:
Fri Apr 23 22:00:24 2021 -0400
parser errors for *stmt_iist/block*
commit f9254b3c98727b5840514a2671293d9b9cb696b3 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Apr 23 18:31:09 2021 -0400
working list, more tests
commit 29f380ae502fed2ced74c38f6e8736e8ad96c6d5 Author: janggg <janghc39@gmail.com> Date:
Fri Apr 23 16:25:21 2021 -0400
broken version w/ attempt for class implementation
commit 99f6e554a356077aa4625d17bddc0b7b233e889b Author: janggg <janghc39@gmail.com> Date:
Fri Apr 23 15:56:36 2021 -0400
moved action_decl_{abovecheck_expr}
commit ede907249f974e36bfa56cc6e3325a3b36e32146 Author: tiffenychen <tc2963@columbia.edu>
Date: Fri Apr 23 14:05:26 2021 -0400
added in create action_decl_mapinsemantunboundfind_acterror
commit 0090ce6fd2046c58ca10d8528caa91185ab81a5d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Apr 23 13:34:20 2021 -0400
fixed error after merging
commit c4a40a33432cbffe8aa06a0c3e308fd99433cb3c Merge: b3e7476 2d89f3f Author: Caitlyn Chen
<caitlyn.chen8@gmail.com> Date: Fri Apr 23 13:16:23 2021 -0400
Merge branch 'stmt_block_{parse}' into action_decl_new
commit 2d89f3f3b06b5c25c09bd011c42c53a8442e1ba6 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Apr 23 13:15:08 2021 -0400
fixed s/r conflicts
commit 2ddb490dc860409adf17e66659d937a63ae2f2f Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Apr 23 13:14:06 2021 -0400
attempted series size, parsing error
commit b3e747690f01cfaf0d3a00c35783e63a2eff3f10 Author: tiffenychen <tc2963@columbia.edu> Date:
Fri Apr 23 13:13:00 2021 -0400
fixed warnings
commit 8b677034ec95206c34ea99b27dbff5c41cfdedde Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Apr 23 12:54:05 2021 -0400
rewrote stmt branch but some s/r conflicts
commit a9819fad0c06ee0ff1ba9e632d2dfe2fcf9ed127 Merge: daadd3f 4497ac9 Author: tiffenychen
<tc2963@columbia.edu> Date: Fri Apr 23 01:41:11 2021 -0400
Merged Merge branch 'action_decl' into action_decl_new
commit 4497ac906847e76fcc906bf8359dd69b156d2bcf Merge: dbc1c90 834304d Author: tiffenychen
<tc2963@columbia.edu> Date: Fri Apr 23 01:39:22 2021 -0400
Merged
commit daadd3f77e9cd80d1dfcbfd100951527dab0b067 Merge: 4ded4ac 834304d Author: Mara Di-
mofte <mara.dimofte123@gmail.com> Date: Fri Apr 23 01:17:13 2021 -0400

Merge branch 'master'₂ of <https://github.com/AllHandsOnDeck-PLT/AHODintomaster₂>
commit dbc1c90c0a1c6d4213c73c0d58e3ceae8b8078cb Author: tiffenychen <tc2963@columbia.edu>
Date: Fri Apr 23 01:10:38 2021 -0400
removed when do
commit 4ded4ac62d58822bebaae4ddc7b58b70dc7cb50f Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Fri Apr 23 01:06:06 2021 -0400
add
commit f1da513bdc3db7842ec4808ecb2b37308e04873b Author: tiffenychen <tc2963@columbia.edu>
Date: Fri Apr 23 00:50:10 2021 -0400
Handled semant incompatible type errors in check_actionbody
commit 4e037864f1b4c9fbc0fd62db031beffd3e45a60 Author: tiffenychen <tc2963@columbia.edu>
Date: Thu Apr 22 23:23:47 2021 -0400
unbound error gone now type mismatch
commit 834304d3731fba7e35aa2aca8534aa7bc04acc23 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Thu Apr 22 22:42:35 2021 -0400
removed extra files
commit 231d3c70af783d1b0c04c1281acfb69b7ef29a20 Merge: f258a6a 979bb94 Author: junebug <47231340+junebug111@u>
Date: Thu Apr 22 22:18:44 2021 -0400
Merge remote-tracking branch 'origin/series;*nProgress*' into master₂
commit 979bb9458b5c74e58acc6e3a97a15eddb96ae97c Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Thu Apr 22 22:13:17 2021 -0400
additional files
commit f258a6a2105f2fd644aed271852c657e50af4a25 Merge: f2939a3 2b93d44 Author: Mara Di-
moft <mara.dimofte123@gmail.com> Date: Thu Apr 22 22:07:09 2021 -0400
merge stuff
commit f2939a3ec34d3d531717979241ce328e4dfc1a2 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Thu Apr 22 21:58:06 2021 -0400
for loops working
commit 2f44454b1cf217cde87d4ff793d4e1a97ea8f822 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Thu Apr 22 21:53:28 2021 -0400
basic series working
commit d677260f372ccc35eaca1452f8084d83ed0ecf0d Author: tiffenychen <tc2963@columbia.edu>
Date: Thu Apr 22 21:35:50 2021 -0400
unbound check_stmt
commit 780ed93ce88974ab42b44232a8895a9666fd5ee4 Author: tiffenychen <tc2963@columbia.edu>
Date: Thu Apr 22 19:35:09 2021 -0400
working test
commit 9a6ab83c0ee0379e5f6ba7353c642eeb52421416 Author: tiffenychen <tc2963@columbia.edu>
Date: Thu Apr 22 19:24:03 2021 -0400
minor action additions in codegen and grammar
commit 63c94d66460c521bb723610ce91d6a8e9c2e2128 Author: tiffenychen <tc2963@columbia.edu>
Date: Thu Apr 22 13:04:09 2021 -0400
Notes from Professor Edwards OH
commit 2b93d4421584cb1fb8b4bb34ef5f7a2a41922c2d Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Mon Apr 12 00:58:02 2021 -0400
type conflict
commit b55537a8d9519bbd292cc7c2cb5619a89b397e91 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Sun Apr 11 21:19:20 2021 -0400
parse

commit 79c59e269f74ab1c6f7a7ea5d892ef2ea3a95960 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 11 21:14:39 2021 -0400
 resolved merge conflicts
 commit 96f74dc2758a33d4a73c072fdaa28dd9fc94e6f2 Merge: eed14df2f6cd92 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 11 21:12:29 2021 -0400
 merge assign and series
 commit 2f6cd92e3c3c699cc73c006c16d1f2c9ff4c33ff Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Sun Apr 11 21:05:56 2021 -0400
 working on series
 commit eed14df10b746cb4010991baa33124cff1327f9d Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Fri Apr 9 16:31:12 2021 -0400
 globals assign working, if, else, while working
 commit c993cd43d0cdf5d30d7970de9b7eff0a2b75f554 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Apr 9 14:51:02 2021 -0400
 fixed reduce/reduce conflicts
 commit c7b9c6aed7f9dc5794e03b414f9dd6f4bc7aeae Author: tiffenychen <tc2963@columbia.edu>
 Date: Fri Apr 9 14:37:11 2021 -0400
 start on class structs
 commit fbee6e4fe56e936d0efdd4f76164ee3ad945f6bc Merge: 9ee0308 d5eea7e Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Apr 9 13:46:43 2021 -0400
 merged but reduce/reduce conflicts
 commit d5eea7e4a812cef166db84bd3667b4f9cc6fedd6 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Apr 9 13:23:07 2021 -0400
 globals assign working
 commit 9ee03081f811c8857edb085817442e068744bfdf Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 8 23:15:52 2021 -0400
 simple class test case added punctuation spacing to scanner
 commit 771165701fb2b40a7751ddf743fa775e69fce74 Author: tiffenychen <tc2963@columbia.edu>
 Date: Thu Apr 8 23:00:03 2021 -0400
 clang class in c and ll
 commit f23703c9a0abadeb955f4c8e1f609bd3e6a8bb54 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 8 18:33:23 2021 -0400
 grammar stuff good for class
 commit ccfab2f4c9c850f5592c20b176ec0585493ce6b0 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Apr 7 20:30:08 2021 -0400
 fixed parse error
 commit b68c60554d326c65e747ded8a16b9db280a9317e Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Wed Apr 7 19:57:31 2021 -0400
 error in parser after adding globals
 commit 30983104b9ee6dabc19b3f74227ac3e3453d981d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Apr 7 13:38:33 2021 -0400
 fix
 commit 830571802ac4ddda9384892e81dc4afab57df50d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Apr 7 13:35:38 2021 -0400
 working globals grammar
 commit 70755c7a15d67a0ff00929e5b567ad33e1f25356 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Apr 7 13:25:33 2021 -0400
 globals grammar
 commit 2e827c2b2763bbd31e3805596a2d4adfcc93f145 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Apr 7 12:44:06 2021 -0400

grammar stuff
 commit 806f6463cdfd386b6397533227924f1b029ed887 Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Wed Apr 7 11:47:37 2021 -0400
 syntax error
 commit c5eec03365e09dedc81bac3e36cd45c5d2a2e0a2 Merge: c57185a c05578d Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Tue Apr 6 18:22:09 2021 -0400
 resolved merge conflict
 commit 50423cfe3940d93a321b765abd79c6b01af767e0 Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Sat Apr 3 17:22:34 2021 -0400
 modified semant and codegen syntax error in codegen
 commit c05578d194d1d87335d6ca67c011f38411b810a5 Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Fri Apr 2 21:03:49 2021 -0400
 Binary ops working
 commit 77dde105124825c069a1a87d76a97bf7a7b5398 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Apr 2 19:16:48 2021 -0400
 binop stuff
 commit 1d5c9ce7dbb73414b32a38b6eb3db89c5a9518d6 Author: tiffenychen <tc2963@columbia.edu>
 Date: Fri Apr 2 16:06:38 2021 -0400
 supports printing of floats, booleans, and ints
 commit 4ba1549914fa3fc83b916ed001aad2e845bbc16e Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 1 18:59:46 2021 -0400
 Hello World Working
 commit b16b974a0442d82bdf5089693591d84c77f5f249 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 1 01:53:09 2021 -0400
 pretty printing but not working
 commit 63cfab485dc075a33b8ee9655f60b447cedec16c Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 1 01:11:52 2021 -0400
 stmt vs stmt list in check_{tmt}insemant
 commit f0ad7a5dde0eabed6752122a882ae801ae362808 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Apr 1 00:20:31 2021 -0400
 error from ommitting fd.typ in semant.ml for ActionCall
 commit 3dece4ebc151252a12e51678985d1c115df68eb2 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Wed Mar 31 18:19:06 2021 -0400
 added classhelpercall and external helper for built-in function support to ast,parser,scanner
 commit c57185a120f5cf77a82365322fda0c5bd024c606 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Thu Mar 25 21:15:44 2021 -0400
 added some pretty print stuff
 commit 0db62fffa2334512349dc1dd54a60477d9e82e92 Author: tiffenychen <tc2963@columbia.edu>
 Date: Thu Mar 25 00:05:05 2021 -0400
 Move README to correct location
 commit 309c90640a28f1c22f67909d162eb2da1b785051 Author: tiffenychen <tc2963@columbia.edu>
 Date: Thu Mar 25 00:01:40 2021 -0400
 submitted
 commit a6131aa07a31b8f9327ae8fa0cb5f2cc5ab715ef Author: tiffenychen <tc2963@columbia.edu>
 Date: Wed Mar 24 01:00:42 2021 -0400
 not working
 commit 837df2ef8dc3f11cd2138a92f9d29f31ae354b82 Author: janggg <janghc39@gmail.com> Date:
 Tue Mar 23 03:17:21 2021 -0400
 fixed unbound module error

commit c7379253f8820864f16c48b80c1f1b85d89d7e36 Author: tiffenychen <tc2963@columbia.edu>
Date: Tue Mar 23 02:33:29 2021 -0400
added AHOD.ml and include native in Makefile, still has AHOD.native error
commit 8a0eb8d072113c091554236a6fa3bf4c57900dbe Author: tiffenychen <tc2963@columbia.edu>
Date: Tue Mar 23 00:17:38 2021 -0400
Test Script for codegen
commit c6714cc9b6d9d33376101fc5bd65979f6c965c80 Merge: 52e8693 38b9e46 Author: tiffenychen
<tc2963@columbia.edu> Date: Mon Mar 22 22:45:59 2021 -0400
Merging Merge branch 'master' of https://github.com/AllHandsOnDeck-PLT/AHOD into md_branch
commit 38b9e46b335ad7dfb793ab7c83b3bd5e023a5c1 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sun Mar 21 22:43:58 2021 -0400
added in string literal
commit 8173a133e60fb6166546bff155230d1a94b7e517 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sun Mar 21 22:22:14 2021 -0400
added sast dependencies to Makefile
commit 027568d3c563fb24c7ea9d89994afd9f78967d0c Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sun Mar 21 22:21:52 2021 -0400
fixed syntax error due to mutually-recursive types in sast
commit 8989ebfb0909e71c6770ea16da04795696457777 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sat Mar 20 19:22:18 2021 -0400
made some fixes to sast
commit 52e8693b788381752b158e839b39bef1a71c6419 Merge: 7349799 3b5d3c5 Author: tiffenychen
<tc2963@columbia.edu> Date: Fri Mar 19 19:39:24 2021 -0400
Merging Merge remote-tracking branch 'origin/master' into md_branch
commit 3b5d3c5c0a8802cb5f06739d756d967585ecab21 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 17:31:25 2021 -0400
created sast file and renamed ast, sast to .ml instead of .mli
commit 78e1c5fe4a6efbf909ebf54aacdaf05531a68cd2 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 15:20:37 2021 -0400
minor call edit
commit cc3c088ca6de0caaeeda9b247e3cf305f1f97d93 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 15:18:15 2021 -0400
converted program into record structure and cleaned up files
commit e5613d8a29b90a8170b4714e69a92d3610b524ce Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 15:07:02 2021 -0400
condensed action_declandconvertedintorecordstructure
commit a41cd5eebbcd6b1f0aac378aeedb2414500d72c7 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 14:01:37 2021 -0400
support for both 2-tuples and 3-tuples now by not overwriting fst and snd
commit 3a10c13ed7568c880dd1d5002ecee854b4ca5693 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 13:57:05 2021 -0400
condensed attr_declmore
commit ba65dbbbb40583dbe16e089b767bfa043dd7dfa2 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 13:50:33 2021 -0400
used option type (None null pointer and Some non-null pointer) to condense attr_declast
commit 80a986356126d7018bac5fe7603ec4dd038d90cb Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 13:22:21 2021 -0400
converted cdecl into record structure
commit 4bade0a8a5f58f04ef466b9fb8e6924bcc703749 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 13:06:47 2021 -0400

fixed Calls
 commit d7077412e910fd8bf0f4b7d7749330ade0a47eb5 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 12:54:09 2021 -0400
 single param, arg, stmt fix
 commit cf24d3c4a670831d85e3133cd14fd0944c3e581c Merge: 87f8dc8 1e972e3 Author: Caitlyn Chen
 <caitlyn.chen8@gmail.com> Date: Fri Mar 19 12:40:04 2021 -0400
 Merge pull request 4 from AllHandsOnDeck-PLT/unbound_{rd}
 Resolved Unbound trd error *class_{de}tupleerror*
 commit 1e972e3216aaab37b13bc94edbbbcf5bc7a67042 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 12:36:13 2021 -0400
 resolved *class_{de}error, thinkmaybeduetowaydefinedfstsndtrdthatmade3tuplesmandatory*
 commit 64d0d1ea2f69cb6462284b2a046eb057522481e0 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 06:58:30 2021 -0400
 resolved unbound value trd error, currently error with types in *class_{de}list*
 commit 7d752a64e9fd9a212db5402e472f2613f20876e9 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 04:38:58 2021 -0400
 ast support for *class_{de}butunboundvaluetrerror*
 commit 87f8dc8c9e83a83d1dd412ee2c922f072f7f5973 Merge: 6881f89 e70c2dd Author: junebug <47231340+junebug111@u>
 Date: Fri Mar 19 01:13:52 2021 -0400
 resolved merge conflict
 commit 6881f89e6806613f8c1a7c394cc0fe0c63969c688 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Fri Mar 19 01:11:51 2021 -0400
 working attr_{de}
 commit e70c2dd949d8fd3854f9b5826e0e2b09469f2fe4 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 00:55:00 2021 -0400
 decls tuple for action_{de}landhelper_{de}
 commit f172c59992aa7fbaeb553c97e7003c7c4169f2fe Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Fri Mar 19 00:48:10 2021 -0400
 added in action_{de}clastsupport
 commit 734979978980f4665ea127e444489603464193ab Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Fri Mar 19 00:21:06 2021 -0400
 barebone codegen
 commit 7a3360cd855347d07671598e9fc68acfd2f35661 Merge: 45fab93 3df8a20 Author: junebug <47231340+junebug111@u>
 Date: Thu Mar 18 23:50:27 2021 -0400
 Merge remote-tracking branch 'origin/master'
 commit 45fab9306bbcf24efafa8178628e0f1eadaf8b Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Thu Mar 18 23:48:21 2021 -0400
 edited helper decl and typ
 commit 3df8a203c1b904555353cc1d15ba426d64e5806e Author: Jang Hun Choi <janghc39@gmail.com>
 Date: Thu Mar 18 23:26:02 2021 -0400
 Add files via upload
 commit 74b09cfdbd6223c593775220849f576f43d2cc71 Author: junebug <47231340+junebug111@users.noreply.github.com>
 Date: Thu Mar 18 22:43:26 2021 -0400
 added param, args, some call stuff
 commit bb51bc4736a76518cdd9ca1dfc5e916686f009d8 Author: Mara Dimofte <mara.dimofte123@gmail.com>
 Date: Thu Mar 18 21:59:57 2021 -0400
 added call_s
 commit 08c7b558112e252449dc7fd9c0d5119603822536 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
 Date: Thu Mar 18 18:03:17 2021 -0400
 if stmt stmt block ast

commit 06597a364fb2199c78ef68118a4ecf4d862d4448 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Tue Mar 16 23:23:04 2021 -0400
make works

commit 2a2f46c830f412097eaa07a7a5f04e5f496e4710 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Tue Mar 16 23:16:05 2021 -0400
worked on stmt

commit d8ae53d3d0e52f60f66096b0c5b66aa73b3e315a Author: janggg <janghc39@gmail.com> Date:
Tue Mar 16 21:34:31 2021 -0400
commented out opt things

commit af2e8dc789dbc2d86844f6415d3c8b62a9950741 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Tue Mar 16 20:42:22 2021 -0400
worked on stmt for ast

commit e92ff4b8fd4ff5f0caafaa6efaf6373cc7565d4a Merge: 5867f14 2d86dfa Author: Caitlyn Chen
<caitlyn.chen8@gmail.com> Date: Sat Mar 6 05:06:39 2021 -0500
Merge pull request 1 from AllHandsOnDeck-PLT/cc,*ranch*
resolved major inconsistencies in grammar and shift/reduce reduce/reduce conflicts

commit 2d86dfa6b611f37efb7c29b219e64929260e9c21 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sat Mar 6 05:03:40 2021 -0500
resolved shift/reduce conflict with *const_opttyp_optandreduce/reduceconflictwithexpr_opt*

commit c3772eeec29f93f584da0ba616b7f2d7834f9807 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Sat Mar 6 04:17:17 2021 -0500
resolved reduce/reduce conflict for arg id = and assignment

commit 1650dafcfa645d4e8d51a9105b01b7b893ebbd27 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 21:31:27 2021 -0500
resolved 2 shift/reduce conflicts from NEWLINE

commit 6d18c813666be7bbfb1e911bc22c273b8b02bb00 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 05:10:08 2021 -0500
fixed up more inconsistencies from the grammar in the parser

commit 5867f1491e6e76d8b8eda22ffa97684a9fd0b991 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 03:50:21 2021 -0500
fixed decls in parser

commit 0ffdc5082b90ccfd1d29bfe7d1e476a1ec20b4d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 01:24:50 2021 -0500
dotted,*angeast*

commit 91b2a1f40aaed150a12720114943571314ddd1ed Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 01:15:04 2021 -0500
comprehension ast

commit 10925daca7fbcfe4ac0f80986c8619e0fe9eddfc Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Thu Mar 4 21:38:12 2021 -0500
fixed shift/reduce conflict for comprehension

commit a81b324b592e7c5c02d1aa5349b5e751e8c08ca9 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Thu Mar 4 21:14:20 2021 -0500
fixed Ast.expr errors

commit a3f69c0d3b55510ae319022921eb945e95904f93 Merge: d259877 dd8a6cc Author: junebug
<47231340+junebug111@users.noreply.github.com> Date: Thu Mar 4 19:15:54 2021 -0500
resolved merge conflicts

commit d259877f77dfc22b3a28e9d070ed63cb1c2c082e Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Thu Mar 4 18:59:26 2021 -0500
worked on ast

commit dd8a6cc4670031a380402b6498ab7fa78674654d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Thu Mar 4 18:04:18 2021 -0500
updated scanner

commit 1cad24acdd274d01b31320a6d1d80283432bbd67 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Thu Mar 4 17:26:51 2021 -0500
added Series*literal*

commit f333dcbf5d15bbbb5033349769817a8a45747f8d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Wed Mar 3 20:41:40 2021 -0500
updated scanner and parser

commit a35cc7587e4733dae9a4d2f61560056f12781190 Author: janggg <janghc39@gmail.com> Date:
Tue Mar 2 18:41:20 2021 -0500
updated parser

commit 5cdd5a0b68fad32a93b0c8698178b1dd9a22395 Author: tiffenychen <tc2963@columbia.edu>
Date: Wed Feb 24 23:25:16 2021 -0500
modified calls

commit e9343ee16ffdb9a8e8863154f00a5c8dbac2d8e4 Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 21:59:28 2021 -0500
modified stmts

commit b663f0946ed10e55d4735c0652b71df15042fb4b Author: tiffenychen <tc2963@columbia.edu>
Date: Wed Feb 24 15:25:09 2021 -0500
handled attr*declstmt*,*lock*and*class*,*lock*

commit 78e6c82bb0a3c0c0dc4e0a1ef94f3179750b880 Merge: c6ed764 efb1ae Author: tiffenychen
<tc2963@columbia.edu> Date: Wed Feb 24 11:56:27 2021 -0500
Merge branch 'master' of https://github.com/AllHandsOnDeck-PLT/AHOD

commit c6ed7644737cc37b329be447b414f3582533d855 Author: tiffenychen <tc2963@columbia.edu>
Date: Wed Feb 24 11:55:59 2021 -0500
delete parse.output

commit 56d1ef41c2f5c39d432a8668e7258368f9960904 Author: tiffenychen <tc2963@columbia.edu>
Date: Wed Feb 24 11:55:21 2021 -0500
added in optional args params typ const

commit efb1ae4a373086f6d831b8d6162d6870c19b231 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Wed Feb 24 11:55:17 2021 -0500
deleted ID for arg

commit 4fb929fc46692eb56baaf7e801a9de34ad445335 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Wed Feb 24 11:42:31 2021 -0500
modified params*list*,*args**list*

commit 7c82fe6528c8f7ac6d9d8165fad8983e29a612d6 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Wed Feb 24 11:09:40 2021 -0500
Modified clas, params*list*,*args**list*

commit 9d85002793ceaf8c1222c1e1627c2aa08084a38f Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Wed Feb 24 11:04:32 2021 -0500
udpated stmt

commit b06b24c37f0b02f97bf9a252a13f75e09e40cfa4 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Wed Feb 24 10:57:10 2021 -0500
updated expr

