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

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 depext 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:

```
main:
{
  do PRINT("Hello world!")
  do PRINT(":)")
}
/* 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.

```
series<string> deck
series<string> hand
main:
{
    /* This is a comment */
    deck = ["d","e","f","g"]
    hand = ["a","b","c"] /* this is another comment */
    hand.push(deck.pop()) /* deck.pop() gives 'e' */
    hand.push(deck.pop()) /* deck.pop() gives 'f' */

    /* hand = ["a", "b", "c", "f", "g"] */
    /* deck = ["d","e"] */
}
```

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
+,-,*,/	arithmetic operators
==, !=, <, >, <=, >=	comparison operators
and, or, not	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:

```
series<string> deck
when do void INIT():
{
  deck = ["R0","R1","R2"]
}
main:
{
  do PRINT("game setup")
  do INIT()
}
```

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.

```
Card card1
Card card2
Card card3
series<Card> deck
Card type1
int i
when do void CREATEDECK():
{
  string red5
  red5 = "R5"
  card1 = Card(red5, true, 5)
  card2 = Card("R6", false, 6)
  card3 = Card("R7", true, 7)
  deck = [card1, card2, card3]
  for  (i = 0; i < deck.size(); i = i + 1):
  {
          type1 = deck[i]
    do PRINT(type1.type)
    do PRINT(type1.faceup)
    do PRINT(type1.value)
  }
}
main:
{
  do CREATEDECK()
}
```

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
```

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

For loops can be nested:

```
int i
int j
for (i = 0 ; i < 5; i = i + 1):
{
    for (j = 0 ; j < 5; j = j + 1):
    {
        do PRINT(i)
        do PRINT(j)
    }
}
```

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 `ACTION`s 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:

```
when do void PLAY():
{
    int i
    for (i = 0; i < 10 ; i = j + 1): /*j not defined*/
    {
    do PRINT("uh oh failure")
    }
}
main:
{
    do PLAY()
}
```

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 includign Ocamlyacc and OCammllex 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@us
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@
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'ofhttps : //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 4c75c226677cf4d8e6eaeb10a892d385becdf975 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$_{m}aster'intotrue_{m}aster$

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$_{m}aster$

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@
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$_{m}aster'intotrue_{m}aster$

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$_{o}bjects_{n}ew$

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@
Date: Mon Apr 26 00:36:46 2021 -0400

merged master$_{2}andlocal/actionstuff$

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@u
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$_{callforgetattributefunctioncalls}$

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$_{printexpression}$

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$_{f}$*uncinstatementspassingactioncasemovedbuilderorder*
commit a8fb65f13436f6840005868dec6fcab8316542c3 Merge: cf288bf 9e968e7 Author: junebug <47231340+junebug111@us
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 a88e793235969fafe7340d24e18925729d81f4de 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$_{l}ist/block$

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$_{d}eclabovecheck_{e}xpr$

commit ede907249f974e36bfa56cc6e3325a3b36e32146 Author: tiffenychen <tc2963@columbia.edu> Date: Fri Apr 23 14:05:26 2021 -0400

added in create action$_{d}ecls_{m}apinsemantunboundfind_{a}cterror$

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$_{b}lock_{p}arse'intoaction_{d}ecls_{n}ew$

commit 2d89f3f3b06b5c25c09bd011c42c53a8442e1ba6 Author: Caitlyn Chen <caitlyn.chen8@gmail.com> Date: Fri Apr 23 13:15:08 2021 -0400

fixed s/r conflicts

commit 2ddbb490dc860409adf17e66659d937a63ae2f2f 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$_{d}ecl'intoaction_{d}ecls_{n}ew$

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 Dimofte <mara.dimofte123@gmail.com> Date: Fri Apr 23 01:17:13 2021 -0400

Merge branch 'master$'_2 of https://github.com/AllHandsOnDeck-PLT/AHODintomaster_2$

commit dbc1c90c0a1c6d4213c73c0d58e3ceae8b8078cb Author: tiffenychen <tc2963@columbia.edu>
Date: Fri Apr 23 01:10:38 2021 -0400

removed whendo

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$_a ctionbody$

commit 4e037864f1b4c9fbcb0fd62db031beffd3e45a60 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$_i nProgress' intomaster_2$

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 Dimofte <mara.dimofte123@gmail.com> Date: Thu Apr 22 22:07:09 2021 -0400

merge stuff

commit f2939a3ec34d3d531717979241ce328e4dcfc1a2 Author: Mara Dimofte <mara.dimofte123@gmail.com>
Date: Thu Apr 22 21:58:06 2021 -0400

for loops working

commit f244454b1cf217cde87d4ff793d4e1a97ea8f822 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$_s tmt$

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: eed14df 2f6cd92 Author: junebug <47231340+junebug111@us
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 c7b9c6aed7f9dc5794e03b414f9dd6f4fbc7aeae 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 771165701fb2b40a7751ddfb743fa775e69fce74 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@u 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 77ddde105124825c069a1a87d76a97bf7a7b5398 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$_stmtinsemant$

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 38b9e46b335ad7dfb793ab7c83b3bbd5e023a5c1 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$_trd$

Resolved Unbound trd error class$_decltupleerror$

commit 1e972e3216aaab37b13bc94edbbbcf5bc7a67042 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 12:36:13 2021 -0400

resolved class$_declerror, thinkmaybeduetowaydefinedfstsndtrdthatmade3tuplesmandatory$

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$_decl_list$

commit 7d752a64e9fd9a212db5402e472f2613f20876e9 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 04:38:58 2021 -0400

ast support for class$_declbutunboundvaluetrderror$

commit 87f8dc8c9e83a83d1dd412ee2c922f072f7f5973 Merge: 6881f89 e70c2dd Author: junebug <47231340+junebug111@u
Date: Fri Mar 19 01:13:52 2021 -0400

resolved merge conflict

commit 6881f896806613f8c1a7c394cc0fe0c63969c688 Author: junebug <47231340+junebug111@users.noreply.github.com>
Date: Fri Mar 19 01:11:51 2021 -0400

working attr$_decl$

commit e70c2dd949d8fd3854f9b5826e0e2b09469f2fe4 Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 00:55:00 2021 -0400

decls tuple for action$_declandhelper_decl$

commit f172c59992aa7fbaeb553c97e7003c7c4169f2fe Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 19 00:48:10 2021 -0400

added in action$_declastsupport$

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 45fab9306bbcfd24efafa8178628e0f1eeadaf8b 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$_*$

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$_branch$

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 0fffdc5082b90ccfd1d29bfe7d1e476a1ec20b4d Author: Caitlyn Chen <caitlyn.chen8@gmail.com>
Date: Fri Mar 5 01:24:50 2021 -0500

dotted$_rangeast$

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

comprehesion 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$_{l}iteral$

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 5cdd5a0b68fadb32a93b0c8698178b1dd9a22395 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$_{d}eclstmt_{b}lockandclass_{b}lock$

commit 78e6c82bbb0a3c0c0dc4e0a1ef94f3179750b880 Merge: c6ed764 efbb1ae 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 efbb1ae4a373086f6d831b8d6162d6870c19b231 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$_{l}ist, args_{l}ist$

commit 7c82fe6528c8f7ac6d9d8165fad8983e29a612d6 Author: Mara Dimofte <mara.dimofte123@gmail.com> Date: Wed Feb 24 11:09:40 2021 -0500

Modified clas, params$_{l}ist, args_{l}ist$

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 122d1c03b74cd279bb835cdbba9d893c0b872268 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

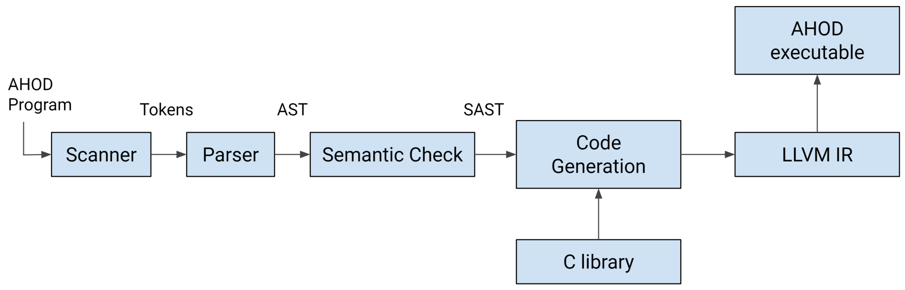
initial v2

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.mll)

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 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)

```
main:
{
    float b
    b = 5.0
    if 3<5:
    {
        for (a = 0 ; a < 5; a = a + 1):
        {
            b = b + 1.0
            do PRINT(b)
        }
    }
}
```