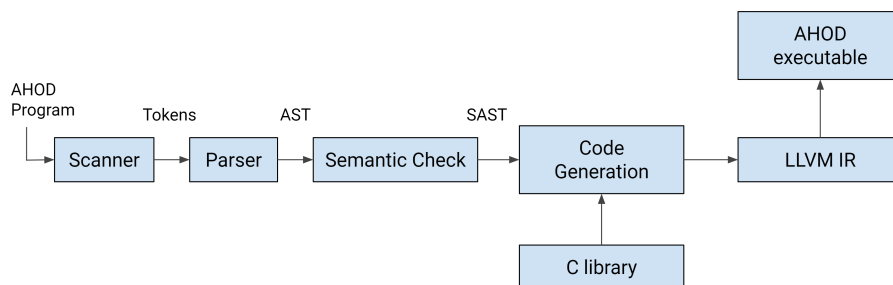
commit 4f27b87cd35716b77db78cdb103466a0d1506f35 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Wed Feb 24 09:40:29 2021 -0500
updated decl and spacing

commit 122d1c03b74cd279bb835cddbba9d893c0b872268 Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 01:45:31 2021 -0500

allocated parts
commit 93d7007d8562515b782bb61f19672bac42ace606 Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 01:38:37 2021 -0500
barebones complete
commit 2075a83ba61d3f2da97d0286e8b0b556549c3b7d Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 00:31:10 2021 -0500
compiling v1 of parser
commit cb895b92a1bfa19c2e668ee69ca3f9588a0007cd Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 00:04:46 2021 -0500
initial v2
commit df08fbafc509f02902de1da8c4f8548cad5dddfc Author: janggg <janghc39@gmail.com> Date:
Wed Feb 24 00:02:42 2021 -0500
initial commit

Chapter 6

Architectural Design



6.1 Scanner (scanner.ml)

The scanner takes in the AHOD program file (.ah) and generates tokens for the identifiers, keywords, operators, and literals specific to the AHOD language.

6.2 Parser (parse.mly, ast.ml)

The parser takes in the tokens generated from the scanner and converts them into an abstract syntax tree (AST) based on the rules of AHOD's grammar.

6.3 Semantic Checking (semant.ml, sast.ml)

During the semantic checking stage, the compiler performs type checking on the AST to verify that the nodes are semantically correct. If the AST passes this semantic check, then the SAST gets returned, which is the semantically checked version of the AST.

6.4 Code Generation (codegen.ml)

During code generation, we traverse the SAST and generate an intermediate representation (IR) of lower-level instructions. We also link to the built-in classes in our language that we wrote in a separate C file. Finally, the AHOD source code is converted into LLVM IR, which can then be compiled further into machine code.

Chapter 7

Test Plan

AHOD was developed on separate branches for different features (series, local assignment, classes, etc.) and aspects – such as scanner, parser, hello world. When features were compiling and functioning in the expected manner, the appropriate test cases were made. After all features were working, team members looked through the testing to ensure that all cases and possible risks of failure were covered or accounted for. If they weren't, then the corresponding tests were added. All team members contributed to the testing development.

In the testing suite, there are both success and fail testing cases. In total, there are 61 test cases.

7.1 Testing Suite and Automation

All tests are in the AHOD/test directory and have file extension "ah", i.e. "test.ah". Success cases are prefixed with "test-", while fail cases are prefixed with "fail-". For output files, success cases have file extension "out", while fail cases have those of "err".

Testing automation is based off of the provided Micro-C testing suite. The script, testall.sh, compiles and runs all of the *.ah files in the AHOD/test directory and compares the output to the corresponding output file also in the same directory – success has *.out format, while fail case has *.err format. When AHOD is compiled, or when make is inputted, the testing script runs.

7.2 Example Test Programs

7.2.1 AHOD Code 1

Testing global and local assignments, control flow statements (if, for)

```
1  main:
2  {
3      float b
4      b = 5.0
5      if 3<5:
6      {
7          for (a = 0 ; a < 5; a = a + 1):
8              {
9                  b = b + 1.0
```

```

10         do PRINT(b)
11     }
12 }
13 }

```

AHOD Code 1 ll file:

```

1 ; ModuleID = 'AHOD'
2 source_filename = "AHOD"
3
4 @str = global [4 x i8] c"%s\0A\00"
5 @str.1 = global [4 x i8] c"%d\0A\00"
6 @str.2 = global [4 x i8] c"%g\0A\00"
7 @str.3 = global [4 x i8] c"%d\0A\00"
8 @a = global i32 0
9
10 declare i32 @printf(i8*, ...)
11
12 declare { i8*, i32 } @playercall(i8*, i32)
13
14 declare i8* @getplayername({ i8*, i32 })
15
16 declare i32 @getplayerscore({ i8*, i32 })
17
18 declare { i8*, i1, i32 } @cardcall(i8*, i1, i32)
19
20 declare i8* @getcardtype({ i8*, i1, i32 })
21
22 declare i1 @getcardfaceup({ i8*, i1, i32 })
23
24 declare i32 @getcardvalue({ i8*, i1, i32 })
25
26 define i32 @main() {
27     entry:
28         %b = alloca double
29         store double 5.000000e+00, double* %b
30         br i1 true, label %then, label %else
31
32     merge:                                     ; preds = %else, %merge6
33         ret i32 0
34
35     then:                                       ; preds = %entry
36         store i32 0, i32* @a
37         br label %while
38
39     while:                                      ; preds = %while_body, %then
40         %a4 = load i32, i32* @a
41         %tmp5 = icmp slt i32 %a4, 5
42         br i1 %tmp5, label %while_body, label %merge6
43
44     while_body:                                ; preds = %while
45         %b1 = load double, double* %b

```



```

46  %tmp = fadd double %b1, 1.000000e+00
47  store double %tmp, double* %b
48  %b2 = load double, double* %b
49  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str.2, i32 0, i32 0), double %b2)
50  %a = load i32, i32* @a
51  %tmp3 = add i32 %a, 1
52  store i32 %tmp3, i32* @a
53  br label %while
54
55  merge6:                                     ; preds = %while
56  br label %merge
57
58  else:                                       ; preds = %entry
59  br label %merge
60 }
61
62 define void @series_pushbool({ i32*, i1* }*, i1) {
63  entry:
64  %series_ptr_alloc = alloca { i32*, i1* }*
65  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
66  %val_alloc = alloca i1
67  store i1 %1, i1* %val_alloc
68  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
69  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
70  %series_load_2 = load i1*, i1** %series_ptr_2
71  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
72  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
73  %series_size = load i32, i32* %series_size_ptr
74  %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
75  %next_size = add i32 %series_size, 1
76  store i32 %next_size, i32* %series_size_ptr
77  %val = load i1, i1* %val_alloc
78  store i1 %val, i1* %series_next_el_ptr
79  ret void
80 }
81
82 define void @series_pushint({ i32*, i32* }*, i32) {
83  entry:
84  %series_ptr_alloc = alloca { i32*, i32* }*
85  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
86  %val_alloc = alloca i32
87  store i32 %1, i32* %val_alloc
88  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
89  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
90  %series_load_2 = load i32*, i32** %series_ptr_2
91  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
92  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
93  %series_size = load i32, i32* %series_size_ptr
94  %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
95  %next_size = add i32 %series_size, 1
96  store i32 %next_size, i32* %series_size_ptr
97  %val = load i32, i32* %val_alloc

```

```

98     store i32 %val, i32* %series_next_el_ptr
99     ret void
100 }
101
102 define void @series_pushfloat({ i32*, double* }*, double) {
103 entry:
104     %series_ptr_alloc = alloca { i32*, double* }*
105     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
106     %val_alloc = alloca double
107     store double %1, double* %val_alloc
108     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
109     %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
110     %series_load_2 = load double*, double** %series_ptr_2
111     %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
112     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
113     %series_size = load i32, i32* %series_size_ptr
114     %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
115     %next_size = add i32 %series_size, 1
116     store i32 %next_size, i32* %series_size_ptr
117     %val = load double, double* %val_alloc
118     store double %val, double* %series_next_el_ptr
119     ret void
120 }
121
122 define void @series_pushstr({ i32*, i8** }*, i8*) {
123 entry:
124     %series_ptr_alloc = alloca { i32*, i8** }*
125     store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
126     %val_alloc = alloca i8*
127     store i8* %1, i8** %val_alloc
128     %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
129     %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
130     %series_load_2 = load i8**, i8*** %series_ptr_2
131     %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
132     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
133     %series_size = load i32, i32* %series_size_ptr
134     %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size
135     %next_size = add i32 %series_size, 1
136     store i32 %next_size, i32* %series_size_ptr
137     %val = load i8*, i8** %val_alloc
138     store i8* %val, i8** %series_next_el_ptr
139     ret void
140 }
141
142 define void @series_pushcard({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
143 entry:
144     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
145     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
146     %val_alloc = alloca { i8*, i1, i32 }
147     store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
148     %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
149     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32

```

```

150 %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
151 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
152 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
153 %series_size = load i32, i32* %series_size_ptr
154 %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
155 %next_size = add i32 %series_size, 1
156 store i32 %next_size, i32* %series_size_ptr
157 %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
158 store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
159 ret void
160 }
161
162 define void @series_pushplayer({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
163 entry:
164 %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
165 store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
166 %val_alloc = alloca { i8*, i32 }
167 store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
168 %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
169 %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
170 %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
171 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
172 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
173 %series_size = load i32, i32* %series_size_ptr
174 %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
175 %next_size = add i32 %series_size, 1
176 store i32 %next_size, i32* %series_size_ptr
177 %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
178 store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
179 ret void
180 }
181
182 define i1 @series_getbool({ i32*, i1* }*, i32) {
183 entry:
184 %series_ptr_alloc = alloca { i32*, i1* }*
185 store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
186 %idx_alloc = alloca i32
187 store i32 %1, i32* %idx_alloc
188 %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
189 %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
190 %array_load = load i1*, i1** %series_ptr_2
191 %idx_load = load i32, i32* %idx_alloc
192 %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
193 %series_el_ptr1 = load i1, i1* %series_el_ptr
194 ret i1 %series_el_ptr1
195 }
196
197 define i32 @series_getint({ i32*, i32* }*, i32) {
198 entry:
199 %series_ptr_alloc = alloca { i32*, i32* }*
200 store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
201 %idx_alloc = alloca i32

```

```

202     store i32 %1, i32* %idx_alloc
203     %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
204     %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
205     %array_load = load i32*, i32** %series_ptr_2
206     %idx_load = load i32, i32* %idx_alloc
207     %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
208     %series_el_ptr1 = load i32, i32* %series_el_ptr
209     ret i32 %series_el_ptr1
210 }
211
212 define double @series_getfloat({ i32*, double* }*, i32) {
213 entry:
214     %series_ptr_alloc = alloca { i32*, double* }*
215     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
216     %idx_alloc = alloca i32
217     store i32 %1, i32* %idx_alloc
218     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
219     %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
220     %array_load = load double*, double** %series_ptr_2
221     %idx_load = load i32, i32* %idx_alloc
222     %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
223     %series_el_ptr1 = load double, double* %series_el_ptr
224     ret double %series_el_ptr1
225 }
226
227 define i8* @series_getstr({ i32*, i8** }*, i32) {
228 entry:
229     %series_ptr_alloc = alloca { i32*, i8** }*
230     store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
231     %idx_alloc = alloca i32
232     store i32 %1, i32* %idx_alloc
233     %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
234     %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
235     %array_load = load i8**, i8*** %series_ptr_2
236     %idx_load = load i32, i32* %idx_alloc
237     %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
238     %series_el_ptr1 = load i8*, i8** %series_el_ptr
239     ret i8* %series_el_ptr1
240 }
241
242 define { i8*, i1, i32 } @series_getcard({ i32*, { i8*, i1, i32 }* }*, i32) {
243 entry:
244     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
245     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
246     %idx_alloc = alloca i32
247     store i32 %1, i32* %idx_alloc
248     %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
249     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 1
250     %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
251     %idx_load = load i32, i32* %idx_alloc
252     %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
253     %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr

```

```

254     ret { i8*, i1, i32 } %series_el_ptr1
255 }
256
257 define { i8*, i32 } @series_getplayer({ i32*, { i8*, i32 }* }*, i32) {
258 entry:
259     %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
260     store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
261     %idx_alloc = alloca i32
262     store i32 %1, i32* %idx_alloc
263     %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
264     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
265     %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
266     %idx_load = load i32, i32* %idx_alloc
267     %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
268     %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
269     ret { i8*, i32 } %series_el_ptr1
270 }
271
272 define i32 @series_sizebool({ i32*, i1* }*) {
273 entry:
274     %series_ptr_alloc = alloca { i32*, i1* }*
275     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
276     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
277     %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
278     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
279     %series_size = load i32, i32* %series_size_ptr
280     ret i32 %series_size
281 }
282
283 define i32 @series_sizeint({ i32*, i32* }*) {
284 entry:
285     %series_ptr_alloc = alloca { i32*, i32* }*
286     store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
287     %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
288     %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
289     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
290     %series_size = load i32, i32* %series_size_ptr
291     ret i32 %series_size
292 }
293
294 define i32 @series_sizefloat({ i32*, double* }*) {
295 entry:
296     %series_ptr_alloc = alloca { i32*, double* }*
297     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
298     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
299     %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
300     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
301     %series_size = load i32, i32* %series_size_ptr
302     ret i32 %series_size
303 }
304
305 define i32 @series_sizestr({ i32*, i8** }*) {

```

```

306 entry:
307   %series_ptr_alloc = alloca { i32*, i8** }*
308   store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
309   %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
310   %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
311   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
312   %series_size = load i32, i32* %series_size_ptr
313   ret i32 %series_size
314 }
315
316 define i32 @series_sizecard({ i32*, { i8*, i1, i32 }* }*) {
317 entry:
318   %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
319   store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
320   %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
321   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 0
322   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
323   %series_size = load i32, i32* %series_size_ptr
324   ret i32 %series_size
325 }
326
327 define i32 @series_sizeplayer({ i32*, { i8*, i32 }* }*) {
328 entry:
329   %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
330   store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
331   %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
332   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 0
333   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
334   %series_size = load i32, i32* %series_size_ptr
335   ret i32 %series_size
336 }
337
338 define i1 @series_popbool({ i32*, i1* }*) {
339 entry:
340   %series_ptr_alloc = alloca { i32*, i1* }*
341   store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
342   %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
343   %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
344   %series_arr_load = load i1*, i1** %series_arr_ptr
345   %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
346   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
347   %series_size = load i32, i32* %series_size_ptr
348   %dec_size = sub i32 %series_size, 1
349   %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
350   %series_array_next_element = load i1, i1* %series_next_el_ptr
351   store i32 %dec_size, i32* %series_size_ptr
352   ret i1 %series_array_next_element
353 }
354
355 define i32 @series_popint({ i32*, i32* }*) {
356 entry:
357   %series_ptr_alloc = alloca { i32*, i32* }*

```

```

358     store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
359     %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
360     %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
361     %series_arr_load = load i32*, i32** %series_arr_ptr
362     %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
363     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
364     %series_size = load i32, i32* %series_size_ptr
365     %dec_size = sub i32 %series_size, 1
366     %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
367     %series_array_next_element = load i32, i32* %series_next_el_ptr
368     store i32 %dec_size, i32* %series_size_ptr
369     ret i32 %series_array_next_element
370 }
371
372 define double @series_popfloat({ i32*, double* }*) {
373     entry:
374     %series_ptr_alloc = alloca { i32*, double* }*
375     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
376     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
377     %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
378     %series_arr_load = load double*, double** %series_arr_ptr
379     %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
380     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
381     %series_size = load i32, i32* %series_size_ptr
382     %dec_size = sub i32 %series_size, 1
383     %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
384     %series_array_next_element = load double, double* %series_next_el_ptr
385     store i32 %dec_size, i32* %series_size_ptr
386     ret double %series_array_next_element
387 }
388
389 define i8* @series_popstr({ i32*, i8** }*) {
390     entry:
391     %series_ptr_alloc = alloca { i32*, i8** }*
392     store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
393     %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
394     %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
395     %series_arr_load = load i8**, i8*** %series_arr_ptr
396     %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
397     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
398     %series_size = load i32, i32* %series_size_ptr
399     %dec_size = sub i32 %series_size, 1
400     %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
401     %series_array_next_element = load i8*, i8** %series_next_el_ptr
402     store i32 %dec_size, i32* %series_size_ptr
403     ret i8* %series_array_next_element
404 }
405
406 define { i8*, i1, i32 } @series_popcard({ i32*, { i8*, i1, i32 }* }*) {
407     entry:
408     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
409     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc

```

```

410 %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
411 %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i
412 %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
413 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
414 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
415 %series_size = load i32, i32* %series_size_ptr
416 %dec_size = sub i32 %series_size, 1
417 %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
418 %series_array_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
419 store i32 %dec_size, i32* %series_size_ptr
420 ret { i8*, i1, i32 } %series_array_next_element
421 }
422
423 define { i8*, i32 } @series_popplayer({ i32*, { i8*, i32 }* }*) {
424 entry:
425 %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
426 store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
427 %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
428 %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
429 %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
430 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
431 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
432 %series_size = load i32, i32* %series_size_ptr
433 %dec_size = sub i32 %series_size, 1
434 %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
435 %series_array_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
436 store i32 %dec_size, i32* %series_size_ptr
437 ret { i8*, i32 } %series_array_next_element
438 }

```

7.2.2 AHOD Code 2

Testing action declarations, built in card class, and series.

```

1 when do series<Card> CREATEPILE():
2 {
3     Card card1
4     card1 = Card("R5", true, 5)
5     return [card1]
6 }
7 main:
8 {
9     series<Card> cards
10    Card card
11    cards = do CREATEPILE()
12    card = cards[0]
13    do PRINT(card.type)
14    do PRINT(card.faceup)
15 }

```

AHOD Code 2 corresponding ll file:

```

1 ; ModuleID = 'AHOD'
2 source_filename = "AHOD"
3
4 @str = global [4 x i8] c"%s\0A\00"
5 @str.1 = global [4 x i8] c"%d\0A\00"
6 @str.2 = global [4 x i8] c"%g\0A\00"
7 @str.3 = global [4 x i8] c"%d\0A\00"
8 @str.4 = private unnamed_addr constant [3 x i8] c"R5\00"
9
10 declare i32 @printf(i8*, ...)
11
12 declare { i8*, i32 } @playercall(i8*, i32)
13
14 declare i8* @getplayername({ i8*, i32 })
15
16 declare i32 @getplayerscore({ i8*, i32 })
17
18 declare { i8*, i1, i32 } @cardcall(i8*, i1, i32)
19
20 declare i8* @getcardtype({ i8*, i1, i32 })
21
22 declare i1 @getcardfaceup({ i8*, i1, i32 })
23
24 declare i32 @getcardvalue({ i8*, i1, i32 })
25
26 define { i32*, { i8*, i1, i32 }* } @CREATEPILE() {
27   entry:
28     %card1 = alloca { i8*, i1, i32 }
29     %cardcall = call { i8*, i1, i32 } @cardcall(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @str.4, i32 0, i32 0), i1 true)
30     store { i8*, i1, i32 } %cardcall, { i8*, i1, i32 }* %card1
31     %new_series_ptr = alloca { i32*, { i8*, i1, i32 }* }
32     %series_size_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr, i32 0
33     %series_size = alloca i32
34     store i32 0, i32* %series_size
35     store i32* %series_size, i32** %series_size_ptr
36     %series = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr, i32 0, i32 1
37     %p = alloca { i8*, i1, i32 }, i32 1028
38     store { i8*, i1, i32 }* %p, { i8*, i1, i32 }** %series
39     %card11 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card1
40     call void @series_pushcard({ i32*, { i8*, i1, i32 }* }* %new_series_ptr, { i8*, i1, i32 } %card11)
41     %new_series = load { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr
42     ret { i32*, { i8*, i1, i32 }* } %new_series
43 }
44
45 define void @series_pushbool({ i32*, i1* }*, i1) {
46   entry:
47     %series_ptr_alloc = alloca { i32*, i1* }*
48     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
49     %val_alloc = alloca i1
50     store i1 %1, i1* %val_alloc
51     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc

```

```

52  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
53  %series_load_2 = load i1*, i1** %series_ptr_2
54  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
55  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
56  %series_size = load i32, i32* %series_size_ptr
57  %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
58  %next_size = add i32 %series_size, 1
59  store i32 %next_size, i32* %series_size_ptr
60  %val = load i1, i1* %val_alloc
61  store i1 %val, i1* %series_next_el_ptr
62  ret void
63 }
64
65 define void @series_pushint({ i32*, i32* }*, i32) {
66 entry:
67  %series_ptr_alloc = alloca { i32*, i32* }*
68  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
69  %val_alloc = alloca i32
70  store i32 %1, i32* %val_alloc
71  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
72  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
73  %series_load_2 = load i32*, i32** %series_ptr_2
74  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
75  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
76  %series_size = load i32, i32* %series_size_ptr
77  %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
78  %next_size = add i32 %series_size, 1
79  store i32 %next_size, i32* %series_size_ptr
80  %val = load i32, i32* %val_alloc
81  store i32 %val, i32* %series_next_el_ptr
82  ret void
83 }
84
85 define void @series_pushfloat({ i32*, double* }*, double) {
86 entry:
87  %series_ptr_alloc = alloca { i32*, double* }*
88  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
89  %val_alloc = alloca double
90  store double %1, double* %val_alloc
91  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
92  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
93  %series_load_2 = load double*, double** %series_ptr_2
94  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
95  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
96  %series_size = load i32, i32* %series_size_ptr
97  %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
98  %next_size = add i32 %series_size, 1
99  store i32 %next_size, i32* %series_size_ptr
100 %val = load double, double* %val_alloc
101 store double %val, double* %series_next_el_ptr
102 ret void
103 }

```

```

104
105 define void @series_pushstr({ i32*, i8** }*, i8*) {
106 entry:
107   %series_ptr_alloc = alloca { i32*, i8** }*
108   store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
109   %val_alloc = alloca i8*
110   store i8* %1, i8** %val_alloc
111   %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
112   %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
113   %series_load_2 = load i8**, i8*** %series_ptr_2
114   %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
115   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
116   %series_size = load i32, i32* %series_size_ptr
117   %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size
118   %next_size = add i32 %series_size, 1
119   store i32 %next_size, i32* %series_size_ptr
120   %val = load i8*, i8** %val_alloc
121   store i8* %val, i8** %series_next_el_ptr
122   ret void
123 }
124
125 define void @series_pushcard({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
126 entry:
127   %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
128   store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
129   %val_alloc = alloca { i8*, i1, i32 }
130   store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
131   %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
132   %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 1
133   %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
134   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 0
135   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
136   %series_size = load i32, i32* %series_size_ptr
137   %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
138   %next_size = add i32 %series_size, 1
139   store i32 %next_size, i32* %series_size_ptr
140   %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
141   store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
142   ret void
143 }
144
145 define void @series_pushplayer({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
146 entry:
147   %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
148   store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
149   %val_alloc = alloca { i8*, i32 }
150   store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
151   %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
152   %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
153   %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
154   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 0
155   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr

```

```

156 %series_size = load i32, i32* %series_size_ptr
157 %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
158 %next_size = add i32 %series_size, 1
159 store i32 %next_size, i32* %series_size_ptr
160 %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
161 store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
162 ret void
163 }
164
165 define i1 @series_getbool({ i32*, i1* }*, i32) {
166 entry:
167 %series_ptr_alloc = alloca { i32*, i1* }*
168 store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
169 %idx_alloc = alloca i32
170 store i32 %1, i32* %idx_alloc
171 %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
172 %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
173 %array_load = load i1*, i1** %series_ptr_2
174 %idx_load = load i32, i32* %idx_alloc
175 %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
176 %series_el_ptr1 = load i1, i1* %series_el_ptr
177 ret i1 %series_el_ptr1
178 }
179
180 define i32 @series_getint({ i32*, i32* }*, i32) {
181 entry:
182 %series_ptr_alloc = alloca { i32*, i32* }*
183 store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
184 %idx_alloc = alloca i32
185 store i32 %1, i32* %idx_alloc
186 %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
187 %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
188 %array_load = load i32*, i32** %series_ptr_2
189 %idx_load = load i32, i32* %idx_alloc
190 %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
191 %series_el_ptr1 = load i32, i32* %series_el_ptr
192 ret i32 %series_el_ptr1
193 }
194
195 define double @series_getfloat({ i32*, double* }*, i32) {
196 entry:
197 %series_ptr_alloc = alloca { i32*, double* }*
198 store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
199 %idx_alloc = alloca i32
200 store i32 %1, i32* %idx_alloc
201 %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
202 %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
203 %array_load = load double*, double** %series_ptr_2
204 %idx_load = load i32, i32* %idx_alloc
205 %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
206 %series_el_ptr1 = load double, double* %series_el_ptr
207 ret double %series_el_ptr1

```

```

208 }
209
210 define i8* @series_getstr({ i32*, i8** }*, i32) {
211 entry:
212   %series_ptr_alloc = alloca { i32*, i8** }*
213   store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
214   %idx_alloc = alloca i32
215   store i32 %1, i32* %idx_alloc
216   %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
217   %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
218   %array_load = load i8**, i8*** %series_ptr_2
219   %idx_load = load i32, i32* %idx_alloc
220   %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
221   %series_el_ptr1 = load i8*, i8** %series_el_ptr
222   ret i8* %series_el_ptr1
223 }
224
225 define { i8*, i1, i32 } @series_getcard({ i32*, { i8*, i1, i32 }* }*, i32) {
226 entry:
227   %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
228   store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
229   %idx_alloc = alloca i32
230   store i32 %1, i32* %idx_alloc
231   %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
232   %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 1
233   %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
234   %idx_load = load i32, i32* %idx_alloc
235   %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
236   %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr
237   ret { i8*, i1, i32 } %series_el_ptr1
238 }
239
240 define { i8*, i32 } @series_getplayer({ i32*, { i8*, i32 }* }*, i32) {
241 entry:
242   %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
243   store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
244   %idx_alloc = alloca i32
245   store i32 %1, i32* %idx_alloc
246   %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
247   %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
248   %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
249   %idx_load = load i32, i32* %idx_alloc
250   %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
251   %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
252   ret { i8*, i32 } %series_el_ptr1
253 }
254
255 define i32 @series_sizebool({ i32*, i1* }*) {
256 entry:
257   %series_ptr_alloc = alloca { i32*, i1* }*
258   store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
259   %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc

```

```

260 %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
261 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
262 %series_size = load i32, i32* %series_size_ptr
263 ret i32 %series_size
264 }
265
266 define i32 @series_sizeint({ i32*, i32* }*) {
267 entry:
268 %series_ptr_alloc = alloca { i32*, i32* }*
269 store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
270 %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
271 %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
272 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
273 %series_size = load i32, i32* %series_size_ptr
274 ret i32 %series_size
275 }
276
277 define i32 @series_sizefloat({ i32*, double* }*) {
278 entry:
279 %series_ptr_alloc = alloca { i32*, double* }*
280 store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
281 %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
282 %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
283 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
284 %series_size = load i32, i32* %series_size_ptr
285 ret i32 %series_size
286 }
287
288 define i32 @series_sizestr({ i32*, i8** }*) {
289 entry:
290 %series_ptr_alloc = alloca { i32*, i8** }*
291 store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
292 %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
293 %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
294 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
295 %series_size = load i32, i32* %series_size_ptr
296 ret i32 %series_size
297 }
298
299 define i32 @series_sizecard({ i32*, { i8*, i1, i32 }* }*) {
300 entry:
301 %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
302 store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
303 %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
304 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 0
305 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
306 %series_size = load i32, i32* %series_size_ptr
307 ret i32 %series_size
308 }
309
310 define i32 @series_sizeplayer({ i32*, { i8*, i32 }* }*) {
311 entry:

```

```

312  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
313  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
314  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
315  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
316  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
317  %series_size = load i32, i32* %series_size_ptr
318  ret i32 %series_size
319 }
320
321 define i1 @series_popbool({ i32*, i1* }*) {
322 entry:
323  %series_ptr_alloc = alloca { i32*, i1* }*
324  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
325  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
326  %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
327  %series_arr_load = load i1*, i1** %series_arr_ptr
328  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
329  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
330  %series_size = load i32, i32* %series_size_ptr
331  %dec_size = sub i32 %series_size, 1
332  %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
333  %series_array_next_element = load i1, i1* %series_next_el_ptr
334  store i32 %dec_size, i32* %series_size_ptr
335  ret i1 %series_array_next_element
336 }
337
338 define i32 @series_popint({ i32*, i32* }*) {
339 entry:
340  %series_ptr_alloc = alloca { i32*, i32* }*
341  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
342  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
343  %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
344  %series_arr_load = load i32*, i32** %series_arr_ptr
345  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
346  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
347  %series_size = load i32, i32* %series_size_ptr
348  %dec_size = sub i32 %series_size, 1
349  %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
350  %series_array_next_element = load i32, i32* %series_next_el_ptr
351  store i32 %dec_size, i32* %series_size_ptr
352  ret i32 %series_array_next_element
353 }
354
355 define double @series_popfloat({ i32*, double* }*) {
356 entry:
357  %series_ptr_alloc = alloca { i32*, double* }*
358  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
359  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
360  %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
361  %series_arr_load = load double*, double** %series_arr_ptr
362  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
363  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr

```

```

364   %series_size = load i32, i32* %series_size_ptr
365   %dec_size = sub i32 %series_size, 1
366   %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
367   %series_array_next_element = load double, double* %series_next_el_ptr
368   store i32 %dec_size, i32* %series_size_ptr
369   ret double %series_array_next_element
370 }
371
372 define i8* @series_popstr({ i32*, i8** }*) {
373 entry:
374   %series_ptr_alloc = alloca { i32*, i8** }*
375   store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
376   %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
377   %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
378   %series_arr_load = load i8**, i8*** %series_arr_ptr
379   %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
380   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
381   %series_size = load i32, i32* %series_size_ptr
382   %dec_size = sub i32 %series_size, 1
383   %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
384   %series_array_next_element = load i8*, i8** %series_next_el_ptr
385   store i32 %dec_size, i32* %series_size_ptr
386   ret i8* %series_array_next_element
387 }
388
389 define { i8*, i1, i32 } @series_popcard({ i32*, { i8*, i1, i32 }* }*) {
390 entry:
391   %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
392   store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
393   %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
394   %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 1
395   %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
396   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 0
397   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
398   %series_size = load i32, i32* %series_size_ptr
399   %dec_size = sub i32 %series_size, 1
400   %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
401   %series_array_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
402   store i32 %dec_size, i32* %series_size_ptr
403   ret { i8*, i1, i32 } %series_array_next_element
404 }
405
406 define { i8*, i32 } @series_popplayer({ i32*, { i8*, i32 }* }*) {
407 entry:
408   %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
409   store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
410   %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
411   %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
412   %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
413   %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 0
414   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
415   %series_size = load i32, i32* %series_size_ptr

```



```

416     %dec_size = sub i32 %series_size, 1
417     %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
418     %series_array_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
419     store i32 %dec_size, i32* %series_size_ptr
420     ret { i8*, i32 } %series_array_next_element
421 }
422
423 define i32 @main() {
424 entry:
425     %card = alloca { i8*, i1, i32 }
426     %cards = alloca { i32*, { i8*, i1, i32 }* }
427     %series_size_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %cards, i32 0, i32 0
428     %series_size = alloca i32
429     store i32 0, i32* %series_size
430     store i32* %series_size, i32** %series_size_ptr
431     %series.array = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %cards, i32 0, i32 1
432     %p = alloca { i8*, i1, i32 }, i32 1028
433     store { i8*, i1, i32 }* %p, { i8*, i1, i32 }** %series.array
434     %CREATEPILE_result = call { i32*, { i8*, i1, i32 }* } @CREATEPILE()
435     store { i32*, { i8*, i1, i32 }* } %CREATEPILE_result, { i32*, { i8*, i1, i32 }* }* %cards
436     %series_get = call { i8*, i1, i32 } @series_getcard.15({ i32*, { i8*, i1, i32 }* }* %cards, i32 0)
437     store { i8*, i1, i32 } %series_get, { i8*, i1, i32 }* %card
438     %card1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card
439     %getcardtype = call i8* @getcardtype({ i8*, i1, i32 } %card1)
440     %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str, i32 0, i32 0), i8* %getcardty
441     %card2 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card
442     %getcardfaceup = call i1 @getcardfaceup({ i8*, i1, i32 } %card2)
443     %printf3 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str.3, i32 0, i32 0), i1 %getcard
444     ret i32 0
445 }
446
447 define void @series_pushbool.5({ i32*, i1* }*, i1) {
448 entry:
449     %series_ptr_alloc = alloca { i32*, i1* }*
450     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
451     %val_alloc = alloca i1
452     store i1 %1, i1* %val_alloc
453     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
454     %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
455     %series_load_2 = load i1*, i1** %series_ptr_2
456     %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
457     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
458     %series_size = load i32, i32* %series_size_ptr
459     %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
460     %next_size = add i32 %series_size, 1
461     store i32 %next_size, i32* %series_size_ptr
462     %val = load i1, i1* %val_alloc
463     store i1 %val, i1* %series_next_el_ptr
464     ret void
465 }
466
467 define void @series_pushint.6({ i32*, i32* }*, i32) {

```

```

468 entry:
469   %series_ptr_alloc = alloca { i32*, i32* }*
470   store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
471   %val_alloc = alloca i32
472   store i32 %1, i32* %val_alloc
473   %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
474   %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
475   %series_load_2 = load i32*, i32** %series_ptr_2
476   %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
477   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
478   %series_size = load i32, i32* %series_size_ptr
479   %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
480   %next_size = add i32 %series_size, 1
481   store i32 %next_size, i32* %series_size_ptr
482   %val = load i32, i32* %val_alloc
483   store i32 %val, i32* %series_next_el_ptr
484   ret void
485 }
486
487 define void @series_pushfloat.7({ i32*, double* }*, double) {
488 entry:
489   %series_ptr_alloc = alloca { i32*, double* }*
490   store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
491   %val_alloc = alloca double
492   store double %1, double* %val_alloc
493   %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
494   %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
495   %series_load_2 = load double*, double** %series_ptr_2
496   %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
497   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
498   %series_size = load i32, i32* %series_size_ptr
499   %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
500   %next_size = add i32 %series_size, 1
501   store i32 %next_size, i32* %series_size_ptr
502   %val = load double, double* %val_alloc
503   store double %val, double* %series_next_el_ptr
504   ret void
505 }
506
507 define void @series_pushstr.8({ i32*, i8** }*, i8*) {
508 entry:
509   %series_ptr_alloc = alloca { i32*, i8** }*
510   store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
511   %val_alloc = alloca i8*
512   store i8* %1, i8** %val_alloc
513   %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
514   %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
515   %series_load_2 = load i8**, i8*** %series_ptr_2
516   %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
517   %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
518   %series_size = load i32, i32* %series_size_ptr
519   %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size

```

```

520     %next_size = add i32 %series_size, 1
521     store i32 %next_size, i32* %series_size_ptr
522     %val = load i8*, i8** %val_alloc
523     store i8* %val, i8** %series_next_el_ptr
524     ret void
525 }
526
527 define void @series_pushcard.9({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
528 entry:
529     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
530     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
531     %val_alloc = alloca { i8*, i1, i32 }
532     store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
533     %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
534     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
535     %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
536     %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
537     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
538     %series_size = load i32, i32* %series_size_ptr
539     %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
540     %next_size = add i32 %series_size, 1
541     store i32 %next_size, i32* %series_size_ptr
542     %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
543     store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
544     ret void
545 }
546
547 define void @series_pushplayer.10({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
548 entry:
549     %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
550     store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
551     %val_alloc = alloca { i8*, i32 }
552     store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
553     %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
554     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
555     %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
556     %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
557     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
558     %series_size = load i32, i32* %series_size_ptr
559     %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
560     %next_size = add i32 %series_size, 1
561     store i32 %next_size, i32* %series_size_ptr
562     %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
563     store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
564     ret void
565 }
566
567 define i1 @series_getbool.11({ i32*, i1* }*, i32) {
568 entry:
569     %series_ptr_alloc = alloca { i32*, i1* }*
570     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
571     %idx_alloc = alloca i32

```

```

572     store i32 %1, i32* %idx_alloc
573     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
574     %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
575     %array_load = load i1*, i1** %series_ptr_2
576     %idx_load = load i32, i32* %idx_alloc
577     %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
578     %series_el_ptr1 = load i1, i1* %series_el_ptr
579     ret i1 %series_el_ptr1
580 }
581
582 define i32 @series_getint.12({ i32*, i32* }*, i32) {
583 entry:
584     %series_ptr_alloc = alloca { i32*, i32* }*
585     store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
586     %idx_alloc = alloca i32
587     store i32 %1, i32* %idx_alloc
588     %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
589     %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
590     %array_load = load i32*, i32** %series_ptr_2
591     %idx_load = load i32, i32* %idx_alloc
592     %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
593     %series_el_ptr1 = load i32, i32* %series_el_ptr
594     ret i32 %series_el_ptr1
595 }
596
597 define double @series_getfloat.13({ i32*, double* }*, i32) {
598 entry:
599     %series_ptr_alloc = alloca { i32*, double* }*
600     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
601     %idx_alloc = alloca i32
602     store i32 %1, i32* %idx_alloc
603     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
604     %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
605     %array_load = load double*, double** %series_ptr_2
606     %idx_load = load i32, i32* %idx_alloc
607     %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
608     %series_el_ptr1 = load double, double* %series_el_ptr
609     ret double %series_el_ptr1
610 }
611
612 define i8* @series_getstr.14({ i32*, i8** }*, i32) {
613 entry:
614     %series_ptr_alloc = alloca { i32*, i8** }*
615     store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
616     %idx_alloc = alloca i32
617     store i32 %1, i32* %idx_alloc
618     %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
619     %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
620     %array_load = load i8**, i8*** %series_ptr_2
621     %idx_load = load i32, i32* %idx_alloc
622     %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
623     %series_el_ptr1 = load i8*, i8** %series_el_ptr