AHOD Code 1 ll file:

```
; ModuleID = 'AHOD'
source_filename = "AHOD"

@str = global [4 x i8] c"%s\0A\00"
@str.1 = global [4 x i8] c"%d\0A\00"
@str.2 = global [4 x i8] c"%g\0A\00"
@str.3 = global [4 x i8] c"%d\0A\00"
@a = global i32 0

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

declare { i8*, i32 } @playercall(i8*, i32)

declare i8* @getplayername({ i8*, i32 })

declare i32 @getplayerscore({ i8*, i32 })

declare { i8*, i1, i32 } @cardcall(i8*, i1, i32)

declare i8* @getcardtype({ i8*, i1, i32 })

declare i1 @getcardfaceup({ i8*, i1, i32 })

declare i32 @getcardvalue({ i8*, i1, i32 })

define i32 @main() {
entry:
  %b = alloca double
  store double 5.000000e+00, double* %b
  br i1 true, label %then, label %else

merge:                                            ; preds = %else, %merge6
  ret i32 0

then:                                             ; preds = %entry
  store i32 0, i32* @a
  br label %while

while:                                            ; preds = %while_body, %then
  %a4 = load i32, i32* @a
  %tmp5 = icmp slt i32 %a4, 5
  br i1 %tmp5, label %while_body, label %merge6

while_body:                                       ; preds = %while
  %b1 = load double, double* %b
  %tmp = fadd double %b1, 1.000000e+00
  store double %tmp, double* %b
  %b2 = load double, double* %b
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str.2, i32 0, i32 0), double %b2)
  %a = load i32, i32* @a
  %tmp3 = add i32 %a, 1
  store i32 %tmp3, i32* @a
  br label %while

merge6:                                           ; preds = %while
  br label %merge

else:                                             ; preds = %entry
  br label %merge
}

define void @series_pushbool({ i32*, i1* }*, i1) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %val_alloc = alloca i1
  store i1 %1, i1* %val_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_load_2 = load i1*, i1** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i1, i1* %val_alloc
  store i1 %val, i1* %series_next_el_ptr
  ret void
}

define void @series_pushint({ i32*, i32* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %val_alloc = alloca i32
  store i32 %1, i32* %val_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_load_2 = load i32*, i32** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i32, i32* %val_alloc
  store i32 %val, i32* %series_next_el_ptr
  ret void
}

define void @series_pushfloat({ i32*, double* }*, double) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %val_alloc = alloca double
  store double %1, double* %val_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_load_2 = load double*, double** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load double, double* %val_alloc
  store double %val, double* %series_next_el_ptr
  ret void
}

define void @series_pushstr({ i32*, i8** }*, i8*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %val_alloc = alloca i8*
  store i8* %1, i8** %val_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_load_2 = load i8**, i8*** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i8*, i8** %val_alloc
  store i8* %val, i8** %series_next_el_ptr
  ret void
}

define void @series_pushcard({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i1, i32 }
  store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
  store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
  ret void
}

define void @series_pushplayer({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i32 }
  store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
  store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
  ret void
}

define i1 @series_getbool({ i32*, i1* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %array_load = load i1*, i1** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
  %series_el_ptr1 = load i1, i1* %series_el_ptr
  ret i1 %series_el_ptr1
}

define i32 @series_getint({ i32*, i32* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %array_load = load i32*, i32** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
  %series_el_ptr1 = load i32, i32* %series_el_ptr
  ret i32 %series_el_ptr1
}

define double @series_getfloat({ i32*, double* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %array_load = load double*, double** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
  %series_el_ptr1 = load double, double* %series_el_ptr
  ret double %series_el_ptr1
}

define i8* @series_getstr({ i32*, i8** }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %array_load = load i8**, i8*** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
  %series_el_ptr1 = load i8*, i8** %series_el_ptr
  ret i8* %series_el_ptr1
}

define { i8*, i1, i32 } @series_getcard({ i32*, { i8*, i1, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr
  ret { i8*, i1, i32 } %series_el_ptr1
}

define { i8*, i32 } @series_getplayer({ i32*, { i8*, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
  ret { i8*, i32 } %series_el_ptr1
}

define i32 @series_sizebool({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeint({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizefloat({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizestr({ i32*, i8** }*) {
```

```
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizecard({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeplayer({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i1 @series_popbool({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_arr_load = load i1*, i1** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i1, i1* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i1 %series_arry_next_element
}

define i32 @series_popint({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_arr_load = load i32*, i32** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i32, i32* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i32 %series_arry_next_element
}

define double @series_popfloat({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_arr_load = load double*, double** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load double, double* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret double %series_arry_next_element
}

define i8* @series_popstr({ i32*, i8** }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_arr_load = load i8**, i8*** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i8*, i8** %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i8* %series_arry_next_element
}

define { i8*, i1, i32 } @series_popcard({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i
  %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i1, i32 } %series_arry_next_element
}

define { i8*, i32 } @series_popplayer({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i32 } %series_arry_next_element
}
```

7.2.2 AHOD Code 2

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

```
when do series<Card> CREATEPILE():
{
    Card card1
    card1 = Card("R5", true, 5)
    return [card1]
}
main:
{
    series<Card> cards
    Card card
    cards = do CREATEPILE()
    card = cards[0]
    do PRINT(card.type)
    do PRINT(card.faceup)
}
```