```

```

624     ret i8* %series_el_ptr1
625 }
626
627 define { i8*, i1, i32 } @series_getcard.15({ i32*, { i8*, i1, i32 }* }*, i32) {
628 entry:
629     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
630     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
631     %idx_alloc = alloca i32
632     store i32 %1, i32* %idx_alloc
633     %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
634     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
635     %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
636     %idx_load = load i32, i32* %idx_alloc
637     %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
638     %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr
639     ret { i8*, i1, i32 } %series_el_ptr1
640 }
641
642 define { i8*, i32 } @series_getplayer.16({ i32*, { i8*, i32 }* }*, i32) {
643 entry:
644     %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
645     store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
646     %idx_alloc = alloca i32
647     store i32 %1, i32* %idx_alloc
648     %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
649     %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
650     %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
651     %idx_load = load i32, i32* %idx_alloc
652     %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
653     %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
654     ret { i8*, i32 } %series_el_ptr1
655 }
656
657 define i32 @series_sizebool.17({ i32*, i1* }*) {
658 entry:
659     %series_ptr_alloc = alloca { i32*, i1* }*
660     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
661     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
662     %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
663     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
664     %series_size = load i32, i32* %series_size_ptr
665     ret i32 %series_size
666 }
667
668 define i32 @series_sizeint.18({ i32*, i32* }*) {
669 entry:
670     %series_ptr_alloc = alloca { i32*, i32* }*
671     store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
672     %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
673     %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
674     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
675     %series_size = load i32, i32* %series_size_ptr

```

```

676     ret i32 %series_size
677 }
678
679 define i32 @series_sizefloat.19({ i32*, double* }*) {
680 entry:
681     %series_ptr_alloc = alloca { i32*, double* }*
682     store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
683     %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
684     %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
685     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
686     %series_size = load i32, i32* %series_size_ptr
687     ret i32 %series_size
688 }
689
690 define i32 @series_sizestr.20({ i32*, i8** }*) {
691 entry:
692     %series_ptr_alloc = alloca { i32*, i8** }*
693     store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
694     %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
695     %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
696     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
697     %series_size = load i32, i32* %series_size_ptr
698     ret i32 %series_size
699 }
700
701 define i32 @series_sizecard.21({ i32*, { i8*, i1, i32 }* }*) {
702 entry:
703     %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
704     store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
705     %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
706     %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32 0
707     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
708     %series_size = load i32, i32* %series_size_ptr
709     ret i32 %series_size
710 }
711
712 define i32 @series_sizeplayer.22({ i32*, { i8*, i32 }* }*) {
713 entry:
714     %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
715     store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
716     %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
717     %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 0
718     %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
719     %series_size = load i32, i32* %series_size_ptr
720     ret i32 %series_size
721 }
722
723 define i1 @series_popbool.23({ i32*, i1* }*) {
724 entry:
725     %series_ptr_alloc = alloca { i32*, i1* }*
726     store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
727     %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc

```

```

728 %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
729 %series_arr_load = load i1*, i1** %series_arr_ptr
730 %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
731 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
732 %series_size = load i32, i32* %series_size_ptr
733 %dec_size = sub i32 %series_size, 1
734 %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
735 %series_array_next_element = load i1, i1* %series_next_el_ptr
736 store i32 %dec_size, i32* %series_size_ptr
737 ret i1 %series_array_next_element
738 }
739
740 define i32 @series_popint.24({ i32*, i32* }*) {
741 entry:
742 %series_ptr_alloc = alloca { i32*, i32* }*
743 store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
744 %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
745 %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
746 %series_arr_load = load i32*, i32** %series_arr_ptr
747 %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
748 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
749 %series_size = load i32, i32* %series_size_ptr
750 %dec_size = sub i32 %series_size, 1
751 %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
752 %series_array_next_element = load i32, i32* %series_next_el_ptr
753 store i32 %dec_size, i32* %series_size_ptr
754 ret i32 %series_array_next_element
755 }
756
757 define double @series_popfloat.25({ i32*, double* }*) {
758 entry:
759 %series_ptr_alloc = alloca { i32*, double* }*
760 store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
761 %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
762 %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
763 %series_arr_load = load double*, double** %series_arr_ptr
764 %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
765 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
766 %series_size = load i32, i32* %series_size_ptr
767 %dec_size = sub i32 %series_size, 1
768 %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
769 %series_array_next_element = load double, double* %series_next_el_ptr
770 store i32 %dec_size, i32* %series_size_ptr
771 ret double %series_array_next_element
772 }
773
774 define i8* @series_popstr.26({ i32*, i8** }*) {
775 entry:
776 %series_ptr_alloc = alloca { i32*, i8** }*
777 store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
778 %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
779 %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1

```

```

780 %series_arr_load = load i8**, i8*** %series_arr_ptr
781 %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
782 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
783 %series_size = load i32, i32* %series_size_ptr
784 %dec_size = sub i32 %series_size, 1
785 %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
786 %series_array_next_element = load i8*, i8** %series_next_el_ptr
787 store i32 %dec_size, i32* %series_size_ptr
788 ret i8* %series_array_next_element
789 }
790
791 define { i8*, i1, i32 } @series_popcard.27({ i32*, { i8*, i1, i32 }* }*) {
792 entry:
793 %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
794 store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
795 %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
796 %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i
797 %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
798 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
799 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
800 %series_size = load i32, i32* %series_size_ptr
801 %dec_size = sub i32 %series_size, 1
802 %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
803 %series_array_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
804 store i32 %dec_size, i32* %series_size_ptr
805 ret { i8*, i1, i32 } %series_array_next_element
806 }
807
808 define { i8*, i32 } @series_popplayer.28({ i32*, { i8*, i32 }* }*) {
809 entry:
810 %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
811 store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
812 %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
813 %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
814 %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
815 %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
816 %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
817 %series_size = load i32, i32* %series_size_ptr
818 %dec_size = sub i32 %series_size, 1
819 %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
820 %series_array_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
821 store i32 %dec_size, i32* %series_size_ptr
822 ret { i8*, i32 } %series_array_next_element
823 }

```

Chapter 8

Lessons Learned

8.1 Caitlyn Chen

Dividing and conquering individual features in the language to implement support for in the compiler back-end stage of the project sped up our efficiency. Pair programming during our group meetings was more efficient when group meetings were structured as work blocks rather than simply having brief asynchronous check-ins. Communication is key, and being proactive about updating the group and taking initiative was also really helpful. While at the beginning we assigned ourselves individual roles to take on in the project, and I would take point on all things language design and grammar related, by the end of the project stages, things were a lot more fluid. Posting private posts on piazza with as much detail as possible about whatever error we were facing, the relevant code snippets, and the steps we had already taken to solve it were always really helpful in giving us helpful guidance for how to proceed. Overall it was a challenging but super fulfilling project – seeing things work sparked much joy!

8.2 Tiffeny Chen

Plan out how you will implement your functionalities to the end! This is a bit tricky since you are learning new content while working on the project, so make sure to gain a thorough understanding of how microc functions. For helloworld, keep things simple, use microc as a guide and it'll help you to learn how to use entry blocks, pattern matching and read LLVM documentation. We initially tried to divide and conquer whole files but later found it was much more productive to divide up features and have members implement functionalities across all the files instead. Pair program earlier on as a group to help with any gaps and debugging together. Don't feel afraid to reach out on Piazza discussions or office hours, there are many helpful resources at your disposal. You will get through this and have a hopefully working compiler with your very own programming language!

8.3 Jang Hun Choi

Python is more complicated than what it advertises itself to be! I will never take its dynamic typing feature for granted again. While it may have been ambitious for us to dream of creating a python-like language, narrowing down the scope was probably better than having to increase it. Even 'til the last weekend of the project, our progress felt like Sisyphus trying to roll that stone uphill – getting somewhere but nowhere. But alas, the pure joy I felt when our smallest version of "demo code" behaved as expected was something else. Definitely would suggest knowing microc thoroughly!

8.4 Mara Dimofte

Planning ahead for potential sources of uncertainty related or unrelated to code is essential to accurately estimating how much time individual tasks take and thus ensuring everything is implemented in time. Communication among teammates, along with reaching for help whenever stuck is as relevant if not more so to the success of the project as the amount of work dedicated to it.

8.5 Christi Kim

Creating a language is fun, but making the corresponding compiler to process the language is an experience. I now understand the hard work that went behind creating popular everyday languages and I definitely have a greater appreciation for them. I'm not sure if I would have been able to create a functioning language without the motivation and hard work of my teammates, so I'm very glad to have had work beside them for this project.

Chapter 9

Appendix

Makefile

```
1  # "make test" Compiles everything and runs the regression tests
2
3  .PHONY : test
4  test : all testall.sh
5         ./testall.sh
6
7  # "make all" builds the executable as well as the "printbig" library designed
8  # to test linking external code
9
10 .PHONY : all
11 all : AHOD.native playercall.o
12
13 # "make microc.native" compiles the compiler
14 #
15 # The _tags file controls the operation of ocamlbuild, e.g., by including
16 # packages, enabling warnings
17 #
18 # See https://github.com/ocaml/ocamlbuild/blob/master/manual/manual.adoc
19
20 AHOD.native :
21     opam config exec -- \
22     ocamlbuild -use-ocamlfind AHOD.native
23
24 # "make clean" removes all generated files
25
26 playercall : playercall.c
27     cc -o playercall -DBUILD_TEST playercall.c
28
29 .PHONY : clean
30 clean :
31     ocamlbuild -clean
32     rm -rf testall.log ocamlllvm *.diff .ll .native *.o
```

scanner.mll

```

1  (* Ocamllex scanner for AHOD *)
2  { open Parse }
3
4  let digit = ['0' - '9']
5  let digits = digit+
6
7  rule token = parse
8    [' ' '\t' '\r' ] { token lexbuf } (* Whitespace *)
9    | "/" * "      { comment lexbuf }      (* Comments *)
10   | ('\n' [' ' ]*)+      { NEWLINE }
11   | '('          { LPAREN }
12   | ')'         { RPAREN }
13   | '{'         { LBRACE }
14   | '}'         { RBRACE }
15   | '['         { LSQUARE }
16   | ']'         { RSQUARE }
17   | "push"     { SERIESPUSH }
18   | "pop"      { SERIESPOP }
19   | "size"     { SERIESSIZE }
20   | "series"   { SERIES }
21   | "Player"   { PLAYER }
22   | "Card"     { CARD }
23   | ':'        { COLON }
24   | ';'        { SEMI }
25   | ','        { COMMA }
26   | '+'        { PLUS }
27   | '-'        { MINUS }
28   | '*'        { MULT }
29   | '/'        { DIVIDE }
30   | '.'        { DOT }
31   | '='        { ASSIGN }
32   | "=="       { EQ }
33   | "!="       { NEQ }
34   | "<"        { LT }
35   | ">"        { GT }
36   | "<="       { LEQ }
37   | ">="       { GEQ }
38   | "and"      { AND }
39   | "or"       { OR }
40   | "not"      { NOT }
41   | "if"       { IF }
42   | "else"     { ELSE }
43   | "for"      { FOR }
44   | "while"    { WHILE }
45   | "when"    { WHEN }
46   | "do"       { DO }
47   | "return"   { RETURN }
48   | "int"      { INT }
49   | "bool"     { BOOL }
50   | "string"   { STRING }
51   | "float"    { FLOAT }

```

```

52 | "void" { VOID }
53 | "true" { BLIT(true) }
54 | "false" { BLIT(false) }
55 | "PRINT" { PRINT }
56 | "main" { MAIN }
57 | digits as lxm { ILIT(int_of_string lxm) }
58 | ['-']? (digits '.' digit* ( ['e' 'E'] ['+' '-']? digits )?) as lxm { FLIT(lxm) }
59 | '"' ([' ' '! ' # '- '&' ' ('-[' ' ]'- '~ ' 'a'-'z' ' ' 'A'-'Z' '0'-'9']* as lxm) '"' { SLIT(lxm) }
60 | ['a'-'z']['a'-'z' '0'-'9' '_']* as lxm { ID(lxm) }
61 | ['A'-'Z']['A'-'Z' '0'-'9' '_']* as actionID { ACTIONID(actionID) }
62 | eof { EOF }
63 | _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }
64
65 and comment = parse
66   "*/" { CEND }
67 | "\n" { token lexbuf }
68 | _ { comment lexbuf }

```

parse.mly

```

1  %{
2  open Ast
3  %}
4
5  %token LPAREN RPAREN LBRACE RBRACE LSQUARE RSQUARE SERIESSIZE SERIESPUSH SERIESPOP SERIES CARD PLAYER COLON SEMI COMMA PLUS
6  %token NOT EQ NEQ LT LEQ GT GEQ AND OR
7  %token RETURN IF ELSE FOR WHILE INT BOOL FLOAT VOID STRING WHEN DO MAIN PRINT
8  %token CEND
9  %token <int> ILIT
10 %token <bool> BLIT
11 %token <string> ID ACTIONID FLIT SLIT
12 %token EOF
13
14 %start program
15 %type <Ast.program> program
16
17 %nonassoc FOR
18 %right ASSIGN
19 %left OR
20 %left AND
21 %left EQ NEQ
22 %left LT GT LEQ GEQ
23 %nonassoc DO
24 %nonassoc COLON
25 %left PLUS MINUS
26 %left MULT DIVIDE
27 %right NOT
28
29 %%
30
31 program:

```

```

32     decls main_decl EOF      { (fst $1, snd $1, $2) }
33
34 decls:
35   | /*nothing*/             { ([], []) }
36   | decls global_decl      { (List.rev ($2::fst $1), snd $1) }
37   | decls action_decl      { (fst $1, List.rev ($2::snd $1)) }
38
39 cend_opt: /*comment */
40   | /*nothing */           { Noexpr }
41   | CEND                   { Noexpr }
42
43 global_decl:
44   typ ID cend_opt NEWLINE { ($1, $2)}
45
46 main_decl:
47   MAIN COLON cend_opt NEWLINE LBRACE cend_opt NEWLINE locals_list stmt_wrap RBRACE cend_opt NEWLINE {{
48   mtyp = Void;
49   mparams = [];
50   mlocals = $8;
51   mbody = [$9] }}
52
53 action_decl:
54   WHEN DO typ ACTIONID LPAREN params_list_opt RPAREN COLON cend_opt NEWLINE LBRACE cend_opt NEWLINE locals_list stmt_wrap
55   {{
56     atyp = $3;
57     aname = $4;
58     aparams = List.rev $6;
59     alocals = List.rev $14;
60     abody = [$15] }}
61
62 params_list_opt:
63   | params_list      {$1}
64   | /*Nothing*/     {[]}
65
66 params_list:
67   | param                { [$1] }
68   | params_list COMMA param { $3::$1 }
69
70 param:
71   | typ ID                { $1, $2 }
72
73 stmt_block:
74   | NEWLINE LBRACE cend_opt NEWLINE stmt_list RBRACE cend_opt NEWLINE { Block(List.rev $5) }
75
76 locals_list:
77   | /*nothing */         {[] }
78   | locals_list global_decl { $2 :: $1 }
79
80 stmt_wrap:
81   | /*nothing */         { Block([]) }
82   | stmt_list            { Block(List.rev $1) }
83

```

```

84  stmt_list:
85      | stmt                                { [$1]      }
86      | stmt_list stmt                      { $2 :: $1 }
87
88  stmt:
89      | stmt_block                          { $1          }
90      | expr cend_opt NEWLINE               { Expr $1     }
91      | RETURN expr_opt NEWLINE            { Return $2   }
92      | if_stmt                             { $1          }
93      | FOR LPAREN expr SEMI expr SEMI expr RPAREN COLON cend_opt stmt_block { For($3, $5, $7, $11) }
94      | WHILE expr COLON cend_opt stmt_block { While($2, $5) }
95      | ID DOT SERIESPUSH LPAREN expr RPAREN cend_opt NEWLINE { SeriesPush($1, $5) }
96
97  if_stmt:
98      | IF expr COLON cend_opt stmt_block else_block_opt { If($2, $5, $6) }
99
100 else_block_opt:
101     | /* nothing */ { Block([]) }
102     | else_block   { $1      }
103
104 else_block:
105     | ELSE COLON cend_opt stmt_block { $4 }
106
107 typ:
108     | INT          { Int      }
109     | BOOL         { Bool     }
110     | FLOAT        { Float    }
111     | STRING       { String   }
112     | VOID         { Void     }
113     | SERIES LT typ GT { Series($3) }
114     | PLAYER       { Player   }
115     | CARD         { Card     }
116
117 expr:
118     | ILIT         { Iliteral($1) }
119     | FLIT         { Fliteral($1) }
120     | BLIT         { Bliteral($1) }
121     | SLIT         { Sliteral($1) }
122     | ID           { Id($1) }
123     | ID ASSIGN expr { Assign($1, $3) }
124     | LSQUARE args_list_opt RSQUARE { Seriesliteral($2) }
125     | ID LSQUARE expr RSQUARE { SeriesGet($1, $3) }
126     | ID DOT SERIESIZE LPAREN RPAREN { SeriesSize($1)}
127     | ID DOT SERIESPOP LPAREN RPAREN { SeriesPop($1)}
128     | CEND         { Noexpr }
129     | expr PLUS expr { Binop($1, Add, $3) }
130     | expr MINUS expr { Binop($1, Sub, $3) }
131     | expr MULT expr { Binop($1, Mult, $3) }
132     | expr DIVIDE expr { Binop($1, Div, $3) }
133     | expr AND expr { Binop($1, And, $3) }
134     | expr OR expr { Binop($1, Or, $3) }
135     | expr EQ expr { Binop($1, Equal, $3) }

```

```

136 | expr NEQ expr { Binop($1, Neq, $3) }
137 | expr LT expr { Binop($1, Less, $3) }
138 | expr LEQ expr { Binop($1, Leq, $3) }
139 | expr GT expr { Binop($1, Greater, $3) }
140 | expr GEQ expr { Binop($1, Geq, $3) }
141 | MINUS expr %prec NOT { Unop(Neg, $2) }
142 | NOT expr { Unop(Not, $2) }
143 | call_print { $1 }
144 | call_class { $1 }
145 | call_action { $1 }
146 | call_attr { $1 }
147
148 args_list_opt:
149 | /*nothing */ { [] }
150 | args_list { List.rev $1 }
151
152 args_list:
153 | expr { [$1] }
154 | args_list COMMA expr { $3 :: $1 }
155
156 call_print:
157 | DO PRINT LPAREN expr RPAREN { PrintCall($4) }
158
159 call_action:
160 | DO ACTIONID LPAREN args_list_opt RPAREN { ActionCall($2, $4) }
161
162 call_class:
163 | PLAYER LPAREN args_list_opt RPAREN { PlayerClassCall($3) }
164 | CARD LPAREN args_list_opt RPAREN { CardClassCall($3 ) }
165
166 call_attr:
167 | ID DOT ID { AttrCall($1, $3) }
168
169 expr_opt:
170 | /* nothing */ { Noexpr }
171 | expr { $1 }

```

ast.ml

```

1 type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq |
2     And | Or
3
4 type uop = Neg | Not
5
6 type typ = Int | Float | Bool | String | Void | Series of typ | Player | Card
7
8 type bind = typ * string
9
10
11 type expr =
12 | Iliteral of int

```



```

13 | Fliteral of string
14 | Bliteral of bool
15 | Sliteral of string
16 | Seriesliteral of expr list
17 | PrintCall of expr
18 | ActionCall of string * expr list
19 | Id of string
20 | Assign of string * expr
21 (* | AttrAssign of string * string * expr *)
22 | Binop of expr * op * expr
23 | Unop of uop * expr
24 | PlayerClassCall of expr list
25 | CardClassCall of expr list
26 | AttrCall of string * string
27 | SeriesGet of string * expr
28 | SeriesSize of string
29 | SeriesPop of string
30 | Noexpr
31
32
33 type stmt =
34 | Block of stmt list
35 | Expr of expr
36 | Return of expr
37 | If of expr * stmt * stmt
38 | For of expr * expr * expr * stmt
39 (* | ForLit of string * expr * stmt *)
40 | While of expr * stmt
41 | SeriesPush of string * expr
42 | Nostmt
43
44 type main_decl = {
45   mtyp : typ;
46   mparams : bind list;
47   mlocals : bind list;
48   mbody: stmt list;
49 }
50
51 type action_decl = {
52   (* entitytyp : typ;
53   entityid : string; *)
54   atyp : typ;
55   aname : string;
56   aparams : bind list;
57   alocals : bind list;
58   abody: stmt list;
59 }
60
61 type program = bind list * action_decl list * main_decl
62
63 (* Pretty-printing functions *)
64 let string_of_op = function

```

```

65     Add -> "+"
66 | Sub -> "-"
67 | Mult -> "*"
68 | Div -> "/"
69 | Equal -> "=="
70 | Neq -> "!="
71 | Less -> "<"
72 | Leq -> "<="
73 | Greater -> ">"
74 | Geq -> ">="
75 | And -> "and"
76 | Or -> "or"
77
78 let string_of_uop = function
79     Neg -> "-"
80 | Not -> "!"
81
82 let rec string_of_typ = function
83     Int -> "int"
84 | Bool -> "bool"
85 | Float -> "float"
86 | String -> "string"
87 | Void -> "void"
88 | Series x -> "series<" ^ (string_of_typ x) ^ ">"
89 | Player -> "player"
90 | Card -> "card"
91
92 let rec string_of_expr = function
93     Iliteral(l) -> string_of_int l
94 | Fliteral(l) -> l
95 | Bliteral(true) -> "true"
96 | Bliteral(false) -> "false"
97 | Sliteral(l) -> l
98 | Seriesliteral(_) -> "series_literal"
99 | PrintCall(e) -> "do" ^ "PRINT" ^ "(" ^ string_of_expr e ^ ")"
100 | ActionCall(f, el) ->
101     "do " ^ f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
102 | PlayerClassCall(el) ->
103     "Player (" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
104 | CardClassCall(el) ->
105     "Card (" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
106 | AttrCall(cls, fld) -> cls ^ "." ^ fld
107 | Id(s) -> s
108 | Assign(v, e) -> v ^ " = " ^ string_of_expr e
109 (* | AttrAssign(s, v, e) -> s ^ "." ^ v ^ " = " ^ string_of_expr e *)
110 | Binop(e1, o, e2) ->
111     string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
112 | Unop(o, e) -> string_of_uop o ^ string_of_expr e
113 | SeriesGet(id, e) -> id ^ "[" ^ (string_of_expr e) ^ "]"
114 | SeriesSize(id) -> "series_size " ^ id
115 | SeriesPop(id) -> "series_pop " ^ id
116 | Noexpr -> ""

```

```

117
118 let rec string_of_stmt = function
119   Block(stmts) -> "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "\n}"
120   | Expr(exp) -> string_of_expr exp ^ "\n"
121   | Return(exp) -> "return" ^ string_of_expr exp ^ "\n"
122   | If(exp, s1, s2) -> "if " ^ string_of_expr exp ^ ":\n" ^ string_of_stmt s1 ^ string_of_stmt s2
123   | For(e1, e2, e3, s) -> "for (" ^ string_of_expr e1 ^ ";" ^ string_of_expr e2 ^ ";" ^
124     string_of_expr e3 ^ "):\n" ^ string_of_stmt s
125   | While(exp, stmt) -> "while " ^ string_of_expr exp ^ ":\n" ^ string_of_stmt stmt
126   | SeriesPush(id, exp) -> id ^ "." ^ "push" ^ "(" ^ string_of_expr exp ^ ")"
127   | Nostmt -> ""

```

sast.ml

```

1  open Ast
2
3  type sexpr = typ * sx
4
5  and sx =
6    | SLiteral of int
7    | SFliteral of string
8    | SBliteral of bool
9    | SSLiteral of string
10   | SSeriesliteral of typ * sexpr list
11   | SSeriesGet of typ * string * sexpr
12   | SSeriesSize of typ * string
13   | SSeriesPop of typ * string
14   | SPrintCall of sexpr
15   | SActionCall of string * sexpr list
16   | SId of string
17   | SAssign of string * sexpr
18   (* | SAttrAssign of string * string * sexpr *)
19   | SBinop of sexpr * op * sexpr
20   | SUnop of uop * sexpr
21   | SPlayerClassCall of sexpr list
22   | SCardClassCall of sexpr list
23   | SAttrCall of string * string
24   | SNoexpr
25
26  type sstmt =
27    | SBlock of sstmt list
28    | SExpr of sexpr
29    | SReturn of sexpr
30    | SIf of sexpr * sstmt * sstmt
31    | SFor of sexpr * sexpr * sexpr * sstmt
32    (* | SForLit of string * sexpr * sstmt *)
33    | SWhile of sexpr * sstmt
34    | SSeriesPush of string * sexpr
35    | SNostmt
36
37  type smain_decl = {

```

```

38   smtyp : typ;
39   smparams : bind list;
40   smlocals : bind list;
41   smbody: sstmt list;
42 }
43
44 type saction_decl = {
45   (* sentitytyp : typ;
46   sentityid : string; *)
47   saname : string;
48   satyp : typ;
49   saparams : bind list;
50   salocals : bind list;
51   sabody: sstmt list;
52 }
53
54 type sprogram = bind list * saction_decl list * smain_decl
55
56 (* Pretty-printing functions *)
57
58 let rec string_of_sexpr (t, e) =
59   "(" ^ string_of_typ t ^ " : " ^ (match e with
60     SIliteral(l) -> string_of_int l
61   | SFliteral(l) -> l
62   | SBliteral(true) -> "true"
63   | SBliteral(false) -> "false"
64   | SSliteral(l) -> l
65   | SSeriesliteral(_) -> "list_literal"
66   | SSeriesGet(_, id, e) -> id ^ "[" ^ (string_of_sexpr e) ^ "]"
67   | SSeriesSize(_, id) -> "series_size " ^ id
68   | SSeriesPop(_, id) -> "series_pop " ^ id
69   | SPrintCall(e) -> "do" ^ "PRINT" ^ "(" ^ string_of_sexpr e ^ ")"
70   | SActionCall(f, el) ->
71     "do " ^ f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
72   | SPlayerClassCall(e1) ->
73     "Player (" ^ String.concat ", " (List.map string_of_sexpr e1) ^ ")"
74   | SCardClassCall(e1) ->
75     "Card (" ^ String.concat ", " (List.map string_of_sexpr e1) ^ ")"
76   | SAttrCall(cls, fld) -> cls ^ "." ^ fld
77   | SId(s) -> s
78   | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
79   (* | SAttrAssign(s, v, e) -> s ^ "." ^ v ^ " = " ^ string_of_sexpr e *)
80   | SBinop(e1, o, e2) ->
81     string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
82   | SUNop(o, e) -> string_of_uop o ^ string_of_sexpr e
83   | SNoexpr -> ""
84
85 let rec string_of_sstmt = function
86   SBlock(stmts) -> "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "\n}"
87   | SExpr(exp) -> string_of_sexpr exp ^ "\n"
88   | SReturn(exp) -> "return" ^ string_of_sexpr exp ^ "\n"
89   | SIf(exp, s1, s2) -> "if " ^ string_of_sexpr exp ^ ":\n" ^ string_of_sstmt s1 ^ string_of_sstmt s2

```

```

90 | SFor(e1, e2, e3, s) -> "for (" ^ string_of_sexpr e1 ^ ";" ^ string_of_sexpr e2 ^ ";" ^
91     string_of_sexpr e3 ^ "):\n" ^ string_of_sstmt s
92 | SWhile(exp, stmt) -> "while " ^ string_of_sexpr exp ^ ":\n" ^ string_of_sstmt stmt
93 | SSeriesPush(id, exp) -> id ^ "." ^ "push" ^ "(" ^ string_of_sexpr exp ^ ")"
94 | SNostmt -> ""

```

semant.ml

```

1  open Ast
2  open Sast
3
4  module StringMap = Map.Make(String)
5
6  let check (globals, action_decls, main_decl) =
7
8      (* Verify a list of bindings has no Void types or duplicate names *)
9      let check_binds (kind : string) (binds : bind list) =
10         List.iter (function
11             | (Void, b) -> raise (Failure ("illegal Void " ^ kind ^ " " ^ b))
12             | _ -> ()) binds;
13         let rec dups = function
14             [] -> ()
15             | ((_,n1) :: (_,n2) :: _) when n1 = n2 -> raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
16             | _ :: t -> dups t
17         in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
18     in
19
20     (**** Check global variables ****)
21     check_binds "global" globals;
22
23     let built_in_decls =
24         let add_bind map (name, ty) = StringMap.add name {
25             atyp = Void;
26             aname = name;
27             aparams = [(ty, "x")];
28             alocals = []; abody = [] } map
29         in List.fold_left add_bind StringMap.empty []
30     in
31
32     let add_action map ad =
33         let built_in_err = "function " ^ ad.aname ^ " may not be defined"
34         and dup_err = "duplicate function " ^ ad.aname
35         and make_err er = raise (Failure er)
36         and n = ad.aname (* Name of the function *)
37         in match ad with (* No duplicate functions or redefinitions of built-ins *)
38             _ when StringMap.mem n built_in_decls -> make_err built_in_err
39             | _ when StringMap.mem n map -> make_err dup_err
40             | _ -> StringMap.add n ad map
41     in
42
43     (* Collect all function names into one symbol table *)

```

```

44 let action_decls_map = List.fold_left add_action built_in_decls action_decls
45 in
46
47 let find_act s =
48   try StringMap.find s action_decls_map
49   with Not_found -> raise (Failure ("unrecognized action " ^ s))
50 in
51
52 let check_main main =
53   (* Make sure no params or locals are void or duplicates *)
54   check_binds "mlocals" main.mlocals;
55
56   (* Raise an exception if the given rvalue type cannot be assigned to
57   the given lvalue type *)
58   let check_assign lvaluet rvaluet err =
59     if lvaluet = rvaluet then lvaluet else raise (Failure err)
60   in
61
62   (* Build local symbol table of variables for this action *)
63   let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
64     StringMap.empty (globals @ main.mlocals)
65   in
66
67   (* Return a variable from our local symbol table *)
68   let type_of_identifier s =
69     try StringMap.find s symbols
70     with Not_found -> raise (Failure ("undeclared identifier " ^ s))
71   in
72
73   let check_series_type id =
74     match (type_of_identifier id) with
75     | Series t -> t
76     | t -> raise (Failure ("check series type error, typ: " ^ string_of_typ t))
77   in
78
79   let rec check_expr = function
80     (*need to figure out typ, if name is defined*)
81     | PlayerClassCall(pparams) as pcall ->
82       let constructor_len = 2 in
83       if List.length pparams != constructor_len then
84         raise (Failure ("expecting " ^ string_of_int constructor_len ^
85           " arguments in " ^ string_of_expr pcall))
86       else (Player, SPlayerClassCall(List.map check_expr pparams))
87     | CardClassCall(pparams) as ccall ->
88       let constructor_len = 3 in
89       if List.length pparams != constructor_len then
90         raise (Failure ("expecting " ^ string_of_int constructor_len ^
91           " arguments in " ^ string_of_expr ccall))
92       else (Card, SCardClassCall(List.map check_expr pparams))
93     (* | AttrAssign(objname, attr, e) -> (Void, SAttrAssign(objname, attr, check_expr e)) *)
94     | AttrCall(objname, attr) ->
95       (match attr with

```

```

96     "name" -> (String, SAttrCall(objname, attr))
97   | "score" -> (Int, SAttrCall(objname, attr))
98   | "type" -> (String, SAttrCall(objname, attr))
99   | "faceup" -> (Bool, SAttrCall(objname, attr))
100  | "value" -> (Int, SAttrCall(objname, attr))
101  | _ -> raise (Failure ("attribute not found"))
102 | Sliteral s -> (String, SSLiteral(s))
103 | Iliteral i -> (Int, SILiteral(i))
104 | Fliteral f -> (Float, SFliteral(f))
105 | Bliteral b -> (Bool, SBliteral(b))
106 | Noexpr      -> (Void, SNoexpr)
107 | Id s        -> (type_of_identifier s, SID s)
108 | Assign(var, e) ->
109   let lt = type_of_identifier var
110   and (rt, e') = check_expr e in
111   let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
112           string_of_typ rt
113   in (check_assign lt rt err, SAssign(var, (rt, e')))
114 | Unop(op, e) as ex ->
115   let (t, e') = check_expr e in
116   let ty = match op with
117     Neg when t = Int || t = Float -> t
118   | Not when t = Bool -> Bool
119   | _ -> raise (Failure ("illegal unary operator " ^
120                       string_of_uop op ^ string_of_typ t ^
121                       " in " ^ string_of_expr ex))
122   in (ty, SUnop(op, (t, e')))
123 | Binop(e1, op, e2) ->
124   let (t1, e1') = check_expr e1
125   and (t2, e2') = check_expr e2 in
126   (* All binary operators require operands of the same type *)
127   let same = t1 = t2 in
128   (* Determine expression type based on operator and operand types *)
129   let ty = match op with
130     Add | Sub | Mult | Div when same && t1 = Int -> Int
131   | Add | Sub | Mult | Div when same && t1 = Float -> Float
132   | Equal | Neq          when same -> Bool
133   | Less | Leq | Greater | Geq
134     when same && (t1 = Int || t1 = Float) -> Bool
135   | And | Or when same && t1 = Bool -> Bool
136   | _ -> raise (
137     Failure ("illegal binary operator of " ^ string_of_op op ^ "for " ^ string_of_typ t1 ^ " and " ^ string_of_t
138     in (ty, SBinop((t1, e1'), op, (t2, e2'))))
139 | PrintCall(e) -> (Void, SPrintCall(check_expr e))
140 | ActionCall(aname, args) as acall ->
141   let ad = find_act aname in
142   let param_length = List.length ad.aparams in
143   if List.length args != param_length then
144     raise (Failure ("expecting " ^ string_of_int param_length ^
145                   " arguments in " ^ string_of_expr acall))
146   else let check_call (at, _) e =
147         let (et, e') = check_expr e in

```

```

148     let err = "illegal argument found " ^ string_of_ttyp et ^
149         " expected " ^ string_of_ttyp at ^ " in " ^ string_of_expr e
150     in (check_assign at et err, e')
151 in
152 let args' = List.map2 check_call ad.aparams args
153 in (ad.atyp, SActionCall(aname, args'))
154 | Seriesliteral vals ->
155     let (t', _) = check_expr (List.hd vals) in
156     let map_func lit = check_expr lit in
157     let vals' = List.map map_func vals in
158     (Series t', SSeriesliteral(t', vals'))
159 | SeriesGet (var, e) ->
160     let (t, e') = check_expr e in
161     let ty = match t with
162         Int -> Int
163         | _ -> raise (Failure ("list_get index must be integer, not " ^ string_of_ttyp t))
164     in let list_type = check_series_type var
165     in (list_type, SSeriesGet(list_type, var, (ty, e')))
166 | SeriesSize var ->
167     (Int, SSeriesSize(check_series_type var, var))
168 | SeriesPop var ->
169     let series_type = check_series_type var
170     in (series_type, SSeriesPop(series_type, var))
171 in
172
173 let check_match_series_type_expr l e =
174     let (t', e') as e'' = check_expr e
175     in let err = "series type and expression type do not match " ^ (string_of_ttyp t') ^ ", " ^ (string_of_sexpr e'')
176     in if t' != (check_series_type l) then raise (Failure err) else (t', e')
177 in
178
179 let check_bool_expr e =
180     let (t', e') = check_expr e
181     and err = "expected Boolean expression in " ^ string_of_expr e
182     in if t' != Bool then raise (Failure err) else (t', e')
183 in
184
185 let rec check_stmt = function (*currently only supports one input -- support for map *)
186     Expr e -> SExpr (check_expr e)
187     | SeriesPush (var, e) ->
188         let _ = check_series_type var in
189         SSeriesPush(var, check_match_series_type_expr var e)
190     | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt b2)
191     | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
192     | For(e1, e2, e3, st) ->
193         SFor(check_expr e1, check_bool_expr e2, check_expr e3, check_stmt st)
194     | Return e -> let (t, e') = check_expr e in
195         if t = main.mtyp then SReturn (t, e')
196         else raise (
197             Failure ("Main does not support return of " ^ string_of_ttyp t ^ " expected " ^
198                 string_of_ttyp main.mtyp ^ " in " ^ string_of_expr e))
199     | Block sl ->

```



```

200     let rec check_stmt_list = function
201       [Return _ as s] -> [check_stmt s]
202     | Return _ :: _ -> raise (Failure "nothing may follow a return")
203     | Block sl :: ss -> check_stmt_list (sl @ ss) (* Flatten blocks *)
204     | s :: ss       -> check_stmt s :: check_stmt_list ss
205     | []            -> []
206     in SBBlock(check_stmt_list sl) (*(List.map check_stmt_list sl)*)
207   | Nostmt -> SNostmt
208   in
209   {
210     smtyp = main.mtyp;
211     smparams = main.mparams;
212     smlocals = main.mlocals;
213     smbody = match check_stmt (Block main.mbody) with
214     SBBlock(sl) -> sl
215     | _ -> raise (Failure ("internal error: block didn't become a block?"))
216   }
217   in
218   let check_action act =
219     (* Make sure no params or locals are void or duplicates *)
220     check_binds "aparams" act.aparams;
221     check_binds "alocals" act.alocals;
222
223     (* Raise an exception if the given rvalue type cannot be assigned to
224     the given lvalue type *)
225     let check_assign lvaluet rvaluet err =
226       if lvaluet = rvaluet then lvaluet else raise (Failure err)
227     in
228
229     (* Build local symbol table of variables for this action *)
230     let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
231       StringMap.empty (globals @ act.aparams @ act.alocals )
232     in
233
234     (* Return a variable from our local symbol table *)
235     let type_of_identifier s =
236       try StringMap.find s symbols
237       with Not_found -> raise (Failure ("undeclared identifier " ^ s))
238     in
239
240     (* Check if id is a series and return series type *)
241     let check_series_type id =
242       match (type_of_identifier id) with
243       Series t -> t
244       | t -> raise (Failure ("check series type error, typ: " ^ string_of_typ t))
245     in
246
247     let rec check_expr = function
248     | PlayerClassCall(pparams) -> (Player, SPlayerClassCall(List.map check_expr pparams))
249     | CardClassCall(pparams) -> (Card, SCardClassCall(List.map check_expr pparams))
250     | AttrCall(objname, attr) ->
251       (match attr with

```

```

252     "name" -> (String, SAttrCall(objname, attr))
253 | "score" -> (Int, SAttrCall(objname, attr))
254 | "type" -> (String, SAttrCall(objname, attr))
255 | "faceup" -> (Bool, SAttrCall(objname, attr))
256 | "value" -> (Int, SAttrCall(objname, attr))
257 | _ -> raise (Failure ("attribute not found"))
258 (* | AttrAssign(objname, attr, e) -> (Void, SAttrAssign(objname, attr, check_expr e)) *)
259     | Sliteral s -> (String, SSLiteral(s))
260     | Iliteral i -> (Int, SILiteral(i))
261     | Fliteral f -> (Float, SFliteral(f))
262 | Bliteral b -> (Bool, SBliteral(b))
263 | Noexpr    -> (Void, SNoexpr)
264 | Id s      -> (type_of_identifier s, SId s)
265 | Assign(var, e) ->
266     let lt = type_of_identifier var
267     and (rt, e') = check_expr e in
268     let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
269             string_of_typ rt
270     in (check_assign lt rt err, SAssign(var, (rt, e')))
271 | Unop(op, e) as ex ->
272     let (t, e') = check_expr e in
273     let ty = match op with
274         Neg when t = Int || t = Float -> t
275     | Not when t = Bool -> Bool
276     | _ -> raise (Failure ("illegal unary operator " ^
277                         string_of_uop op ^ string_of_typ t ^
278                         " in " ^ string_of_expr ex))
279     in (ty, SUnop(op, (t, e')))
280 | Binop(e1, op, e2) ->
281     let (t1, e1') = check_expr e1
282     and (t2, e2') = check_expr e2 in
283     let same = t1 = t2 in
284     let ty = match op with
285         Add | Sub | Mult | Div when same && t1 = Int    -> Int
286     | Add | Sub | Mult | Div when same && t1 = Float    -> Float
287     | Equal | Neq           when same                   -> Bool
288     | Less | Leq | Greater | Geq
289         when same && (t1 = Int || t1 = Float) -> Bool
290     | And | Or when same && t1 = Bool -> Bool
291     | _ -> raise (
292         Failure ("illegal binary operator of " ^ string_of_op op ^ "for " ^ string_of_typ t1 ^ " and " ^ string_of_typ
293         in (ty, SBinop((t1, e1'), op, (t2, e2')))
294 | PrintCall(e) -> (Void, SPrintCall(check_expr e))
295 | ActionCall(aname, args) as acall ->
296     let ad = find_act aname in
297     let param_length = List.length ad.aparams in
298     if List.length args != param_length then
299         raise (Failure ("expecting " ^ string_of_int param_length ^
300                         " arguments in " ^ string_of_expr acall))
301     else let check_call (at, _) e =
302         let (et, e') = check_expr e in
303         let err = "illegal argument found " ^ string_of_typ et ^