AHOD Code 2 corresponding ll file:

```
; ModuleID = 'AHOD'
source_filename = "AHOD"

@str = global [4 x i8] c"%s\0A\00"
@str.1 = global [4 x i8] c"%d\0A\00"
@str.2 = global [4 x i8] c"%g\0A\00"
@str.3 = global [4 x i8] c"%d\0A\00"
@str.4 = private unnamed_addr constant [3 x i8] c"R5\00"

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

declare { i8*, i32 } @playercall(i8*, i32)

declare i8* @getplayername({ i8*, i32 })

declare i32 @getplayerscore({ i8*, i32 })

declare { i8*, i1, i32 } @cardcall(i8*, i1, i32)

declare i8* @getcardtype({ i8*, i1, i32 })

declare i1 @getcardfaceup({ i8*, i1, i32 })

declare i32 @getcardvalue({ i8*, i1, i32 })

define { i32*, { i8*, i1, i32 }* } @CREATEPILE() {
entry:
  %card1 = alloca { i8*, i1, i32 }
  %cardcall = call { i8*, i1, i32 } @cardcall(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @str.4, i32 0, i32 0), i1 tru
  store { i8*, i1, i32 } %cardcall, { i8*, i1, i32 }* %card1
  %new_series_ptr = alloca { i32*, { i8*, i1, i32 }* }
  %series_size_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr, i32
  %series_size = alloca i32
  store i32 0, i32* %series_size
  store i32* %series_size, i32** %series_size_ptr
  %series = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr, i32 0, i32 1
  %p = alloca { i8*, i1, i32 }, i32 1028
  store { i8*, i1, i32 }* %p, { i8*, i1, i32 }** %series
  %card11 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card1
  call void @series_pushcard({ i32*, { i8*, i1, i32 }* }* %new_series_ptr, { i8*, i1, i32 } %card11)
  %new_series = load { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %new_series_ptr
  ret { i32*, { i8*, i1, i32 }* } %new_series
}

define void @series_pushbool({ i32*, i1* }*, i1) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %val_alloc = alloca i1
  store i1 %1, i1* %val_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_load_2 = load i1*, i1** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i1, i1* %val_alloc
  store i1 %val, i1* %series_next_el_ptr
  ret void
}

define void @series_pushint({ i32*, i32* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %val_alloc = alloca i32
  store i32 %1, i32* %val_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_load_2 = load i32*, i32** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i32, i32* %val_alloc
  store i32 %val, i32* %series_next_el_ptr
  ret void
}

define void @series_pushfloat({ i32*, double* }*, double) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %val_alloc = alloca double
  store double %1, double* %val_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_load_2 = load double*, double** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load double, double* %val_alloc
  store double %val, double* %series_next_el_ptr
  ret void
}
```

```

define void @series_pushstr({ i32*, i8** }*, i8*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %val_alloc = alloca i8*
  store i8* %1, i8** %val_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_load_2 = load i8**, i8*** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i8*, i8** %val_alloc
  store i8* %val, i8** %series_next_el_ptr
  ret void
}

define void @series_pushcard({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i1, i32 }
  store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
  store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
  ret void
}

define void @series_pushplayer({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i32 }
  store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
  store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
  ret void
}

define i1 @series_getbool({ i32*, i1* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %array_load = load i1*, i1** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
  %series_el_ptr1 = load i1, i1* %series_el_ptr
  ret i1 %series_el_ptr1
}

define i32 @series_getint({ i32*, i32* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %array_load = load i32*, i32** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
  %series_el_ptr1 = load i32, i32* %series_el_ptr
  ret i32 %series_el_ptr1
}

define double @series_getfloat({ i32*, double* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %array_load = load double*, double** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
  %series_el_ptr1 = load double, double* %series_el_ptr
  ret double %series_el_ptr1
```

```
}

define i8* @series_getstr({ i32*, i8** }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %array_load = load i8**, i8*** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
  %series_el_ptr1 = load i8*, i8** %series_el_ptr
  ret i8* %series_el_ptr1
}

define { i8*, i1, i32 } @series_getcard({ i32*, { i8*, i1, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr
  ret { i8*, i1, i32 } %series_el_ptr1
}

define { i8*, i32 } @series_getplayer({ i32*, { i8*, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
  ret { i8*, i32 } %series_el_ptr1
}

define i32 @series_sizebool({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeint({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizefloat({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizestr({ i32*, i8** }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizecard({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeplayer({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i1 @series_popbool({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_arr_load = load i1*, i1** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i1, i1* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i1 %series_arry_next_element
}

define i32 @series_popint({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_arr_load = load i32*, i32** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i32, i32* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i32 %series_arry_next_element
}

define double @series_popfloat({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_arr_load = load double*, double** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load double, double* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret double %series_arry_next_element
}

define i8* @series_popstr({ i32*, i8** }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_arr_load = load i8**, i8*** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i8*, i8** %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i8* %series_arry_next_element
}

define { i8*, i1, i32 } @series_popcard({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i
  %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i1, i32 } %series_arry_next_element
}

define { i8*, i32 } @series_popplayer({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i32 } %series_arry_next_element
}

define i32 @main() {
entry:
  %card = alloca { i8*, i1, i32 }
  %cards = alloca { i32*, { i8*, i1, i32 }* }
  %series_size_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %cards, i32 0, i32 0
  %series_size = alloca i32
  store i32 0, i32* %series_size
  store i32* %series_size, i32** %series_size_ptr
  %series.arry = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %cards, i32 0, i32 1
  %p = alloca { i8*, i1, i32 }, i32 1028
  store { i8*, i1, i32 }* %p, { i8*, i1, i32 }** %series.arry
  %CREATEPILE_result = call { i32*, { i8*, i1, i32 }* } @CREATEPILE()
  store { i32*, { i8*, i1, i32 }* } %CREATEPILE_result, { i32*, { i8*, i1, i32 }* }* %cards
  %series_get = call { i8*, i1, i32 } @series_getcard.15({ i32*, { i8*, i1, i32 }* }* %cards, i32 0)
  store { i8*, i1, i32 } %series_get, { i8*, i1, i32 }* %card
  %card1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card
  %getcardtype = call i8* @getcardtype({ i8*, i1, i32 } %card1)
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str, i32 0, i32 0), i8* %getcardty
  %card2 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %card
  %getcardfaceup = call i1 @getcardfaceup({ i8*, i1, i32 } %card2)
  %printf3 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([4 x i8], [4 x i8]* @str.3, i32 0, i32 0), i1 %getcard
  ret i32 0
}

define void @series_pushbool.5({ i32*, i1* }*, i1) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %val_alloc = alloca i1
  store i1 %1, i1* %val_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_load_2 = load i1*, i1** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i1, i1* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i1, i1* %val_alloc
  store i1 %val, i1* %series_next_el_ptr
  ret void
}

define void @series_pushint.6({ i32*, i32* }*, i32) {
```

```
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %val_alloc = alloca i32
  store i32 %1, i32* %val_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_load_2 = load i32*, i32** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i32, i32* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i32, i32* %val_alloc
  store i32 %val, i32* %series_next_el_ptr
  ret void
}

define void @series_pushfloat.7({ i32*, double* }*, double) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %val_alloc = alloca double
  store double %1, double* %val_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_load_2 = load double*, double** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr double, double* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load double, double* %val_alloc
  store double %val, double* %series_next_el_ptr
  ret void
}

define void @series_pushstr.8({ i32*, i8** }*, i8*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %val_alloc = alloca i8*
  store i8* %1, i8** %val_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_load_2 = load i8**, i8*** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr i8*, i8** %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load i8*, i8** %val_alloc
  store i8* %val, i8** %series_next_el_ptr
  ret void
}

define void @series_pushcard.9({ i32*, { i8*, i1, i32 }* }*, { i8*, i1, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i1, i32 }
  store { i8*, i1, i32 } %1, { i8*, i1, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %series_load_2 = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i1, i32 }, { i8*, i1, i32 }* %val_alloc
  store { i8*, i1, i32 } %val, { i8*, i1, i32 }* %series_next_el_ptr
  ret void
}

define void @series_pushplayer.10({ i32*, { i8*, i32 }* }*, { i8*, i32 }) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %val_alloc = alloca { i8*, i32 }
  store { i8*, i32 } %1, { i8*, i32 }* %val_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_load_2 = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_load_2, i32 %series_size
  %next_size = add i32 %series_size, 1
  store i32 %next_size, i32* %series_size_ptr
  %val = load { i8*, i32 }, { i8*, i32 }* %val_alloc
  store { i8*, i32 } %val, { i8*, i32 }* %series_next_el_ptr
  ret void
}

define i1 @series_getbool.11({ i32*, i1* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %array_load = load i1*, i1** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i1, i1* %array_load, i32 %idx_load
  %series_el_ptr1 = load i1, i1* %series_el_ptr
  ret i1 %series_el_ptr1
}

define i32 @series_getint.12({ i32*, i32* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %array_load = load i32*, i32** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i32, i32* %array_load, i32 %idx_load
  %series_el_ptr1 = load i32, i32* %series_el_ptr
  ret i32 %series_el_ptr1
}

define double @series_getfloat.13({ i32*, double* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %array_load = load double*, double** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr double, double* %array_load, i32 %idx_load
  %series_el_ptr1 = load double, double* %series_el_ptr
  ret double %series_el_ptr1
}

define i8* @series_getstr.14({ i32*, i8** }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %array_load = load i8**, i8*** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr i8*, i8** %array_load, i32 %idx_load
  %series_el_ptr1 = load i8*, i8** %series_el_ptr
  ret i8* %series_el_ptr1
}

define { i8*, i1, i32 } @series_getcard.15({ i32*, { i8*, i1, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i32
  %array_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_el_ptr
  ret { i8*, i1, i32 } %series_el_ptr1
}

define { i8*, i32 } @series_getplayer.16({ i32*, { i8*, i32 }* }*, i32) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %idx_alloc = alloca i32
  store i32 %1, i32* %idx_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_ptr_2 = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %array_load = load { i8*, i32 }*, { i8*, i32 }** %series_ptr_2
  %idx_load = load i32, i32* %idx_alloc
  %series_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %array_load, i32 %idx_load
  %series_el_ptr1 = load { i8*, i32 }, { i8*, i32 }* %series_el_ptr
  ret { i8*, i32 } %series_el_ptr1
}

define i32 @series_sizebool.17({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeint.18({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizefloat.19({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizestr.20({ i32*, i8** }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizecard.21({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i32 @series_sizeplayer.22({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  ret i32 %series_size
}

define i1 @series_popbool.23({ i32*, i1* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i1* }*
  store { i32*, i1* }* %0, { i32*, i1* }** %series_ptr_alloc
  %series_load = load { i32*, i1* }*, { i32*, i1* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 1
  %series_arr_load = load i1*, i1** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i1* }, { i32*, i1* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i1, i1* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i1, i1* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i1 %series_arry_next_element
}

define i32 @series_popint.24({ i32*, i32* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i32* }*
  store { i32*, i32* }* %0, { i32*, i32* }** %series_ptr_alloc
  %series_load = load { i32*, i32* }*, { i32*, i32* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 1
  %series_arr_load = load i32*, i32** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i32* }, { i32*, i32* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i32, i32* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i32, i32* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i32 %series_arry_next_element
}

define double @series_popfloat.25({ i32*, double* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, double* }*
  store { i32*, double* }* %0, { i32*, double* }** %series_ptr_alloc
  %series_load = load { i32*, double* }*, { i32*, double* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 1
  %series_arr_load = load double*, double** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, double* }, { i32*, double* }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr double, double* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load double, double* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret double %series_arry_next_element
}

define i8* @series_popstr.26({ i32*, i8** }*) {
entry:
  %series_ptr_alloc = alloca { i32*, i8** }*
  store { i32*, i8** }* %0, { i32*, i8** }** %series_ptr_alloc
  %series_load = load { i32*, i8** }*, { i32*, i8** }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 1
  %series_arr_load = load i8**, i8*** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, i8** }, { i32*, i8** }* %series_load, i32 0, i32 0
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr i8*, i8** %series_arr_load, i32 %dec_size
  %series_arry_next_element = load i8*, i8** %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret i8* %series_arry_next_element
}

define { i8*, i1, i32 } @series_popcard.27({ i32*, { i8*, i1, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i1, i32 }* }*
  store { i32*, { i8*, i1, i32 }* }* %0, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i1, i32 }* }*, { i32*, { i8*, i1, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32 0, i
  %series_arr_load = load { i8*, i1, i32 }*, { i8*, i1, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i1, i32 }* }, { i32*, { i8*, i1, i32 }* }* %series_load, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i1, i32 }, { i8*, i1, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i1, i32 }, { i8*, i1, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i1, i32 } %series_arry_next_element
}

define { i8*, i32 } @series_popplayer.28({ i32*, { i8*, i32 }* }*) {
entry:
  %series_ptr_alloc = alloca { i32*, { i8*, i32 }* }*
  store { i32*, { i8*, i32 }* }* %0, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_load = load { i32*, { i8*, i32 }* }*, { i32*, { i8*, i32 }* }** %series_ptr_alloc
  %series_arr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32 1
  %series_arr_load = load { i8*, i32 }*, { i8*, i32 }** %series_arr_ptr
  %series_size_ptr_ptr = getelementptr inbounds { i32*, { i8*, i32 }* }, { i32*, { i8*, i32 }* }* %series_load, i32 0, i32
  %series_size_ptr = load i32*, i32** %series_size_ptr_ptr
  %series_size = load i32, i32* %series_size_ptr
  %dec_size = sub i32 %series_size, 1
  %series_next_el_ptr = getelementptr { i8*, i32 }, { i8*, i32 }* %series_arr_load, i32 %dec_size
  %series_arry_next_element = load { i8*, i32 }, { i8*, i32 }* %series_next_el_ptr
  store i32 %dec_size, i32* %series_size_ptr
  ret { i8*, i32 } %series_arry_next_element
}
```

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 helps 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 dat 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

```
# "make test" Compiles everything and runs the regression tests

.PHONY : test
test : all testall.sh
	./testall.sh

# "make all" builds the executable as well as the "printbig" library designed
# to test linking external code

.PHONY : all
all : AHOD.native playercall.o

# "make microc.native" compiles the compiler
#
# The _tags file controls the operation of ocamlbuild, e.g., by including
# packages, enabling warnings
#
# See https://github.com/ocaml/ocamlbuild/blob/master/manual/manual.adoc

AHOD.native :
	opam config exec -- \
	ocamlbuild -use-ocamlfind AHOD.native

# "make clean" removes all generated files

playercall : playercall.c
	cc -o playercall -DBUILD_TEST playercall.c

.PHONY : clean
clean :
	ocamlbuild -clean
	rm -rf testall.log ocamlllvm *.diff .ll .native *.o
```

scanner.mll

```
(* Ocamllex scanner for AHOD *)
{ open Parse }

let digit = ['0' - '9']
let digits = digit+

rule token = parse
  [' ' '\t' '\r' ] { token lexbuf } (* Whitespace *)
| "/*"     { comment lexbuf }           (* Comments *)
| ('\n' [' ']*)+            { NEWLINE }
| '('      { LPAREN }
| ')'      { RPAREN }
| '{'      { LBRACE }
| '}'      { RBRACE }
| '['      { LSQUARE }
| ']'      { RSQUARE }
| "push"   { SERIESPUSH }
| "pop"    { SERIESPOP }
| "size"   { SERIESSIZE }
| "series" { SERIES }
| "Player" { PLAYER }
| "Card"   { CARD }
| ':'      { COLON }
| ';'      { SEMI }
| ','      { COMMA }
| '+'      { PLUS }
| '-'      { MINUS }
| '*'      { MULT }
| '/'      { DIVIDE }
| '.'      { DOT }
| '='      { ASSIGN }
| "=="     { EQ }
| "!="     { NEQ }
| "<"      { LT }
| ">"      { GT }
| "<="     { LEQ }
| ">="     { GEQ }
| "and"    { AND }
| "or"     { OR }
| "not"    { NOT }
| "if"     { IF }
| "else"   { ELSE }
| "for"    { FOR }
| "while"  { WHILE }
| "when"   { WHEN }
| "do"     { DO }
| "return" { RETURN }
| "int"    { INT }
| "bool"   { BOOL }
| "string" { STRING }
| "float"  { FLOAT }
| "void"   { VOID }
| "true"   { BLIT(true)  }
| "false"  { BLIT(false) }
| "PRINT"   { PRINT }
| "main"   { MAIN }
| digits as lxm { ILIT(int_of_string lxm) }
| ['-']? (digits '.'  digit* ( ['e' 'E'] ['+' '-']? digits )?) as lxm { FLIT(lxm) }
| '"' ([' ' '-'!' '#'-'&' '('-'[' ']'-'~' 'a'-'z'' ' 'A'-'Z' '0'-'9']* as lxm) '"' { SLIT(lxm) }
| ['a'-'z']['a'-'z' '0'-'9' '_']*                     as lxm { ID(lxm) }
| ['A'-'Z']['A'-'Z' '0'-'9' '_']*                     as actionID { ACTIONID(actionID) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char)) }

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

parse.mly

```
%{
open Ast
%}

%token LPAREN RPAREN LBRACE RBRACE LSQUARE RSQUARE SERIESSIZE SERIESPUSH SERIESPOP SERIES CARD PLAYER COLON SEMI COMMA PLUS
%token NOT EQ NEQ LT LEQ GT GEQ AND OR
%token RETURN IF ELSE FOR WHILE INT BOOL FLOAT VOID STRING WHEN DO MAIN PRINT
%token CEND
%token <int> ILIT
%token <bool> BLIT
%token <string> ID ACTIONID FLIT SLIT
%token EOF

%start program
%type <Ast.program> program

%nonassoc FOR
%right ASSIGN
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%nonassoc DO
%nonassoc COLON
%left PLUS MINUS
%left MULT DIVIDE
%right NOT

%%

program:
    decls main_decl EOF     { (fst $1, snd $1, $2) }

decls:
    | /*nothing*/           { ([], [])                        }
    | decls global_decl     { (List.rev ($2::fst $1), snd $1) }
    | decls action_decl     { (fst $1, List.rev ($2::snd $1)) }

cend_opt: /*comment */
    | /*nothing */      { Noexpr }
    | CEND              { Noexpr }

global_decl:
    typ ID cend_opt NEWLINE { ($1, $2)}

main_decl:
    MAIN COLON cend_opt NEWLINE LBRACE cend_opt NEWLINE locals_list stmt_wrap RBRACE cend_opt NEWLINE {{
    mtyp = Void;
    mparams = [];
    mlocals = $8;
    mbody = [$9] }}

action_decl:
    WHEN DO typ ACTIONID LPAREN params_list_opt RPAREN COLON cend_opt NEWLINE LBRACE cend_opt NEWLINE locals_list stmt_wrap
    {{
      atyp = $3;
      aname = $4;
      aparams = List.rev $6;
      alocals = List.rev $14;
      abody = [$15] }}

params_list_opt:
    | params_list    {$1}
    | /*Nothing*/    {[]}

params_list:
    | param                         { [$1]   }
    | params_list COMMA param       { $3::$1 }

param:
    | typ ID                        { $1, $2 }

stmt_block:
    | NEWLINE LBRACE cend_opt NEWLINE stmt_list RBRACE cend_opt NEWLINE              { Block(List.rev $5) }

locals_list:
    | /*nothing */                          {[]       }
    | locals_list global_decl               {$2 :: $1 }

stmt_wrap:
    | /*nothing */                          { Block([])          }
    | stmt_list                             { Block(List.rev $1) }

```

```
stmt_list:
    | stmt                              { [$1]     }
    | stmt_list stmt                    { $2 :: $1 }

stmt:
    | stmt_block                                                       { $1                    }
    | expr cend_opt NEWLINE                                            { Expr $1               }
    | RETURN expr_opt NEWLINE                                          { Return $2             }
    | if_stmt                                                          { $1                    }
    | FOR LPAREN expr SEMI expr SEMI expr RPAREN COLON cend_opt stmt_block  { For($3, $5, $7, $11)   }
    | WHILE expr COLON cend_opt stmt_block                             { While($2, $5)         }
    | ID DOT SERIESPUSH LPAREN expr RPAREN cend_opt NEWLINE            { SeriesPush($1, $5)    }

if_stmt:
    | IF expr COLON cend_opt stmt_block else_block_opt   { If($2, $5, $6) }

else_block_opt:
    | /* nothing */      { Block([]) }
    | else_block         { $1        }

else_block:
    | ELSE COLON cend_opt stmt_block     { $4 }

typ:
    | INT                                        { Int       }
    | BOOL                                       { Bool      }
    | FLOAT                                      { Float     }
    | STRING                                     { String    }
    | VOID                                       { Void      }
    | SERIES LT typ GT                           { Series($3)}
    | PLAYER                                     { Player    }
    | CARD                                       { Card      }

expr:
    | ILIT                              { Iliteral($1) }
    | FLIT                              { Fliteral($1) }
    | BLIT                              { Bliteral($1) }
    | SLIT                                     { Sliteral($1) }
    | ID                                { Id($1) }
    | ID ASSIGN expr                    { Assign($1, $3) }
    | LSQUARE args_list_opt RSQUARE     { Seriesliteral($2) }
    | ID LSQUARE expr RSQUARE           { SeriesGet($1, $3) }
    | ID DOT SERIESSIZE LPAREN RPAREN   { SeriesSize($1)}
    | ID DOT SERIESPOP LPAREN RPAREN    { SeriesPop($1)}
    | CEND                              { Noexpr }
    | expr PLUS   expr                  { Binop($1, Add,     $3) }
    | expr MINUS  expr                  { Binop($1, Sub,     $3) }
    | expr MULT   expr                  { Binop($1, Mult,    $3) }
    | expr DIVIDE expr                  { Binop($1, Div,     $3) }
    | expr AND    expr                  { Binop($1, And,     $3) }
    | expr OR     expr                  { Binop($1, Or,      $3) }
    | expr EQ     expr                  { Binop($1, Equal,   $3) }
    | expr NEQ    expr                 { Binop($1, Neq,    $3)   }
    | expr LT     expr                 { Binop($1, Less,    $3) }
    | expr LEQ    expr                 { Binop($1, Leq,     $3) }
    | expr GT     expr                 { Binop($1, Greater, $3) }
    | expr GEQ    expr                 { Binop($1, Geq,     $3) }
    | MINUS expr %prec NOT             { Unop(Neg, $2)          }
    | NOT expr                         { Unop(Not, $2)          }
    | call_print                       { $1 }
    | call_class                       { $1 }
    | call_action                      { $1 }
    | call_attr                        { $1 }

args_list_opt:
    | /*nothing */               { []          }
    | args_list                  { List.rev $1 }