```

```

304         " expected " ^ string_of_typ at ^ " in " ^ string_of_expr e
305     in (check_assign at et err, e')
306 in
307 let args' = List.map2 check_call ad.aparams args
308 in (ad.atyp, SActionCall(aname, args'))
309 | Seriesliteral vals ->
310     let (t', _) = check_expr (List.hd vals) in
311     let map_func lit = check_expr lit in
312     let vals' = List.map map_func vals in
313     (Series t', SSeriesliteral(t', vals'))
314 | SeriesGet (var, e) ->
315     let (t, e') = check_expr e in
316     let ty = match t with
317         Int -> Int
318         | _ -> raise (Failure ("list_get index must be integer, not " ^ string_of_typ t))
319     in let list_type = check_series_type var
320     in (list_type, SSeriesGet(list_type, var, (ty, e')))
321 | SeriesSize var ->
322     (Int, SSeriesSize(check_series_type var, var))
323 | SeriesPop var ->
324     let series_type = check_series_type var
325     in (series_type, SSeriesPop(series_type, var))
326 in
327
328 let check_match_series_type_expr l e =
329     let (t', e') as e'' = check_expr e
330     in let err = "series type and expression type do not match " ^ (string_of_typ t') ^ ", " ^ (string_of_sexpr e'')
331     in if t' != (check_series_type l) then raise (Failure err) else (t', e')
332 in
333
334 let check_bool_expr e =
335     let (t', e') = check_expr e
336     and err = "expected Boolean expression in " ^ string_of_expr e
337     in if t' != Bool then raise (Failure err) else (t', e')
338 in
339
340 let rec check_stmt = function (*currently only supports one input -- support for map *)
341 Expr e -> SExpr (check_expr e)
342 | SeriesPush (var, e) ->
343     let _ = check_series_type var in
344     SSeriesPush(var, check_match_series_type_expr var e)
345 | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt b2)
346 | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
347 | For(e1, e2, e3, st) ->
348     SFor(check_expr e1, check_bool_expr e2, check_expr e3, check_stmt st)
349 | Return e -> let (t, e') = check_expr e in
350     if t = act.atyp then SReturn (t, e')
351     else raise (
352         Failure ("return gives " ^ string_of_typ t ^ " expected " ^
353             string_of_typ act.atyp ^ " in " ^ string_of_expr e))
354 | Block s1 ->
355     let rec check_stmt_list = function

```

```

356     [Return _ as s] -> [check_stmt s]
357   | Return _ :: _   -> raise (Failure "nothing may follow a return")
358   | Block s1 :: ss -> check_stmt_list (s1 @ ss) (* Flatten blocks *)
359   | s :: ss        -> check_stmt s :: check_stmt_list ss
360   | []             -> []
361   in SBlock(check_stmt_list s1) (*(List.map check_stmt_list s1)*)
362 | Nostmt -> SNostmt
363 in
364 {
365   saname = act.aname;
366   satyp  = act.atyp;
367   saparams = act.aparams;
368   salocals = act.alocals;
369   sabody = match check_stmt (Block act.abody) with
370   SBlock(s1) -> s1
371   | _ -> raise (Failure ("internal error: block didn't become a block?"))
372 }
373
374 in (globals, List.map check_action action_decls, check_main main_decl)

```

codegen.ml

```

1  module L = Llvml
2  module A = Ast
3  open Sast
4
5  module StringMap = Map.Make(String)
6
7  let translate (globals, action_decls, main_decl) =
8      let context = L.global_context () in
9
10     let the_module = L.create_module context "AHOD" in
11
12     let str_format_str = L.define_global "str" (L.const_stringz context "%s\n") the_module
13     and int_format_str = L.define_global "str" (L.const_stringz context "%d\n") the_module
14     and float_format_str = L.define_global "str" (L.const_stringz context "%g\n") the_module
15     and bool_format_str = L.define_global "str" (L.const_stringz context "%d\n") the_module in
16
17     let i32_t = L.i32_type context
18     and i8_t = L.i8_type context
19     and i1_t = L.i1_type context
20     and float_t = L.double_type context
21     and string_t = L.pointer_type (L.i8_type context)
22     and void_t = L.void_type context
23     and series_t t = L.struct_type context [| L.pointer_type (L.i32_type context); (L.pointer_type t) |]
24     and player_t = L.struct_type context [| (L.pointer_type (L.i8_type context)); (L.i32_type context) |]
25     and card_t = L.struct_type context [| (L.pointer_type (L.i8_type context)); (L.i1_type context); (L.i32_type context) |]
26     (*struct_set_body class_t *)
27     in
28
29     let rec ltype_of_ttyp = function

```

```

30 | A.Int    -> i32_t
31 | A.Bool   -> i1_t
32 | A.Float  -> float_t
33 | A.String -> string_t
34 | A.Void   -> void_t
35 | A.Series t -> series_t (ltype_of_typ t)
36 | A.Player -> player_t
37 | A.Card   -> card_t
38 in
39
40 let type_str t = match t with
41 | A.Int    -> "int"
42 | A.Bool   -> "bool"
43 | A.Float  -> "float"
44 | A.String -> "str"
45 | A.Card   -> "card"
46 | A.Player -> "player"
47 | _       -> raise (Failure "Invalid string map key type")
48 in
49
50 let global_vars : L.llvalue StringMap.t = (* type: L.llvalue StringMap.t *)
51 let global_var m (t, n) = (*t: type, n:name*)
52   let init = match t with
53     | A.Float  -> L.const_float (ltype_of_typ t) 0.0
54     | A.String -> L.const_pointer_null (ltype_of_typ t)
55     | A.Series series_type -> L.const_struct context ([| L.const_pointer_null (L.pointer_type(L.i32_type context)); L.con
56     (* ===== initialized the class ===== *)
57     | A.Player -> L.const_struct context ([|L.const_pointer_null (L.pointer_type(L.i8_type context)) ; L.const_pointer_nu
58     | A.Card -> L.const_struct context ([|L.const_pointer_null (L.pointer_type(L.i8_type context)) ; L.const_pointer_nu
59     (* ===== *)
60     | _ -> L.const_int (ltype_of_typ t) 0
61   in StringMap.add n (L.define_global n init the_module) m in
62 List.fold_left global_var StringMap.empty globals in
63
64   let printf_t : L.lltype =
65     L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
66   let printf_func : L.llvalue =
67     L.declare_function "printf" printf_t the_module in
68
69   (* ----- *)
70 let playercall_t : L.lltype =
71   L.function_type player_t [| string_t ; i32_t |] in
72 let playercall_func : L.llvalue =
73   L.declare_function "playercall" playercall_t the_module in
74
75 let getplayername_t : L.lltype =
76   L.function_type string_t [| player_t |] in
77 let getplayername_func : L.llvalue =
78   L.declare_function "getplayername" getplayername_t the_module in
79
80 let getplayerscore_t : L.lltype =
81   L.function_type i32_t [| player_t |] in

```

```

82 let getplayerscore_func : L.llvalue =
83     L.declare_function "getplayerscore" getplayerscore_t the_module in
84
85     (* let setplayername_t : L.lltype =
86         L.function_type string_t [| player_t ; string_t |] in
87     let setplayername_func : L.llvalue =
88         L.declare_function "setplayername" setplayername_t the_module in
89
90     let setplayerscore_t : L.lltype =
91         L.function_type i32_t [| player_t ; i32_t |] in
92     let setplayerscore_func : L.llvalue =
93         L.declare_function "setplayerscore" setplayerscore_t the_module in *)
94
95     (* ----- *)
96
97     let cardcall_t : L.lltype =
98         L.function_type card_t [| string_t ; i1_t ; i32_t |] in
99     let cardcall_func : L.llvalue =
100         L.declare_function "cardcall" cardcall_t the_module in
101
102     let getcardtype_t : L.lltype =
103         L.function_type string_t [| card_t |] in
104     let getcardtype_func : L.llvalue =
105         L.declare_function "getcardtype" getcardtype_t the_module in
106
107     let getcardfaceup_t : L.lltype =
108         L.function_type i1_t [| card_t |] in
109     let getcardfaceup_func : L.llvalue =
110         L.declare_function "getcardfaceup" getcardfaceup_t the_module in
111     let getcardvalue_t : L.lltype =
112         L.function_type i32_t [| card_t |] in
113     let getcardvalue_func : L.llvalue =
114         L.declare_function "getcardvalue" getcardvalue_t the_module in
115
116     (* ----- *)
117
118     (*series generation*)
119     let init_series builder series_ptr series_type =
120         (* size to 0 *)
121         let sizePtrPtr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
122         let sizePtr = L.build_alloca i32_t "series_size" builder in
123         let _ = L.build_store (L.const_int i32_t 0) sizePtr builder in
124         ignore(L.build_store sizePtr sizePtrPtr builder);
125         (* init series *)
126         let series_array_ptr = L.build_struct_gep series_ptr 1 "series.array" builder in
127         (* ERROR: when there's nothing so like a = [] *)
128         let p = L.build_array_alloca (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
129         ignore(L.build_store p series_array_ptr builder);
130     in
131
132     (*----- define actions -----*)
133     let action_decls_map : (L.llvalue * saction_decl) StringMap.t =

```

```

134   let action_decl m adecl =
135     let name = adecl.saname
136       and param_types = Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) adecl.saparams) in
137     let atype = L.function_type (ltype_of_typ adecl.satyp) param_types in
138     StringMap.add name (L.define_function name atype the_module, adecl) m in
139 List.fold_left action_decl StringMap.empty action_decls in
140
141 (*~~~~~ action generation top-level ~~~~~*)
142 let build_action_body adecl =
143   let (the_action, _) = StringMap.find adecl.saname action_decls_map in
144   let builder = L.builder_at_end context (L.entry_block the_action) in
145
146   let local_vars =
147     let add_param m (t, n) p = L.set_value_name n p;
148     let local = L.build_alloca (ltype_of_typ t) n builder in
149     ignore(
150       match t with
151       | A.Series series_type -> init_series builder local series_type
152       | _ -> ()
153     );
154     ignore (L.build_store p local builder);
155     StringMap.add n local m
156
157     (* Allocate space for any locally declared variables and add the
158      * resulting registers to our map *)
159     and add_local m (t, n) =
160     let local_var = L.build_alloca (ltype_of_typ t) n builder in
161     ignore(
162       match t with
163       | A.Series series_type -> init_series builder local_var series_type
164       | _ -> ()
165     );
166     StringMap.add n local_var m
167   in
168
169   let params = List.fold_left2 add_param StringMap.empty adecl.saparams
170     (Array.to_list (L.params the_action)) in
171   List.fold_left add_local params adecl.salocals
172   in
173
174
175   let lookup n = try StringMap.find n local_vars
176     with Not_found -> StringMap.find n global_vars
177   in
178   (*for the series stuff, referenced past projects + clang *)
179   let init_series builder series_ptr series_type =
180     let size_ptr_ptr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
181     let size_ptr = L.build_alloca i32_t "series_size" builder in
182     ignore(L.build_store (L.const_int i32_t 0) size_ptr builder);
183     ignore(L.build_store size_ptr size_ptr_ptr builder);
184     let series_ptr = L.build_struct_gep series_ptr 1 "series" builder in (* init array (series) *)
185     (* fails when there's []*)

```

```

186     let p = L.build_array_alloc (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
187     ignore(L.build_store p series_ptr builder);
188 in
189
190 let series_push : L.llvalue StringMap.t =
191     let series_push_ty m typ =
192         let series_push_def = L.define_function ("series_push" ^ (type_str typ)) (L.function_type void_t [| L.pointer_type
193             let build = L.builder_at_end context (L.entry_block series_push_def) in
194             let series_ptr = L.build_alloc (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
195             ignore(L.build_store (L.param series_push_def 0) series_ptr build);
196             let valPtr = L.build_alloc (ltype_of_typ typ) "val_alloc" build in
197             ignore(L.build_store (L.param series_push_def 1) valPtr build);
198             let series_load = L.build_load series_ptr "series_load" build in
199             let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
200             let series_load_2 = L.build_load series_ptr_2 "series_load_2" build in
201             let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
202             let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
203             let series_size = L.build_load series_size_ptr "series_size" build in
204             let next_el_ptr = L.build_gep series_load_2 [| series_size |] "series_next_el_ptr" build in
205             let next_size = L.build_add series_size (L.const_int i32_t 1) "next_size" build in
206             ignore(L.build_store next_size series_size_ptr build);
207             ignore(L.build_store (L.build_load valPtr "val" build) next_el_ptr build);
208             ignore(L.build_ret_void build);
209             StringMap.add (type_str typ) series_push_def m in
210             List.fold_left series_push_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in (*change or
211
212 let series_get : L.llvalue StringMap.t =
213     let series_get_ty m typ =
214         let def = L.define_function ("series_get" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type
215             let build = L.builder_at_end context (L.entry_block def) in
216             let series_ptr = L.build_alloc (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
217             ignore(L.build_store (L.param def 0) series_ptr build);
218             let idx_ptr = L.build_alloc i32_t "idx_alloc" build in
219             ignore(L.build_store (L.param def 1) idx_ptr build);
220             let series_load = L.build_load series_ptr "series_load" build in
221             let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
222             let series_load_2 = L.build_load series_ptr_2 "array_load" build in
223             let idx = L.build_load idx_ptr "idx_load" build in
224             let series_el_ptr = L.build_gep series_load_2 [| idx |] "series_el_ptr" build in
225             let element_val = L.build_load series_el_ptr "series_el_ptr" build in
226             ignore(L.build_ret element_val build);
227             StringMap.add (type_str typ) def m in
228             List.fold_left series_get_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
229
230 let series_size : L.llvalue StringMap.t =
231     let series_size_ty m typ =
232         let series_size_def = L.define_function ("series_size" ^ (type_str typ)) (L.function_type i32_t [| L.pointer_type
233             let build = L.builder_at_end context (L.entry_block series_size_def) in
234             let series_ptr = L.build_alloc (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
235             ignore(L.build_store (L.param series_size_def 0) series_ptr build);
236             let series_load = L.build_load series_ptr "series_load" build in
237             let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in

```



```

238     let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
239     let series_size = L.build_load series_size_ptr "series_size" build in
240     ignore(L.build_ret series_size build);
241     StringMap.add (type_str typ) series_size_def m in
242 List.fold_left series_size_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
243
244 let series_pop : L.lvalue StringMap.t =
245 let series_pop_ty m typ =
246 let def = L.define_function ("series_pop" ^ (type_str typ)) (L.function_type (ltype_of_type typ) [| L.pointer_type (series_t (ltype_of_type typ)) |]) in
247 let build = L.builder_at_end context (L.entry_block def) in
248 let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_type typ))) "series_ptr_alloc" build in
249 ignore(L.build_store (L.param def 0) series_ptr build);
250 let series_load = L.build_load series_ptr "series_load" build in
251 let series_arr_ptr = L.build_struct_gep series_load 1 "series_arr_ptr" build in
252 let series_arr_load = L.build_load series_arr_ptr "series_arr_load" build in
253 let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
254 let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
255 let series_size = L.build_load series_size_ptr "series_size" build in
256 let series_sizeMin = L.build_sub series_size (L.const_int i32_t 1) "dec_size" build in
257 let last_el_ptr = L.build_gep series_arr_load [| series_sizeMin |] "series_next_el_ptr" build in
258 let last_el_val = L.build_load last_el_ptr "series_arry_next_element" build in
259 let _ = L.build_store series_sizeMin series_size_ptr build in
260 let _ = L.build_ret last_el_val build in
261 StringMap.add (type_str typ) def m in
262 List.fold_left series_pop_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
263
264 (*expression generation*)
265 let rec expr builder ((_, e) : sexpr) = match e with
266 | SSliteral s -> L.build_global_stringptr s "str" builder
267 | SBliteral b -> L.const_int i1_t (if b then 1 else 0)
268 | SIliteral i -> L.const_int i32_t i
269 | SFliteral f -> L.const_float_of_string float_t f
270 | SId s -> L.build_load (lookup s) s builder
271 | SNoexpr -> L.const_int i32_t 0
272 | SAssign (s, e) -> let e' = expr builder e in
273     ignore(L.build_store e' (lookup s) builder); e'
274 (* | SAttrAssign (objname, attr, e) -> let e' = expr builder e in
275 (match attr with
276 "name" -> (L.build_call setplayername_func [|L.build_load (lookup objname) objname builder|]; e') "setplayername" b
277 | "score" -> (L.build_call setplayerscore_func [|L.build_load (lookup objname) objname builder|]; e') "setplayersco
278 ) *)
279 | SPrintCall(e) ->
280 (match fst e with
281 A.String -> L.build_call printf_func [| L.const_in_bounds_gep str_format_str [|L.const_int i32_t 0; L.const_int i32_t 0|]; e'] "printf" builder
282 | A.Int | A.Void -> L.build_call printf_func [| L.const_in_bounds_gep int_format_str [|L.const_int i32_t 0; L.const_int i32_t 0|]; e'] "printf" builder
283 | A.Float -> L.build_call printf_func [| L.const_in_bounds_gep float_format_str [|L.const_int i32_t 0; L.const_int i32_t 0|]; e'] "printf" builder
284 | A.Bool -> L.build_call printf_func [| L.const_in_bounds_gep bool_format_str [|L.const_int i32_t 0; L.const_int i32_t 0|]; e'] "printf" builder
285 | _ -> raise (Failure "Print of this type is not supported") (* Potentially need to support Class, Series, and None

```

```

290     )
291 | SPlayerClassCall(e) ->
292   L.build_call playercall_func (Array.of_list (List.map (expr builder) (e))) "playercall" builder
293 | SCardClassCall(e) ->
294   L.build_call cardcall_func (Array.of_list (List.map (expr builder) (e))) "cardcall" builder
295 | SAttrCall(objname, attr) ->
296   (match attr with
297   | "name" -> L.build_call getplayername_func [(L.build_load (lookup objname) objname builder)] "getplayername" build
298   | "score" -> L.build_call getplayerscore_func [(L.build_load (lookup objname) objname builder)] "getplayerscore"
299   | "type" -> L.build_call getcardtype_func [(L.build_load (lookup objname) objname builder)] "getcardtype" builder
300   | "faceup" -> L.build_call getcardfaceup_func [(L.build_load (lookup objname) objname builder)] "getcardfaceup" b
301   | "value" -> L.build_call getcardvalue_func [(L.build_load (lookup objname) objname builder)] "getcardvalue" buil
302   | _ -> raise (Failure "attribute is not supported")
303   )
304 | SActionCall(a, args) ->
305   let (adef, main_func) = StringMap.find a action_decls_map in
306   let llargs = List.rev (List.map (expr builder) (List.rev args)) in
307   let result = (match main_func.satyp with
308               | A.Void -> ""
309               | _ -> a ^ "_result")
310   in L.build_call adef (Array.of_list llargs) result builder
311 | SBinop ((A.Float, _) as e1, op, e2) ->
312   let e1' = expr builder e1
313   and e2' = expr builder e2 in
314   (match op with
315   | A.Add -> L.build_fadd
316   | A.Sub -> L.build_fsub
317   | A.Mult -> L.build_fmuls
318   | A.Div -> L.build_fdiv
319   | A.Equal -> L.build_fcmp L.Fcmp.Oeq
320   | A.Neq -> L.build_fcmp L.Fcmp.One
321   | A.Less -> L.build_fcmp L.Fcmp.Olt
322   | A.Leq -> L.build_fcmp L.Fcmp.Ole
323   | A.Greater -> L.build_fcmp L.Fcmp.Ogt
324   | A.Geq -> L.build_fcmp L.Fcmp.Oge
325   | A.And | A.Or ->
326     raise (Failure "internal error: semant should have rejected and/or on float")
327   ) e1' e2' "tmp" builder
328 | SBinop (e1, op, e2) ->
329   let e1' = expr builder e1
330   and e2' = expr builder e2
331   in
332   (match op with
333   | A.Add -> L.build_add
334   | A.Sub -> L.build_sub
335   | A.Mult -> L.build_mul
336   | A.Div -> L.build_sdiv
337   | A.And -> L.build_and
338   | A.Or -> L.build_or
339   | A.Equal -> L.build_icmp L.Icmp.Eq
340   | A.Neq -> L.build_icmp L.Icmp.Ne
341   | A.Less -> L.build_icmp L.Icmp.Slt