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

call_print:
    | DO PRINT LPAREN expr RPAREN        { PrintCall($4) }

call_action:
    | DO ACTIONID LPAREN args_list_opt RPAREN        { ActionCall($2, $4) }

call_class:
    | PLAYER LPAREN args_list_opt RPAREN            { PlayerClassCall($3) }
    | CARD LPAREN args_list_opt RPAREN              { CardClassCall($3   ) }

call_attr:
    | ID DOT ID      { AttrCall($1, $3) }

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

ast.ml

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

type uop = Neg | Not

type typ = Int | Float | Bool | String | Void | Series of typ | Player | Card

type bind = typ * string


type expr =
  | Iliteral of int
  | Fliteral of string
  | Bliteral of bool
  | Sliteral of string
  | Seriesliteral of expr list
  | PrintCall of expr
  | ActionCall of string * expr list
  | Id of string
  | Assign of string * expr
  (* | AttrAssign of string * string * expr *)
  | Binop of expr * op * expr
  | Unop of uop * expr
  | PlayerClassCall of expr list
  | CardClassCall of expr list
  | AttrCall of string * string
  | SeriesGet of string * expr
  | SeriesSize of string
  | SeriesPop of string
  | Noexpr


type stmt =
  | Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  (* | ForLit of string * expr * stmt  *)
  | While of expr * stmt
  | SeriesPush of string * expr
  | Nostmt

type main_decl = {
  mtyp : typ;
  mparams : bind list;
  mlocals : bind list;
  mbody: stmt list;
}

type action_decl = {
  (* entitytyp : typ;
  entityid : string; *)
  atyp : typ;
  aname : string;
  aparams : bind list;
  alocals : bind list;
  abody: stmt list;
}

type program = bind list * action_decl list * main_decl

(*  Pretty-printing functions *)
let string_of_op = function
```

```
    Add -> "+"
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">"
  | Geq -> ">="
  | And -> "and"
  | Or -> "or"

let string_of_uop = function
  Neg -> "-"
  | Not -> "!"

let rec string_of_typ = function
    Int -> "int"
  | Bool -> "bool"
  | Float -> "float"
  | String -> "string"
  | Void -> "void"
  | Series x -> "series<" ^ (string_of_typ x) ^ ">"
  | Player -> "player"
  | Card -> "card"

let rec string_of_expr = function
    Iliteral(l) -> string_of_int l
  | Fliteral(l) -> l
  | Bliteral(true) -> "true"
  | Bliteral(false) -> "false"
  | Sliteral(l) -> l
  | Seriesliteral(_) -> "series_literal"
  | PrintCall(e) -> "do" ^ "PRINT" ^ "(" ^ string_of_expr e ^ ")"
  | ActionCall(f, el) ->
  "do " ^ f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | PlayerClassCall(el) ->
    "Player (" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | CardClassCall(el) ->
    "Card (" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | AttrCall(cls,fld) -> cls ^ "." ^ fld
  | Id(s) -> s
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  (* | AttrAssign(s, v, e) ->  s ^ "." ^ v ^ " = " ^ string_of_expr e *)
  | Binop(e1, o, e2) ->
  string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_expr e2
  | Unop(o, e) -> string_of_uop o ^ string_of_expr e
  | SeriesGet(id, e) ->  id ^ "[" ^ (string_of_expr e) ^ "]"
  | SeriesSize(id) -> "series_size " ^ id
  | SeriesPop(id) -> "series_pop " ^ id
  | Noexpr -> ""
```

```

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

sast.ml

```
open Ast

type sexpr = typ * sx

and sx =
  | SIliteral of int
  | SFliteral of string
  | SBliteral of bool
  | SSliteral of string
  | SSeriesliteral of typ * sexpr list
  | SSeriesGet of typ * string * sexpr
  | SSeriesSize of typ * string
  | SSeriesPop of typ * string
  | SPrintCall of sexpr
  | SActionCall of string * sexpr list
  | SId of string
  | SAssign of string * sexpr
  (* | SAttrAssign of string * string * sexpr *)
  | SBinop of sexpr * op * sexpr
  | SUnop of uop * sexpr
  | SPlayerClassCall of sexpr list
  | SCardClassCall of sexpr list
  | SAttrCall of string * string
  | SNoexpr

type sstmt =
  | SBlock of sstmt list
  | SExpr of sexpr
  | SReturn of sexpr
  | SIf of sexpr * sstmt * sstmt
  | SFor of sexpr * sexpr * sexpr * sstmt
  (* | SForLit of string * sexpr * sstmt  *)
  | SWhile of sexpr * sstmt
  | SSeriesPush of string * sexpr
  | SNostmt

type smain_decl = {
  smtyp : typ;
  smparams : bind list;
  smlocals : bind list;
  smbody: sstmt list;
}

type saction_decl = {
  (* sentitytyp : typ;
  sentityid : string; *)
  saname : string;
  satyp : typ;
  saparams : bind list;
  salocals : bind list;
  sabody: sstmt list;
}

type sprogram = bind list * saction_decl list * smain_decl

(* Pretty-printing functions *)

let rec string_of_sexpr (t, e) =
  "(" ^ string_of_typ t ^ " : " ^ (match e with
    SIliteral(l) -> string_of_int l
  | SFliteral(l) -> l
  | SBliteral(true) -> "true"
  | SBliteral(false) -> "false"
  | SSliteral(l) -> l
  | SSeriesliteral(_) -> "list_literal"
  | SSeriesGet(_, id, e) -> id ^ "[" ^ (string_of_sexpr e) ^ "]"
  | SSeriesSize(_, id) -> "series_size " ^ id
  | SSeriesPop(_, id) -> "series_pop " ^ id
  | SPrintCall(e) -> "do" ^ "PRINT" ^ "(" ^ string_of_sexpr e ^ ")"
  | SActionCall(f, el) ->
  "do " ^f ^ "(" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SPlayerClassCall(el) ->
    "Player (" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SCardClassCall(el) ->
    "Card (" ^ String.concat ", " (List.map string_of_sexpr el) ^ ")"
  | SAttrCall(cls,fld) -> cls ^ "." ^ fld
  | SId(s) -> s
  | SAssign(v, e) -> v ^ " = " ^ string_of_sexpr e
  (* | SAttrAssign(s, v, e) ->  s ^ "." ^ v ^ " = " ^ string_of_sexpr e *)
  | SBinop(e1, o, e2) ->
      string_of_sexpr e1 ^ " " ^ string_of_op o ^ " " ^ string_of_sexpr e2
  | SUnop(o, e) -> string_of_uop o ^ string_of_sexpr e
  | SNoexpr -> "")

let rec string_of_sstmt = function
  SBlock(stmts) -> "{\n" ^ String.concat "" (List.map string_of_sstmt stmts) ^ "\n}"
  | SExpr(exp) -> string_of_sexpr exp ^ "\n"
  | SReturn(exp) -> "return" ^ string_of_sexpr exp ^ "\n"
  | SIf(exp, s1, s2) -> "if " ^ string_of_sexpr exp  ^ ":\n" ^ string_of_sstmt s1 ^ string_of_sstmt s2
  | SFor(e1, e2, e3, s) -> "for (" ^ string_of_sexpr e1 ^ ";" ^  string_of_sexpr e2 ^ ";" ^
                  string_of_sexpr e3 ^ "):\n" ^ string_of_sstmt s
  | SWhile(exp, stmt) -> "while " ^ string_of_sexpr exp ^ ":\n" ^ string_of_sstmt stmt
  | SSeriesPush(id, exp) -> id ^ "." ^ "push" ^ "(" ^ string_of_sexpr exp ^ ")"
  | SNostmt -> ""
```

semant.ml

```
open Ast
open Sast

module StringMap = Map.Make(String)

let check (globals, action_decls, main_decl) =

  (* Verify a list of bindings has no Void types or duplicate names *)
  let check_binds (kind : string) (binds : bind list) =
    List.iter (function
      | (Void, b) -> raise (Failure ("illegal Void " ^ kind ^ " " ^ b))
      | _ -> ()) binds;
    let rec dups = function
        [] -> ()
      | ((_,n1) :: (_,n2) :: _) when n1 = n2 -> raise (Failure ("duplicate " ^ kind ^ " " ^ n1))
      | _ :: t -> dups t
    in dups (List.sort (fun (_,a) (_,b) -> compare a b) binds)
  in

  (**** Check global variables ****)
  check_binds "global" globals;

  let built_in_decls =
    let add_bind map (name, ty) = StringMap.add name {
      atyp = Void;
      aname = name;
      aparams = [(ty, "x")];
      alocals = []; abody = [] } map
    in List.fold_left add_bind StringMap.empty []
  in

  let add_action map ad =
    let built_in_err = "function " ^ ad.aname ^ " may not be defined"
    and dup_err = "duplicate function " ^ ad.aname
    and make_err er = raise (Failure er)
    and n = ad.aname (* Name of the function *)
    in match ad with (* No duplicate functions or redefinitions of built-ins *)
        _ when StringMap.mem n built_in_decls -> make_err built_in_err
      | _ when StringMap.mem n map -> make_err dup_err
      | _ ->  StringMap.add n ad map
  in

  (* Collect all function names into one symbol table *)
```

```
let action_decls_map = List.fold_left add_action built_in_decls action_decls
in

let find_act s =
  try StringMap.find s action_decls_map
  with Not_found -> raise (Failure ("unrecognized action " ^ s))
in

let check_main main =
  (* Make sure no params or locals are void or duplicates *)
  check_binds "mlocals" main.mlocals;

  (* Raise an exception if the given rvalue type cannot be assigned to
  the given lvalue type *)
  let check_assign lvaluet rvaluet err =
    if lvaluet = rvaluet then lvaluet else raise (Failure err)
  in

  (* Build local symbol table of variables for this action *)
  let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
              StringMap.empty (globals @ main.mlocals)
  in

  (* Return a variable from our local symbol table *)
  let type_of_identifier s =
    try StringMap.find s symbols
    with Not_found -> raise (Failure ("undeclared identifier " ^ s))
  in

  let check_series_type id =
    match (type_of_identifier id) with
      Series t -> t
    | t -> raise (Failure ("check series type error, typ: " ^ string_of_typ t))
  in

  let rec check_expr = function
    (*need to figure out typ, if name is defined*)
      | PlayerClassCall(pparams) as pcall ->
        let constructor_len = 2 in
        if List.length pparams != constructor_len then
        raise (Failure ("expecting " ^ string_of_int constructor_len ^
                          " arguments in " ^ string_of_expr pcall))
        else (Player, SPlayerClassCall(List.map check_expr pparams))
      | CardClassCall(pparams) as ccall ->
        let constructor_len = 3 in
        if List.length pparams != constructor_len then
        raise (Failure ("expecting " ^ string_of_int constructor_len ^
                          " arguments in " ^ string_of_expr ccall))
        else (Card, SCardClassCall(List.map check_expr pparams))
      (* | AttrAssign(objname, attr, e) -> (Void, SAttrAssign(objname, attr, check_expr e)) *)
      | AttrCall(objname, attr) ->
        (match attr with
        "name" -> (String, SAttrCall(objname, attr))
        | "score" -> (Int, SAttrCall(objname, attr))
        | "type" -> (String, SAttrCall(objname, attr))
        | "faceup" -> (Bool, SAttrCall(objname, attr))
        | "value" -> (Int, SAttrCall(objname, attr))
        | _ -> raise (Failure ("attribute not found")))
      | Sliteral s -> (String, SSliteral(s))
      | Iliteral i -> (Int, SIliteral(i))
      | Fliteral f -> (Float, SFliteral(f))
      | Bliteral b -> (Bool, SBliteral(b))
      | Noexpr     -> (Void, SNoexpr)
      | Id s       -> (type_of_identifier s, SId s)
      | Assign(var, e) ->
        let lt = type_of_identifier var
        and (rt, e') = check_expr e in
        let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
              string_of_typ rt
        in (check_assign lt rt err, SAssign(var, (rt, e')))
        | Unop(op, e) as ex ->
          let (t, e') = check_expr e in
          let ty = match op with
            Neg when t = Int || t = Float -> t
          | Not when t = Bool -> Bool
          | _ -> raise (Failure ("illegal unary operator " ^
                                  string_of_uop op ^ string_of_typ t ^
                                  " in " ^ string_of_expr ex))
          in (ty, SUnop(op, (t, e')))
        | Binop(e1, op, e2) ->
            let (t1, e1') = check_expr e1
            and (t2, e2') = check_expr e2 in
            (* All binary operators require operands of the same type *)
            let same = t1 = t2 in
            (* Determine expression type based on operator and operand types *)
            let ty = match op with
              Add | Sub | Mult | Div when same && t1 = Int   -> Int
            | Add | Sub | Mult | Div when same && t1 = Float -> Float
            | Equal | Neq            when same               -> Bool
            | Less | Leq | Greater | Geq
                      when same && (t1 = Int || t1 = Float) -> Bool
            | And | Or when same && t1 = Bool -> Bool
            | _ -> raise (
              Failure ("illegal binary operator of " ^ string_of_op op ^"for " ^ string_of_typ t1 ^ " and " ^ string_of_t
            in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | PrintCall(e) -> (Void,SPrintCall(check_expr e))
      | ActionCall(aname, args) as acall ->
        let ad = find_act aname in
        let param_length = List.length ad.aparams in
        if List.length args != param_length then
          raise (Failure ("expecting " ^ string_of_int param_length ^
                          " arguments in " ^ string_of_expr acall))
        else let check_call (at, _) e =
          let (et, e') = check_expr e in
          let err = "illegal argument found " ^ string_of_typ et ^
            " expected " ^ string_of_typ at ^ " in " ^ string_of_expr e
          in (check_assign at et err, e')
        in
        let args' = List.map2 check_call ad.aparams args
        in (ad.atyp, SActionCall(aname, args'))
      | Seriesliteral vals ->
          let (t', _) = check_expr (List.hd vals) in
          let map_func lit = check_expr lit in
          let vals' = List.map map_func vals in
          (Series t', SSeriesliteral(t', vals'))
      | SeriesGet (var, e) ->
          let (t, e') = check_expr e in
          let ty = match t with
              Int -> Int
              | _ -> raise (Failure ("list_get index must be integer, not " ^ string_of_typ t))
          in let list_type = check_series_type var
          in (list_type, SSeriesGet(list_type, var, (ty, e')))
      | SeriesSize var ->
        (Int, SSeriesSize(check_series_type var, var))
      | SeriesPop var ->
        let series_type = check_series_type var
        in (series_type, SSeriesPop(series_type, var))
  in

  let check_match_series_type_expr l e =
    let (t', e') as e'' = check_expr e
     in let err = "series type and expression type do not match " ^ (string_of_typ t') ^ ", " ^ (string_of_sexpr e'')
     in if t' != (check_series_type l) then raise (Failure err) else (t', e')
   in

  let check_bool_expr e =
    let (t', e') = check_expr e
      and err = "expected Boolean expression in " ^ string_of_expr e
      in if t' != Bool then raise (Failure err) else (t', e')
  in

  let rec check_stmt = function (*currently only supports one input -- support for map *)
    Expr e -> SExpr (check_expr e)
    | SeriesPush (var, e) ->
      let _ = check_series_type var in
      SSeriesPush(var, check_match_series_type_expr var e)
    | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt b2)
    | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
    | For(e1, e2, e3, st) ->