```

```

342 | A.Leq    -> L.build_icmp L.Icmp.Sle
343 | A.Greater -> L.build_icmp L.Icmp.Sgt
344 | A.Geq    -> L.build_icmp L.Icmp.Sge
345 ) e1' e2' "tmp" builder
346 | SUnop(op, ((t, _) as e)) ->
347   let e' = expr builder e in
348   (match op with
349   | A.Neg when t = A.Float -> L.build_fneg
350   | A.Neg                -> L.build_neg
351   | A.Not                -> L.build_not)
352   e' "tmp" builder
353 | SSeriesliteral (series_type, literals) ->
354   let ltype = (ltype_of_type series_type) in (*gets type of elements in arr *)
355   let new_series_ptr = L.build_alloca (series_t ltype) "new_series_ptr" builder in
356   let _ = init_series builder new_series_ptr series_type in
357   let map_func literal =
358     ignore(L.build_call (StringMap.find (type_str series_type) series_push) [| new_series_ptr; (expr builder literal)
359   in
360   let _ = List.rev (List.map map_func literals) in
361   L.build_load new_series_ptr "new_series" builder
362 | SSeriesGet (series_type, id, e) ->
363   L.build_call (StringMap.find (type_str series_type) series_get) [| (lookup id); (expr builder e) |] "series_get"
364 | SSeriesSize (series_type, id) ->
365   L.build_call ((StringMap.find (type_str series_type)) series_size) [| (lookup id) |] "series_size" builder
366 | SSeriesPop (series_type, id) ->
367   L.build_call ((StringMap.find (type_str series_type)) series_pop) [| (lookup id) |] "series_pop" builder
368 in
369
370 let add_terminal builder instr =
371 match L.block_terminator (L.insertion_block builder) with
372   Some _ -> ()
373   | None -> ignore (instr builder) in
374
375 let rec stmt builder = function
376 | SBlock stmt_list -> List.fold_left stmt builder stmt_list
377 | SExpr e -> ignore(expr builder e); builder
378 | SReturn e -> ignore(match adecl.satyp with
379   (* Special "return nothing" instr *)
380   A.Void -> L.build_ret_void builder
381   (* Build return statement *)
382   | _ -> L.build_ret (expr builder e) builder );
383   builder
384 | SSeriesPush (id, e) ->
385   ignore(L.build_call (StringMap.find (type_str (fst e)) series_push) [| (lookup id); (expr builder e) |] "" builder)
386 | SIf (predicate, then_stmt, else_stmt) ->
387   let bool_val = expr builder predicate in
388   let merge_bb = L.append_block context "merge" the_action in
389   let build_br_merge = L.build_br merge_bb in (* partial function *)
390   let then_bb = L.append_block context "then" the_action in
391   add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
392   build_br_merge;
393   let else_bb = L.append_block context "else" the_action in

```

```

394     add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
395     build_br_merge;
396     ignore(L.build_cond_br bool_val then_bb else_bb builder);
397     L.builder_at_end context merge_bb
398     | SWhile (predicate, body) ->
399     let pred_bb = L.append_block context "while" the_action in
400     ignore(L.build_br pred_bb builder);
401
402     let body_bb = L.append_block context "while_body" the_action in
403     add_terminal (stmt (L.builder_at_end context body_bb) body)
404     (L.build_br pred_bb);
405     let pred_builder = L.builder_at_end context pred_bb in
406     let bool_val = expr pred_builder predicate in
407
408     let merge_bb = L.append_block context "merge" the_action in
409     ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
410     L.builder_at_end context merge_bb
411     | SFor (e1, e2, e3, body) -> stmt builder
412     ( SBlock [SEExpr e1 ; SWhile (e2, SBlock [body ; SEExpr e3]) ] )
413     | SNostmt -> builder
414     in
415
416 let builder = stmt builder (SBlock adecl.sabody) in
417   add_terminal builder (match adecl.satyp with
418     A.Void -> L.build_ret_void
419     | A.Float -> L.build_ret (L.const_float float_t 0.0)
420     | t -> L.build_ret (L.const_int (ltype_of_typ t) 0));
421   ()
422   in
423 let _ = List.iter build_action_body action_decls in
424
425 (*~~~~~ main function generation top-level ~~~~~*)
426 let build_main main_func =
427     let fty = L.function_type i32_t[[]] in
428     let f = L.define_function "main" fty the_module in
429     let builder = L.builder_at_end context (L.entry_block f) in
430
431 let local_vars =
432   let add_param m (t, n) p = L.set_value_name n p;
433   let local = L.build_alloca (ltype_of_typ t) n builder in
434   ignore(
435     match t with
436     A.Series series_type -> init_series builder local series_type
437     | _ -> ()
438   );
439   ignore (L.build_store p local builder);
440 StringMap.add n local m
441
442 (* Allocate space for any locally declared variables and add the
443  * resulting registers to our map *)
444 and add_local m (t, n) =
445 let local_var = L.build_alloca (ltype_of_typ t) n builder in

```

```

446     ignore(
447         match t with
448             A.Series series_type -> init_series builder local_var series_type
449         | _ -> ()
450     );
451     StringMap.add n local_var m
452 in
453
454     let params = List.fold_left2 add_param StringMap.empty main_func.smparams
455         (Array.to_list (L.params f)) in (*not sure about this f *)
456     List.fold_left add_local params main_func.smlocals
457 in
458
459 let lookup n = try StringMap.find n local_vars
460     with Not_found -> StringMap.find n global_vars
461 in
462 (*for the series stuff, referenced past projects + clang *)
463 let init_series builder series_ptr series_type =
464     let size_ptr_ptr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
465     let size_ptr = L.build_alloca i32_t "series_size" builder in
466     ignore(L.build_store (L.const_int i32_t 0) size_ptr builder);
467     ignore(L.build_store size_ptr size_ptr_ptr builder);
468     let series_ptr = L.build_struct_gep series_ptr 1 "series" builder in (* init array (series) *)
469     (* fails when a = [] *)
470     let p = L.build_array_alloca (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
471     ignore(L.build_store p series_ptr builder);
472 in
473
474 let series_push : L.llvalue StringMap.t =
475     let series_push_ty m typ =
476         let series_push_def = L.define_function ("series_push" ^ (type_str typ)) (L.function_type void_t [] L.pointer_type (s
477         let build = L.builder_at_end context (L.entry_block series_push_def) in
478         let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
479         ignore(L.build_store (L.param series_push_def 0) series_ptr build);
480         let valPtr = L.build_alloca (ltype_of_typ typ) "val_alloc" build in
481         ignore(L.build_store (L.param series_push_def 1) valPtr build);
482         let series_load = L.build_load series_ptr "series_load" build in
483         let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
484         let series_load_2 = L.build_load series_ptr_2 "series_load_2" build in
485         let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
486         let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
487         let series_size = L.build_load series_size_ptr "series_size" build in
488         let next_el_ptr = L.build_gep series_load_2 [| series_size |] "series_next_el_ptr" build in
489         let next_size = L.build_add series_size (L.const_int i32_t 1) "next_size" build in
490         ignore(L.build_store next_size series_size_ptr build);
491         ignore(L.build_store (L.build_load valPtr "val" build) next_el_ptr build);
492         ignore(L.build_ret_void build);
493         StringMap.add (type_str typ) series_push_def m in
494     List.fold_left series_push_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in (*change orde
495
496 let series_get : L.llvalue StringMap.t =
497     let series_get_ty m typ =

```

```

498     let def = L.define_function ("series_get" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (se
499     let build = L.builder_at_end context (L.entry_block def) in
500     let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
501     ignore(L.build_store (L.param def 0) series_ptr build);
502     let idx_ptr = L.build_alloca i32_t "idx_alloc" build in
503     ignore(L.build_store (L.param def 1) idx_ptr build);
504     let series_load = L.build_load series_ptr "series_load" build in
505     let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
506     let series_load_2 = L.build_load series_ptr_2 "array_load" build in
507     let idx = L.build_load idx_ptr "idx_load" build in
508     let series_el_ptr = L.build_gep series_load_2 [| idx |] "series_el_ptr" build in
509     let element_val = L.build_load series_el_ptr "series_el_ptr" build in
510     ignore(L.build_ret element_val build);
511     StringMap.add (type_str typ) def m in
512     List.fold_left series_get_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
513
514     let series_size : L.llvalue StringMap.t =
515         let series_size_ty m typ =
516             let series_size_def = L.define_function ("series_size" ^ (type_str typ)) (L.function_type i32_t [| L.pointer_type (se
517             let build = L.builder_at_end context (L.entry_block series_size_def) in
518             let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
519             ignore(L.build_store (L.param series_size_def 0) series_ptr build);
520             let series_load = L.build_load series_ptr "series_load" build in
521             let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
522             let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
523             let series_size = L.build_load series_size_ptr "series_size" build in
524             ignore(L.build_ret series_size build);
525             StringMap.add (type_str typ) series_size_def m in
526             List.fold_left series_size_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
527
528         let series_pop : L.llvalue StringMap.t =
529             let series_pop_ty m typ =
530                 let def = L.define_function ("series_pop" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (series
531                 let build = L.builder_at_end context (L.entry_block def) in
532                 let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
533                 ignore(L.build_store (L.param def 0) series_ptr build);
534                 let series_load = L.build_load series_ptr "series_load" build in
535                 let series_arr_ptr = L.build_struct_gep series_load 1 "series_arr_ptr" build in
536                 let series_arr_load = L.build_load series_arr_ptr "series_arr_load" build in
537                 let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
538                 let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
539                 let series_size = L.build_load series_size_ptr "series_size" build in
540                 let series_sizeMin = L.build_sub series_size (L.const_int i32_t 1) "dec_size" build in
541                 let last_el_ptr = L.build_gep series_arr_load [| series_sizeMin |] "series_next_el_ptr" build in
542                 let last_el_val = L.build_load last_el_ptr "series_array_next_element" build in
543                 let _ = L.build_store series_sizeMin series_size_ptr build in
544                 let _ = L.build_ret last_el_val build in
545                 StringMap.add (type_str typ) def m in
546                 List.fold_left series_pop_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in
547
548             (*expression generation*)
549             let rec expr builder ((_, e) : sexpr) = match e with

```

```

550 | SSliteral s  -> L.build_global_stringptr s "str" builder
551 | SBliteral b -> L.const_int i1_t (if b then 1 else 0)
552 | SIliteral i -> L.const_int i32_t i
553 | SFliteral f -> L.const_float_of_string float_t f
554 | SId s       -> L.build_load (lookup s) s builder
555 | SNoexpr     -> L.const_int i32_t 0
556 | SAssign (s, e) -> let e' = expr builder e in
557 |               ignore(L.build_store e' (lookup s) builder); e'
558 | SPrintCall(e) ->
559 |   (match fst e with
560 |   A.String -> L.build_call printf_func [| L.const_in_bounds_gep str_format_str [|L.const_int i32_t 0; L.const_int i32_t
561 |   "printf" builder
562 |   A.Int | A.Void -> L.build_call printf_func [| L.const_in_bounds_gep int_format_str [|L.const_int i32_t 0; L.const_int i32_t
563 |   "printf" builder
564 |   A.Float -> L.build_call printf_func [| L.const_in_bounds_gep float_format_str [|L.const_int i32_t 0; L.const_int i32_t
565 |   "printf" builder
566 |   A.Bool -> L.build_call printf_func [| L.const_in_bounds_gep bool_format_str [|L.const_int i32_t 0; L.const_int i32_t
567 |   "printf" builder
568 |   _ -> raise (Failure "Print of this type is not supported") (* Potentially need to support Class, Series, and None c
569 |   )
570 |   (* | SAttrAssign (objname, attr, e) -> let e' = expr builder e in
571 |   (match attr with
572 |   "name" -> (L.build_call setplayername_func [|L.build_load (lookup objname) objname builder) ; e']] "setplayername" b
573 |   / "score" -> (L.build_call setplayerscore_func [|L.build_load (lookup objname) objname builder) ; e']] "setplayersco
574 |   ) *)
575 |   SPlayerClassCall(e) ->
576 |   L.build_call playercall_func (Array.of_list (List.map (expr builder) (e))) "playercall" builder
577 |   SCardClassCall(e) ->
578 |   L.build_call cardcall_func (Array.of_list (List.map (expr builder) (e))) "cardcall" builder
579 |   SAttrCall(objname, attr) ->
580 |   (match attr with
581 |   "name" -> L.build_call getplayername_func [|L.build_load (lookup objname) objname builder)|] "getplayername" build
582 |   | "score" -> L.build_call getplayerscore_func [|L.build_load (lookup objname) objname builder)|] "getplayerscore"
583 |   | "type" -> L.build_call getcardtype_func [|L.build_load (lookup objname) objname builder)|] "getcardtype" builder
584 |   | "faceup" -> L.build_call getcardfaceup_func [|L.build_load (lookup objname) objname builder)|] "getcardfaceup" b
585 |   | "value" -> L.build_call getcardvalue_func [|L.build_load (lookup objname) objname builder)|] "getcardvalue" buil
586 |   | _ -> raise (Failure "attribute is not supported")
587 |   )
588 |   SActionCall(a, args) ->
589 |   let (adef, main_func) = StringMap.find a action_decls_map in
590 |   let llargs = List.rev (List.map (expr builder) (List.rev args)) in
591 |   let result = (match main_func.satyp with
592 |   A.Void -> ""
593 |   | _ -> a ^ "_result")
594 |   in L.build_call adef (Array.of_list llargs) result builder
595 |   SBinop ((A.Float,_) as e1, op, e2) ->
596 |   let e1' = expr builder e1
597 |   and e2' = expr builder e2 in
598 |   (match op with
599 |   A.Add -> L.build_fadd
600 |   | A.Sub -> L.build_fsub
601 |   | A.Mult -> L.build_fmuls

```

```

602 | A.Div    -> L.build_fdiv
603 | A.Equal  -> L.build_fcmp L.Fcmp.Oeq
604 | A.Neq    -> L.build_fcmp L.Fcmp.One
605 | A.Less   -> L.build_fcmp L.Fcmp.Olt
606 | A.Leq    -> L.build_fcmp L.Fcmp.Ole
607 | A.Greater -> L.build_fcmp L.Fcmp.Ogt
608 | A.Geq    -> L.build_fcmp L.Fcmp.Oge
609 | A.And | A.Or ->
610 raise (Failure "internal error: semant should have rejected and/or on float")
611 ) e1' e2' "tmp" builder
612 | SBinop (e1, op, e2) ->
613 let e1' = expr builder e1
614 and e2' = expr builder e2
615 in
616 (match op with
617 A.Add    -> L.build_add
618 | A.Sub   -> L.build_sub
619 | A.Mult  -> L.build_mul
620 | A.Div   -> L.build_sdiv
621 | A.And   -> L.build_and
622 | A.Or    -> L.build_or
623 | A.Equal -> L.build_icmp L.Icmp.Eq
624 | A.Neq   -> L.build_icmp L.Icmp.Ne
625 | A.Less  -> L.build_icmp L.Icmp.Slt
626 | A.Leq   -> L.build_icmp L.Icmp.Sle
627 | A.Greater -> L.build_icmp L.Icmp.Sgt
628 | A.Geq   -> L.build_icmp L.Icmp.Sge
629 ) e1' e2' "tmp" builder
630 | SUnop(op, ((t, _) as e)) ->
631 let e' = expr builder e in
632 (match op with
633 | A.Neg when t = A.Float -> L.build_fneg
634 | A.Neg                -> L.build_neg
635 | A.Not                 -> L.build_not)
636 e' "tmp" builder
637 | SSeriesliteral (series_type, literals) ->
638 let ltype = (ltype_of_ttyp series_type) in (*gets type of elements in arr *)
639 let new_series_ptr = L.build_alloc (series_t ltype) "new_series_ptr" builder in
640 let _ = init_series builder new_series_ptr series_type in
641 let map_func literal =
642     ignore(L.build_call (StringMap.find (type_str series_type) series_push) [| new_series_ptr; (expr builder literal)
643 in
644 let _ = List.rev (List.map map_func literals) in
645 L.build_load new_series_ptr "new_series" builder
646 | SSeriesGet (series_type, id, e) ->
647     L.build_call (StringMap.find (type_str series_type) series_get) [| (lookup id); (expr builder e) |] "series_get"
648 | SSeriesSize (series_type, id) ->
649     L.build_call ((StringMap.find (type_str series_type)) series_size) [| (lookup id) |] "series_size" builder
650 | SSeriesPop (series_type, id) ->
651     L.build_call ((StringMap.find (type_str series_type)) series_pop) [| (lookup id) |] "series_pop" builder
652 in
653

```



```

654 let add_terminal builder instr =
655   match L.block_terminator (L.insertion_block builder) with
656     Some _ -> ()
657     | None -> ignore (instr builder) in
658
659 let rec stmt builder = function
660   | SBlock stmt_list -> List.fold_left stmt builder stmt_list
661   | SExpr e -> ignore(expr builder e); builder
662   | SReturn e -> ignore(match main_func.smtyp with (*Should be unused*)
663     (* Special "return nothing" instr *)
664     A.Void -> L.build_ret_void builder
665     (* Build return statement *)
666     | _ -> L.build_ret (expr builder e) builder );
667     builder
668   | SSeriesPush (id, e) ->
669     ignore(L.build_call (StringMap.find (type_str (fst e)) series_push) [] (lookup id); (expr builder e) [] "" builder)
670   | SIf (predicate, then_stmt, else_stmt) ->
671     let bool_val = expr builder predicate in
672     let merge_bb = L.append_block context "merge" f in
673     let build_br_merge = L.build_br merge_bb in (* partial function *)
674
675     let then_bb = L.append_block context "then" f in
676     add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
677     build_br_merge;
678
679     let else_bb = L.append_block context "else" f in
680     add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
681     build_br_merge;
682     ignore(L.build_cond_br bool_val then_bb else_bb builder);
683     L.builder_at_end context merge_bb
684     | SWhile (predicate, body) ->
685     let pred_bb = L.append_block context "while" f in
686     ignore(L.build_br pred_bb builder);
687
688     let body_bb = L.append_block context "while_body" f in
689     add_terminal (stmt (L.builder_at_end context body_bb) body)
690     (L.build_br pred_bb);
691
692     let pred_builder = L.builder_at_end context pred_bb in
693     let bool_val = expr pred_builder predicate in
694
695     let merge_bb = L.append_block context "merge" f in
696     ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
697     L.builder_at_end context merge_bb
698     | SFor (e1, e2, e3, body) -> stmt builder
699     ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )
700   | SNostmt -> builder
701     (*| SForLit ( e1, e2, body) -> stmt builder
702     ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e]) ] ) *)
703   in
704
705 let builder = stmt builder (SBlock main_func.smbody) in

```

```

706     let _ = L.build_ret (L.const_int i32_t 0) (builder) in
707     () in
708
709 let _ = build_main main_decl in
710 the_module;

```

playercall.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4
5  typedef struct {
6     int *array;
7     size_t used;
8     size_t size;
9 } Array;
10
11 void initArray(Array *a, size_t initialSize) {
12     a->array = malloc(initialSize * sizeof(int));
13     a->used = 0;
14     a->size = initialSize;
15 }
16
17 void insertArray(Array *a, int element) {
18     // a->used is the number of used entries, because a->array[a->used++] updates a->used only *after* the array has been acc
19     // Therefore a->used can go up to a->size
20     if (a->used == a->size) {
21         a->size *= 2;
22         a->array = realloc(a->array, a->size * sizeof(int));
23     }
24     a->array[a->used++] = element;
25 }
26
27 void freeArray(Array *a) {
28     free(a->array);
29     a->array = NULL;
30     a->used = a->size = 0;
31 }
32
33 struct Player
34 {
35     char *name;
36     int score;
37 };
38
39 /*int main() {
40
41     struct Player *p;
42     p = player("Bob", 1);
43

```

```

44     printf("Pointer is: (%s, %d)", p->name, p->score);
45
46     return 0;
47 }*/
48
49 struct Player *playercall(char* name, int score)
50 {
51     struct Player *p;
52     p = (struct Player *) malloc(sizeof(struct Player));
53     p->name = name;
54     p->score = score;
55     return p;
56 }
57
58 char *getplayername(struct Player *player)
59 {
60     return player->name;
61 }
62
63 int getplayerscore(struct Player *player)
64 {
65     return player->score;
66 }
67
68 void *setplayername(struct Player *player, char* name)
69 {
70     player->name = name;
71     return (void *)player->name;
72 }
73
74 int setplayerscore(struct Player *player, int score)
75 {
76     player->score = score;
77     return player->score;
78 }
79
80 struct Card
81 {
82     char *type;
83     bool faceup;
84     int value;
85 };
86
87 struct Card *cardcall(char* type, bool faceup, int value)
88 {
89     struct Card *c;
90     c = (struct Card *) malloc(sizeof(struct Card));
91     c->type = type;
92     c->faceup = faceup;
93     c->value = value;
94     return c;
95 }

```

```

96
97 char *getcardtype(struct Card *card)
98 {
99     return card->type;
100 }
101
102 bool getcardfaceup(struct Card *card)
103 {
104     return card->faceup;
105 }
106
107 int getcardvalue(struct Card *card)
108 {
109     return card->value;
110 }

```

AHOD.ml