      SFor(check_expr e1, check_bool_expr e2, check_expr e3, check_stmt st)
    | Return e -> let (t, e') = check_expr e in
      if t = main.mtyp then SReturn (t, e')
      else raise (
              Failure ("Main does not support return of " ^ string_of_typ t ^ " expected " ^
                  string_of_typ main.mtyp ^ " in " ^ string_of_expr e))
    | Block sl ->
```

```
        let rec check_stmt_list = function
           [Return _ as s] -> [check_stmt s]
          | Return _ :: _   -> raise (Failure "nothing may follow a return")
          | Block sl :: ss  -> check_stmt_list (sl @ ss) (* Flatten blocks *)
          | s :: ss         -> check_stmt s :: check_stmt_list ss
          | []              -> []
        in  SBlock(check_stmt_list sl) (*(List.map check_stmt_list sl)*)
      | Nostmt -> SNostmt
      in
    {
      smtyp = main.mtyp;
      smparams = main.mparams;
      smlocals  = main.mlocals;
      smbody = match check_stmt (Block main.mbody) with
      SBlock(sl) -> sl
          | _ -> raise (Failure ("internal error: block didn't become a block?"))
    }
  in
  let check_action act =
    (* Make sure no params or locals are void or duplicates *)
    check_binds "aparams" act.aparams;
    check_binds "alocals" act.alocals;

    (* Raise an exception if the given rvalue type cannot be assigned to
    the given lvalue type *)
    let check_assign lvaluet rvaluet err =
      if lvaluet = rvaluet then lvaluet else raise (Failure err)
    in

    (* Build local symbol table of variables for this action *)
    let symbols = List.fold_left (fun m (ty, name) -> StringMap.add name ty m)
                    StringMap.empty (globals @ act.aparams @ act.alocals )
    in

    (* Return a variable from our local symbol table *)
    let type_of_identifier s =
      try StringMap.find s symbols
      with Not_found -> raise (Failure ("undeclared identifier " ^ s))
    in

   (* Check if id is a series and return series type *)
   let check_series_type id =
    match (type_of_identifier id) with
      Series t -> t
    | t -> raise (Failure ("check series type error, typ: " ^ string_of_typ t))
  in

  let rec check_expr = function
    | PlayerClassCall(pparams) ->  (Player, SPlayerClassCall(List.map check_expr pparams))
    | CardClassCall(pparams) ->  (Card, SCardClassCall(List.map check_expr pparams))
    | AttrCall(objname, attr) ->
      (match attr with
      "name" -> (String, SAttrCall(objname, attr))
      | "score" -> (Int, SAttrCall(objname, attr))
      | "type" -> (String, SAttrCall(objname, attr))
      | "faceup" -> (Bool, SAttrCall(objname, attr))
      | "value" -> (Int, SAttrCall(objname, attr))
      | _ -> raise (Failure ("attribute not found")))
    (* | AttrAssign(objname, attr, e) -> (Void, SAttrAssign(objname, attr, check_expr e)) *)
                | Sliteral s -> (String, SSliteral(s))
                | Iliteral i -> (Int, SIliteral(i))
                | Fliteral f -> (Float, SFliteral(f))
    | Bliteral b -> (Bool, SBliteral(b))
    | Noexpr     -> (Void, SNoexpr)
    | Id s       -> (type_of_identifier s, SId s)
    | Assign(var, e) ->
      let lt = type_of_identifier var
      and (rt, e') = check_expr e in
      let err = "illegal assignment " ^ string_of_typ lt ^ " = " ^
             string_of_typ rt
      in (check_assign lt rt err, SAssign(var, (rt, e')))
    | Unop(op, e) as ex ->
          let (t, e') = check_expr e in
          let ty = match op with
            Neg when t = Int || t = Float -> t
          | Not when t = Bool -> Bool
          | _ -> raise (Failure ("illegal unary operator " ^
                                 string_of_uop op ^ string_of_typ t ^
                                 " in " ^ string_of_expr ex))
          in (ty, SUnop(op, (t, e')))
    | Binop(e1, op, e2) ->
          let (t1, e1') = check_expr e1
          and (t2, e2') = check_expr e2 in
          let same = t1 = t2 in
          let ty = match op with
            Add | Sub | Mult | Div when same && t1 = Int   -> Int
            | Add | Sub | Mult | Div when same && t1 = Float -> Float
            | Equal | Neq            when same               -> Bool
            | Less | Leq | Greater | Geq
                      when same && (t1 = Int || t1 = Float) -> Bool
            | And | Or when same && t1 = Bool -> Bool
            | _ -> raise (
              Failure ("illegal binary operator of " ^ string_of_op op ^"for " ^ string_of_typ t1 ^ " and " ^ string_of_typ
            in (ty, SBinop((t1, e1'), op, (t2, e2')))
      | PrintCall(e) -> (Void,SPrintCall(check_expr e))
      | ActionCall(aname, args) as acall ->
        let ad = find_act aname in
        let param_length = List.length ad.aparams in
        if List.length args != param_length then
          raise (Failure ("expecting " ^ string_of_int param_length ^
                          " arguments in " ^ string_of_expr acall))
        else let check_call (at, _) e =
          let (et, e') = check_expr e in
          let err = "illegal argument found " ^ string_of_typ et ^
            " expected " ^ string_of_typ at ^ " in " ^ string_of_expr e
          in (check_assign at et err, e')
        in
        let args' = List.map2 check_call ad.aparams args
        in (ad.atyp, SActionCall(aname, args'))
      | Seriesliteral vals ->
          let (t', _) = check_expr (List.hd vals) in
          let map_func lit = check_expr lit in
          let vals' = List.map map_func vals in
          (Series t', SSeriesliteral(t', vals'))
      | SeriesGet (var, e) ->
          let (t, e') = check_expr e in
          let ty = match t with
              Int -> Int
              | _ -> raise (Failure ("list_get index must be integer, not " ^ string_of_typ t))
          in let list_type = check_series_type var
          in (list_type, SSeriesGet(list_type, var, (ty, e')))
      | SeriesSize var ->
        (Int, SSeriesSize(check_series_type var, var))
      | SeriesPop var ->
        let series_type = check_series_type var
        in (series_type, SSeriesPop(series_type, var))
  in

  let check_match_series_type_expr l e =
    let (t', e') as e'' = check_expr e
     in let err = "series type and expression type do not match " ^ (string_of_typ t') ^ ", " ^ (string_of_sexpr e'')
     in if t' != (check_series_type l) then raise (Failure err) else (t', e')
   in

   let check_bool_expr e =
    let (t', e') = check_expr e
      and err = "expected Boolean expression in " ^ string_of_expr e
      in if t' != Bool then raise (Failure err) else (t', e')
  in

  let rec check_stmt = function (*currently only supports one input -- support for map *)
    Expr e -> SExpr (check_expr e)
    | SeriesPush (var, e) ->
      let _ = check_series_type var in
      SSeriesPush(var, check_match_series_type_expr var e)
    | If(p, b1, b2) -> SIf(check_bool_expr p, check_stmt b1, check_stmt b2)
    | While(p, s) -> SWhile(check_bool_expr p, check_stmt s)
    | For(e1, e2, e3, st) ->
      SFor(check_expr e1, check_bool_expr e2, check_expr e3, check_stmt st)
    | Return e -> let (t, e') = check_expr e in
        if t = act.atyp then SReturn (t, e')
        else raise (
                Failure ("return gives " ^ string_of_typ t ^ " expected " ^
                    string_of_typ act.atyp ^ " in " ^ string_of_expr e))
    | Block sl ->
      let rec check_stmt_list = function
```

```
        [Return _ as s] -> [check_stmt s]
        | Return _ :: _   -> raise (Failure "nothing may follow a return")
        | Block sl :: ss  -> check_stmt_list (sl @ ss) (* Flatten blocks *)
        | s :: ss         -> check_stmt s :: check_stmt_list ss
        | []              -> []
      in SBlock(check_stmt_list sl) (*(List.map check_stmt_list sl)*)
    | Nostmt -> SNostmt
    in
    {
      saname = act.aname;
      satyp = act.atyp;
      saparams = act.aparams;
      salocals  = act.alocals;
      sabody = match check_stmt (Block act.abody) with
      SBlock(sl) -> sl
          | _ -> raise (Failure ("internal error: block didn't become a block?"))
    }

  in (globals, List.map check_action action_decls, check_main main_decl)
```