```

1  (* Top-level of the MicroC compiler: scan & parse the input,
2     check the resulting AST and generate an SAST from it, generate LLVM IR,
3     and dump the module *)
4
5  type action = Ast | Sast | LLVM_IR | Compile
6  (* | Token *)
7
8  let () =
9      let action = ref Compile in
10     let set_action a () = action := a in
11     let speclist = [
12         ("-a", Arg.Unit (set_action Ast), "Print the AST");
13         ("-s", Arg.Unit (set_action Sast), "Print the SAST");
14         (* ("-t", Arg.Unit (set_action Token ), "Print Tokens"); *)
15         ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
16         ("-c", Arg.Unit (set_action Compile),
17            "Check and print the generated LLVM IR (default)");
18     ] in
19     let usage_msg = "usage: ./AHOD.native [-a|-s|-t|-l|-c] [test-helloworld.ah]" in
20     let channel = ref stdin in
21     Arg.parse speclist (fun filename -> channel := open_in filename) usage_msg;
22
23     let lexbuf = Lexing.from_channel !channel in
24     (* match action with
25      | Token -> print_string
26      | _ -> *)
27     let ast = Parse.program Scanner.token lexbuf
28     in
29     match !action with
30     | Ast -> ()
31     | _ -> let sast = Semant.check ast in
32         match !action with
33         | Ast -> ()

```

```

34     | Sast    -> ()
35     | LLVM_IR -> ()
36     | Compile -> let m = Codegen.translate sast
37     in
38     Llvm_analysis.assert_valid_module m;
39     print_string (Llvm.string_of_llmodule m)

```

testall.sh

```

1  #!/bin/sh
2
3  # Regression testing script for MicroC
4  # Step through a list of files
5  # Compile, run, and check the output of each expected-to-work test
6  # Compile and check the error of each expected-to-fail test
7
8  # Path to the LLVM interpreter
9  LLI="lli"
10 #LLI="/usr/local/opt/llvm/bin/lli"
11
12 # Path to the LLVM compiler
13 LLC="llc"
14
15 # Path to the C compiler
16 CC="cc"
17
18 # Path to the microc compiler. Usually "./microc.native"
19 # Try "_build/microc.native" if ocamlbuild was unable to create a symbolic link.
20 AHOD="./AHOD.native"
21 #MICROC="_build/microc.native"
22
23 # Set time limit for all operations
24 ulimit -t 30
25
26 globallog=testall.log
27 rm -f $globallog
28 error=0
29 globalerror=0
30
31 keep=0
32
33 Usage() {
34     echo "Usage: testall.sh [options] [.mc files]"
35     echo "-k    Keep intermediate files"
36     echo "-h    Print this help"
37     exit 1
38 }
39
40 SignalError() {
41     if [ $error -eq 0 ] ; then
42         echo "FAILED"

```

```

43     error=1
44     fi
45     echo " $1"
46 }
47
48 # Compare <outfile> <reffile> <difffile>
49 # Compares the outfile with reffile. Differences, if any, written to difffile
50 Compare() {
51     generatedfiles="$generatedfiles $3"
52     echo diff -b $1 $2 ">" $3 1>&2
53     diff -b "$1" "$2" > "$3" 2>&1 || {
54         SignalError "$1 differs"
55         echo "FAILED $1 differs from $2" 1>&2
56     }
57 }
58
59 # Run <args>
60 # Report the command, run it, and report any errors
61 Run() {
62     echo $* 1>&2
63     eval $* || {
64         SignalError "$1 failed on $*"
65         return 1
66     }
67 }
68
69 # RunFail <args>
70 # Report the command, run it, and expect an error
71 RunFail() {
72     echo $* 1>&2
73     eval $* && {
74         SignalError "failed: $* did not report an error"
75         return 1
76     }
77     return 0
78 }
79
80 Check() {
81     error=0
82     basename=`echo $1 | sed 's/.*\\//\\
83                 s/.ah//'\`
84     reffile=`echo $1 | sed 's/.ah$//'\`
85     basedir=""`echo $1 | sed 's/\/[^\//]*$//'\`/."
86
87     echo -n "$basename..."
88
89     echo 1>&2
90     echo "##### Testing $basename" 1>&2
91
92     generatedfiles=""
93
94     generatedfiles="$generatedfiles ${basename}.11 ${basename}.s ${basename}.exe ${basename}.out" &&

```

```

95 Run "$AHOD" "$1" ">" "${basename}.ll" &&
96 Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
97 Run "$CC" "-o" "${basename}.exe" "${basename}.s" "playercall.o" &&
98 Run "./${basename}.exe" > "${basename}.out" &&
99 Compare ${basename}.out ${reffile}.out ${basename}.diff
100
101 # Report the status and clean up the generated files
102
103 if [ $Error -eq 0 ] ; then
104     if [ $keep -eq 0 ] ; then
105         rm -f $generatedfiles
106     fi
107     echo "OK"
108     echo "##### SUCCESS" 1>&2
109 else
110     echo "##### FAILED" 1>&2
111     globalerror=$Error
112 fi
113 }
114
115 CheckFail() {
116     error=0
117     basename=`echo $1 | sed 's/.*\\\/\\\/
118                 s/.ah//'\`
119     reffile=`echo $1 | sed 's/.ah$//'\`
120     basedir=`echo $1 | sed 's/\[/\[/]*$//'\`
121
122     echo -n "$basename..."
123
124     echo 1>&2
125     echo "##### Testing $basename" 1>&2
126
127     generatedfiles=""
128
129     generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
130     RunFail "$AHOD" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
131     Compare ${basename}.err ${reffile}.err ${basename}.diff
132
133     # Report the status and clean up the generated files
134
135     if [ $Error -eq 0 ] ; then
136         if [ $keep -eq 0 ] ; then
137             rm -f $generatedfiles
138         fi
139         echo "OK"
140         echo "##### SUCCESS" 1>&2
141     else
142         echo "##### FAILED" 1>&2
143         globalerror=$Error
144     fi
145 }
146

```

```

147 while getopts kdpsh c; do
148     case $c in
149         k) # Keep intermediate files
150             keep=1
151             ;;
152         h) # Help
153             Usage
154             ;;
155         esac
156     done
157
158     shift `expr $OPTIND - 1`
159
160     LLIFail() {
161         echo "Could not find the LLVM interpreter \"$LLI\"."
162         echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
163         exit 1
164     }
165
166     which "$LLI" >> $globallog || LLIFail
167
168     if [ ! -f playercall.o ]
169     then
170         echo "Could not find playercall.o"
171         echo "Try \"make playercall.o\""
172         exit 1
173     fi
174
175     if [ $# -ge 1 ]
176     then
177         files=$@
178     else
179         files="tests/test-*.ah tests/fail-*.ah"
180     fi
181
182     for file in $files
183     do
184         case $file in
185             *test-*)
186             Check $file 2>> $globallog
187             ;;
188             *fail-*)
189             CheckFail $file 2>> $globallog
190             ;;
191             *)
192             echo "unknown file type $file"
193             globalerror=1
194             ;;
195         esac
196     done
197
198     exit $globalerror

```

test-action.ah

```
1 int x
2 when do void PLAY():
3 {
4     do PRINT("actions are cool")
5 }
6 main:
7 {
8     do PLAY()
9 }
```

test-action2.ah

```
1 int x
2 when do void PLAY():
3 {
4     do PRINT("actions are cool")
5 }
6 when do void PLAYTWO():
7 {
8     do PRINT("two actions are cooler")
9 }
10 main:
11 {
12     do PLAY()
13     do PLAYTWO()
14 }
```

test-action3-while.ah

```
1 when do void PLAY():
2 {
3     while 3>5:
4     {
5         do PRINT(0)
6     }
7     do PRINT(1)
8 }
9 main:
10 {
11     do PLAY()
12 }
```

test-action4-for.ah

```
1 when do void PLAY():
2 {
3     int i
4     for (i = 0 ; i < 5; i = i + 1):
5     {
6     do PRINT(i)
7     }
8 }
9
10 main:
11 {
12     do PLAY()
13 }
```

test-aciton5-if.ah

```
1 when do void PLAY():
2 {
3     if 3<5 :
4     {
5     do PRINT(1)
6     }
7 }
8 main:
9 {
10     do PLAY()
11 }
```

test-action6-ifelse.ah

```
1 when do void PLAY():
2 {
3     if 3>5 :
4     {
5     do PRINT(1)
6     }
7     else:
8     {
9     do PRINT(0)
10    }
11 }
12 main:
13 {
14     do PLAY()
15 }
```

test-action7-params.ah

```
1 when do string A(string x,int y):
2 {
3     string z
4     z = x
5     do PRINT(y)
6     return z
7 }
8 main:
9 {
10    string w
11    w = do A("Ahoy Matey, to AHOD", 3)
12    do PRINT(w)
13 }
```

test-advanced.ah

```
1 when do series<Card> CREATEPILE():
2 {
3     Card card1
4     card1 = Card("R5", true, 5)
5     return [card1]
6 }
7 main:
8 {
9     series<Card> cards
10    Card card
11    cards = do CREATEPILE()
12    card = cards[0]
13    do PRINT(card.type)
14    do PRINT(card.faceup)
15 }
```

test-assign-global.ah

```
1 int x
2 bool y
3 string z
4 main:
5 {
6     x = 4
7     do PRINT(x)
8     y = true
9     do PRINT(y)
10    z = "banana"
11    do PRINT(z)
12 }
```

test-assign-local.ah

```
1 main:
2 {
3   int x
4   x = 3
5   do PRINT(x)
6 }
```

test-assign-local2.ah

```
1 int z
2 when do void PLAY():
3 {
4   int x
5   int y
6   x = 3
7   y = 4
8   do PRINT(x)
9   do PRINT(y)
10 }
11 main:
12 {
13   int a
14   a = 3
15   z = 5
16   do PRINT(z)
17   do PRINT(a)
18   do PLAY()
19 }
```

test-basic.ah

```
1 int a
2 main:
3 {
4   float b
5   b = 5.0
6   if 3<5:
7     {
8       for (a = 0 ; a < 5; a = a + 1):
9         {
10          b = b + 1.0
11          do PRINT(b)
12        }
13      }
14 }
```

test-binop.ah

```

1  main:
2  {
3  do PRINT(1 + 2)
4  do PRINT(1 - 2)
5  do PRINT(1 * 2)
6  do PRINT(1.0 / 2.0)
7  do PRINT(true)
8  do PRINT(true and true)
9  do PRINT(true and false)
10 do PRINT(true and true)
11 do PRINT(false and false)
12 do PRINT(true or true)
13 do PRINT(true or false)
14 do PRINT(true or true)
15 do PRINT(false or false)
16 do PRINT(1 == 2)
17 do PRINT(1 == 1)
18 do PRINT(1 != 2)
19 do PRINT(1 != 1)
20 do PRINT(1 < 2)
21 do PRINT(2 < 1)
22 do PRINT(1 <= 2)
23 do PRINT(3 <= 2)
24 do PRINT(1 > 2)
25 do PRINT(2 > 1)
26 do PRINT(1 >= 2)
27 do PRINT(3 >= 2)
28 do PRINT(-3 + 1)
29 do PRINT(-3 - 1)
30 do PRINT(not true)
31 }

```

test-bjsim.ah

```

1  int i
2  series<Card> deck
3  int score
4  Player player
5  when do void INIT():
6  {
7  player = Player("Stephen", 0)
8  do PRINT("Hello, and welcome to our Blackjack Simulator")
9  do PRINT("First things first let us welcome our player for the evening")
10 do PRINT(player.name)
11 deck = [Card("R6", true,6)]
12 deck = do CREATEDECK(deck)
13 }
14 when do string COMMENT(int x):
15 {
16 string localquip

```

```

17     if x > 6 :
18     {
19         localquip = "That is a high value card"
20     }
21     else:
22     {
23         localquip = "That is a low value card"
24     }
25     return localquip
26 }
27 when do series<Card> CREATEDECK(series<Card> deck):
28 {
29     series<string> types
30     series<int> values
31     int value
32     string type
33     Card card
34     types = ["R5", "R9", "B9", "B2", "R3"]
35     values = [3, 9, 9, 2, 3]
36     for (i = 0; i < types.size(); i = i + 1):
37     {
38         type = types[i]
39         value = values[i]
40         card = Card(type, true,value)
41         deck.push(card)
42     }
43     return deck
44 }
45 main:
46 {
47     Card card
48     int currvalue
49     string quip
50     do INIT()
51     do PRINT("There are 5 cards in the hand. A Hit Request")
52     card = deck[deck.size()-1]
53     score = 0
54     do PRINT("Card:")
55     do PRINT(card.type)
56     do PRINT("Current Score of:")
57     score = score + card.value
58     do PRINT(score)
59     deck.pop()
60     do PRINT("Another hit request")
61     card = deck[deck.size()-1]
62     do PRINT("Card:")
63     do PRINT(card.type)
64     do PRINT("Current Score of:")
65     score = score + card.value
66     do PRINT(score)
67     deck.pop()
68     do PRINT("Looks like we stop here for tonight. Time to see what the next cards were")

```

```

69   for (i = 0 ; i < deck.size(); i = i + 1):
70   {
71       card = deck[i]
72       do PRINT("Card:")
73       do PRINT(card.type)
74       score = score + card.value
75       quip = do COMMENT(score)
76       do PRINT(quip)
77       if score < 21 :
78       {
79       do PRINT("Could have had a higher score of: ")
80       do PRINT(score)
81       }
82       else:
83       {
84       do PRINT("Good that you stopped otherwise you would have a score of: ")
85       do PRINT(score)
86       }
87   }
88 }

```

test-class1.ah

```

1  Player player
2  main:
3  {
4      player = Player("bob", 0)
5      do PRINT(player.name)
6      do PRINT(player.score)
7  }

```

test-class2.ah

```

1  Card card
2  main:
3  {
4      card = Card("R5", true, 5)
5      do PRINT(card.type)
6      do PRINT(card.faceup)
7      do PRINT(card.value)
8  }

```

test-class3.ah

```

1  Card card
2  main:
3  {
4      card = Card("R5", true, 5)

```

```

5   do PRINT(card.type)
6   do PRINT(card.faceup)
7   do PRINT(card.value)
8   card = Card("R5", false, 5)
9   do PRINT(card.faceup)
10  }

```

test-class4.ah

```

1  Player player
2  main:
3  {
4    string name
5    int score
6    name = "bob"
7    score = 0
8    player = Player(name, score)
9    do PRINT(player.name)
10   do PRINT(player.score)
11   /*player.score = 1*/
12   /*do PRINT(player.score)*/
13  }

```

test-class5.ah

```

1  Card card1
2  Card card2
3  Card card3
4  series<Card> deck
5  Card type1
6  int i
7  main:
8  {
9    string red5
10   red5 = "R5"
11   card1 = Card(red5, true, 5)
12   card2 = Card("R6", false, 6)
13   card3 = Card("R7", true, 7)
14   deck = [card1, card2, card3]
15   for (i = 0; i < deck.size(); i = i + 1):
16   {
17     type1 = deck[i]
18     do PRINT(type1.type)
19     do PRINT(type1.faceup)
20     do PRINT(type1.value)
21   }
22  }

```

test-class6.ah

```

1  main:
2  {
3      Card card
4      card = Card("R5", true, 5)
5      do PRINT(card.type)
6      do PRINT(card.faceup)
7      do PRINT(card.value)
8  }

```

test-comments.ah

```

1  int a /* nothing should be printed out */
2  when do void PLAY(): /*test*/
3  {/*test*/
4      do PRINT("actions are cool")/*test*/
5  }/*test*/
6  main: /*comments everywhere */
7      { /* woo */
8          /* only single line comments, decieving huh*/
9          do PLAY()
10         a = 1 /* comment and code woah */
11         /*test*/
12     } /* woo */

```

test-demo1.ah

```

1  int i
2  series<Card> deck
3  when do series<Card> CREATEDECK(series<Card> deck):
4  {
5      series<string> types
6      string type
7      Card card
8      types = ["R0", "R1", "R2", "R3", "R4"]
9      for (i = 0; i < types.size(); i = i + 1):
10     {
11         type = types[i]
12         card = Card(type, true, 5)
13         deck.push(card)
14     }
15     return deck
16 }
17 main:
18 {
19     Card card
20     /*card = Card("R0", true)*/
21     /*deck = [card]*/
22     deck = [Card("DO", true, 5)]
23     deck = do CREATEDECK(deck)

```

```
24     for (i = 0 ; i < 5; i = i + 1):
25     {
26         card = deck[i]
27         do PRINT(card.type)
28         do PRINT(card.faceup)
29     }
30 }
```

test-for1.ah

```
1  int i
2  main:
3  {
4      for (i = 0 ; i < 10; i = i + 1):
5      {
6          do PRINT(i)
7      }
8  }
```

test-for2.ah

```
1  int i
2  series<int> a
3  main:
4  {
5      a = [8,2,5]
6      for (i = 0; i < a.size(); i = i + 1):
7      {
8          do PRINT(a[i])
9      }
10     a.pop()
11     for (i = 0; i < a.size(); i = i + 1):
12     {
13         do PRINT(a[i])
14     }
15 }
```

test-helloworld.ah

```
1  main:
2  {
3      do PRINT("Hello world!")
4      do PRINT(":".)
5  }
```

test-if-else.ah

```
1 main:
2 {
3     if 3>5 :
4     {
5         do PRINT("true")
6     }
7     else:
8     {
9         do PRINT("false")
10    }
11 }
```

test-if.ah

```
1 main:
2 {
3     if 3<5 :
4     {
5         do PRINT("true")
6     }
7 }
```

test-return.ah

```
1 int w
2 string x
3 bool y
4 float z
5 when do int A():
6 {
7     return 3
8 }
9 when do string B():
10 {
11     return "some plt fun"
12 }
13 when do bool C():
14 {
15     return true
16 }
17 when do float D():
18 {
19     return 3.5
20 }
21 main:
22 {
23     w = do A()
24     do PRINT(w)
25     x = do B()
```

```
26 do PRINT(x)
27 y = do C()
28 do PRINT(y)
29 z = do D()
30 do PRINT(z)
31 }
```

test-series.ah

```
1 series<int> a
2 series<string> b
3 series<float> c
4 series<bool> d
5 main:
6 {
7     a = [1,3]
8     do PRINT(a[1])
9     b = ["banana", "apple"]
10    do PRINT(b[1])
11    c = [1.1]
12    do PRINT(c[0])
13    d = [true, false]
14    do PRINT(d[1])
15 }
```

test-series2.ah

```
1 series<int> a
2 series<string> b
3 main:
4 {
5     a = [2]
6     a.push(1)
7     do PRINT(a[1])
8     a.push(3)
9     do PRINT(a[2])
10    do PRINT(a[2-1])
11    b = ["stmt1", "stmt2"]
12    b.push("stmt3")
13    b.push("stmt4")
14    do PRINT(b[3])
15 }
```

test-series3.ah

```
1 series<int> a
2 main:
3 {
```

```
4     a = [2, 3, 4]
5     do PRINT(a.size())
6 }
```

test-series4.ah

```
1 series<int> a
2 main:
3 {
4     a = [2, 3, 4]
5     a.pop()
6     do PRINT(a.size())
7 }
```

test-series5-fr.ah

```
1 series<string> deck
2 series<string> hand
3 int i
4 main:
5 {
6     /* This is a comment */
7     deck = ["d","e","f","g"]
8     hand = ["a","b","c"] /* this is another comment */
9     hand.push(deck.pop()) /* deck.pop() gives 'e' */
10    hand.push(deck.pop()) /* deck.pop() gives 'f' */
11
12    /* hand = ["a", "b", "c", "f", "g"] */
13    /* deck = ["d","e"] */
14
15    do PRINT("hand is now:")
16    for (i = 0; i < hand.size(); i = i + 1):
17        {
18            do PRINT(hand[i])
19        }
20    do PRINT("deck is now:")
21    for (i = 0; i < deck.size(); i = i + 1):
22        {
23            do PRINT(deck[i])
24        }
25 }
```

test-while-gcd.ah

```
1 int x
2 int y
3 main:
4 {
```

```
5   x = 18
6   y = 12
7   while x!=y:
8   {
9   if x>y:
10  {
11      x = x - y
12  }
13  else:
14  {
15      y = y - x
16  }
17  }
18  do PRINT(x)
19 }
```

test-while1.ah

```
1  main:
2  {
3      while 3>5:
4      {
5          do PRINT("true")
6      }
7      do PRINT("false")
8  }
```

test-while2.ah

```
1  int x
2  main:
3  {
4      x = 1
5      while x<5:
6      {
7          x = x + 1
8      }
9      do PRINT(x)
10 }
```