codegen.ml

```
module L = Llvm
module A = Ast
open Sast

module StringMap = Map.Make(String)

let translate (globals, action_decls, main_decl) =
        let context    = L.global_context () in

        let the_module = L.create_module context "AHOD" in

        let str_format_str = L.define_global "str" (L.const_stringz context "%s\n") the_module
        and int_format_str =  L.define_global "str" (L.const_stringz context "%d\n") the_module
        and float_format_str = L.define_global "str" (L.const_stringz context "%g\n") the_module
        and bool_format_str = L.define_global "str" (L.const_stringz context "%d\n") the_module in

        let i32_t       = L.i32_type    context
        and i8_t        = L.i8_type     context
        and i1_t        = L.i1_type     context
        and float_t     = L.double_type context
        and string_t    = L.pointer_type (L.i8_type context)
   and void_t      = L.void_type   context
   and series_t t  = L.struct_type context [| L.pointer_type (L.i32_type context); (L.pointer_type t) |]
   and player_t    = L.struct_type context [| (L.pointer_type (L.i8_type context)); (L.i32_type context) |]
   and card_t      = L.struct_type context [| (L.pointer_type (L.i8_type context)); (L.i1_type context); (L.i32_type contex
   (*struct_set_body  class_t *)
  in

  let rec ltype_of_typ = function
```

```
    | A.Int   -> i32_t
    | A.Bool  -> i1_t
    | A.Float -> float_t
    | A.String -> string_t
    | A.Void -> void_t
    | A.Series t -> series_t (ltype_of_typ t)
    | A.Player -> player_t
    | A.Card -> card_t
  in

  let type_str t = match t with
    | A.Int -> "int"
    | A.Bool -> "bool"
    | A.Float -> "float"
    | A.String -> "str"
    | A.Card -> "card"
    | A.Player -> "player"
    | _ -> raise (Failure "Invalid string map key type")
  in

 let global_vars : L.llvalue StringMap.t = (* type:  L.llvalue StringMap.t *)
  let global_var m (t, n) = (*t: type, n:name*)
    let init = match t with
        A.Float -> L.const_float (ltype_of_typ t) 0.0
      | A.String -> L.const_pointer_null (ltype_of_typ t)
      | A.Series series_type -> L.const_struct context ([| L.const_pointer_null (L.pointer_type(L.i32_type context)); L.con
      (* ======================== initialized the class ========================= *)
      | A.Player -> L.const_struct context ([|L.const_pointer_null (L.pointer_type(L.i8_type context)) ; L.const_pointer_nu
      | A.Card -> L.const_struct context ([|L.const_pointer_null (L.pointer_type(L.i8_type context)) ; L.const_pointer_null
      (* ====================================================================== *)
      | _ -> L.const_int (ltype_of_typ t) 0
    in StringMap.add n (L.define_global n init the_module) m in
  List.fold_left global_var StringMap.empty globals in

    let printf_t : L.lltype =
      L.var_arg_function_type i32_t [| L.pointer_type i8_t |] in
    let printf_func : L.llvalue =
      L.declare_function "printf" printf_t the_module in

  (* ------------------------------------------------------------- *)
  let playercall_t : L.lltype =
        L.function_type player_t [| string_t ; i32_t |] in
  let playercall_func : L.llvalue =
      L.declare_function "playercall" playercall_t the_module in

  let getplayername_t : L.lltype =
        L.function_type string_t [| player_t |] in
  let getplayername_func : L.llvalue =
      L.declare_function "getplayername" getplayername_t the_module in

  let getplayerscore_t : L.lltype =
        L.function_type i32_t [| player_t |] in
  let getplayerscore_func : L.llvalue =
      L.declare_function "getplayerscore" getplayerscore_t the_module in

  (* let setplayername_t : L.lltype =
        L.function_type string_t [| player_t ; string_t |] in
  let setplayername_func : L.llvalue =
      L.declare_function "setplayername" setplayername_t the_module in

  let setplayerscore_t : L.lltype =
        L.function_type i32_t [| player_t ; i32_t |] in
  let setplayerscore_func : L.llvalue =
      L.declare_function "setplayerscore" setplayerscore_t the_module in *)

  (* ------------------------------------------------------------- *)

  let cardcall_t : L.lltype =
        L.function_type card_t [| string_t ; i1_t ; i32_t |] in
  let cardcall_func : L.llvalue =
      L.declare_function "cardcall" cardcall_t the_module in

  let getcardtype_t : L.lltype =
        L.function_type string_t [| card_t |] in
  let getcardtype_func : L.llvalue =
      L.declare_function "getcardtype" getcardtype_t the_module in

  let getcardfaceup_t : L.lltype =
        L.function_type i1_t [| card_t |] in
  let getcardfaceup_func : L.llvalue =
      L.declare_function "getcardfaceup" getcardfaceup_t the_module in
  let getcardvalue_t : L.lltype =
      L.function_type i32_t [| card_t |] in
  let getcardvalue_func : L.llvalue =
      L.declare_function "getcardvalue" getcardvalue_t the_module in

(* ------------------------------------------------------------- *)

  (*series generation*)
  let init_series builder series_ptr series_type =
    (* size to 0 *)
    let sizePtrPtr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
        let sizePtr = L.build_alloca i32_t "series_size" builder in
        let _ = L.build_store (L.const_int i32_t 0) sizePtr builder in
        ignore(L.build_store sizePtr sizePtrPtr builder);
    (* init series *)
    let series_array_ptr = L.build_struct_gep series_ptr 1 "series.arry" builder in
      (* ERROR: when there's nothing so like a = [] *)
      let p = L.build_array_alloca (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
      ignore(L.build_store p series_array_ptr builder);
  in

        (*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ define actions ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
  let action_decls_map : (L.llvalue * saction_decl) StringMap.t =
```

```
    let action_decl m adecl =
      let name = adecl.saname
        and param_types = Array.of_list (List.map (fun (t,_) -> ltype_of_typ t) adecl.saparams) in
      let atype = L.function_type (ltype_of_typ adecl.satyp) param_types in
    StringMap.add name (L.define_function name atype the_module, adecl) m in
  List.fold_left action_decl StringMap.empty action_decls in

  (*~~~~~~~~~~~~~~~~~~~~~~~~~ action generation top-level ~~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
  let build_action_body adecl =
    let (the_action, _) = StringMap.find adecl.saname action_decls_map in
    let builder = L.builder_at_end context (L.entry_block the_action) in

    let local_vars =
      let add_param m (t, n) p = L.set_value_name n p;
      let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore(
          match t with
            A.Series series_type -> init_series builder local series_type
          | _ -> ()
        );
        ignore (L.build_store p local builder);
      StringMap.add n local m

      (* Allocate space for any locally declared variables and add the
        * resulting registers to our map *)
      and add_local m (t, n) =
      let local_var = L.build_alloca (ltype_of_typ t) n builder in
        ignore(
          match t with
            A.Series series_type -> init_series builder local_var series_type
          | _ -> ()
        );
        StringMap.add n local_var m
      in

      let params = List.fold_left2 add_param StringMap.empty adecl.saparams
          (Array.to_list (L.params the_action)) in
      List.fold_left add_local params adecl.salocals
      in


      let lookup n = try StringMap.find n local_vars
      with Not_found -> StringMap.find n global_vars
      in
    (*for the series stuff, referenced past projects + clang *)
    let init_series builder series_ptr series_type =
      let size_ptr_ptr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
          let size_ptr = L.build_alloca i32_t "series_size" builder in
          ignore(L.build_store (L.const_int i32_t 0) size_ptr builder);
          ignore(L.build_store size_ptr size_ptr_ptr builder);
      let series_ptr = L.build_struct_gep series_ptr 1 "series" builder in (* init array (series) *)
        (* fails when there's []*)
```

```
      let p = L.build_array_alloca (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
      ignore(L.build_store p series_ptr builder);
  in

  let series_push : L.llvalue StringMap.t =
    let series_push_ty m typ =
      let series_push_def = L.define_function ("series_push" ^ (type_str typ)) (L.function_type void_t [| L.pointer_type
      let build = L.builder_at_end context (L.entry_block series_push_def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param series_push_def 0) series_ptr build);
      let valPtr = L.build_alloca (ltype_of_typ typ) "val_alloc" build in
      ignore(L.build_store (L.param series_push_def 1) valPtr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
      let series_load_2 = L.build_load series_ptr_2 "series_load_2" build in
      let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
      let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
      let series_size = L.build_load series_size_ptr "series_size" build in
      let next_el_ptr = L.build_gep series_load_2 [| series_size |] "series_next_el_ptr" build in
      let next_size = L.build_add series_size (L.const_int i32_t 1) "next_size" build in
      ignore(L.build_store next_size series_size_ptr build);
      ignore(L.build_store (L.build_load valPtr "val" build) next_el_ptr build);
      ignore(L.build_ret_void build);
      StringMap.add (type_str typ) series_push_def m in
      List.fold_left series_push_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in (*change or

  let series_get : L.llvalue StringMap.t =
    let series_get_ty m typ =
      let def = L.define_function ("series_get" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (
      let build = L.builder_at_end context (L.entry_block def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param def 0) series_ptr build);
      let idx_ptr = L.build_alloca i32_t "idx_alloc" build in
      ignore(L.build_store (L.param def 1) idx_ptr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
      let series_load_2 = L.build_load series_ptr_2 "array_load" build in
      let idx = L.build_load idx_ptr "idx_load" build in
      let series_el_ptr = L.build_gep series_load_2 [| idx |] "series_el_ptr" build in
      let element_val = L.build_load series_el_ptr "series_el_ptr" build in
      ignore(L.build_ret element_val build);
    StringMap.add (type_str typ) def m in
  List.fold_left series_get_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

  let series_size : L.llvalue StringMap.t =
    let series_size_ty m typ =
      let series_size_def = L.define_function ("series_size" ^ (type_str typ)) (L.function_type i32_t [| L.pointer_type (
      let build = L.builder_at_end context (L.entry_block series_size_def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param series_size_def 0) series_ptr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
```

```
        let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
        let series_size = L.build_load series_size_ptr "series_size" build in
        ignore(L.build_ret series_size build);
      StringMap.add (type_str typ) series_size_def m in
    List.fold_left series_size_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

    let series_pop : L.llvalue StringMap.t =
    let series_pop_ty m typ =
    let def = L.define_function ("series_pop" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (seri
    let build = L.builder_at_end context (L.entry_block def) in
    let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
    ignore(L.build_store (L.param def 0) series_ptr build);
    let series_load = L.build_load series_ptr "series_load" build in
    let series_arr_ptr = L.build_struct_gep series_load 1 "series_arr_ptr" build in
    let series_arr_load = L.build_load series_arr_ptr "series_arr_load" build in
    let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
    let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
    let series_size = L.build_load series_size_ptr "series_size" build in
    let series_sizeMin = L.build_sub series_size (L.const_int i32_t 1) "dec_size" build in
    let last_el_ptr = L.build_gep series_arr_load [| series_sizeMin |] "series_next_el_ptr" build in
    let last_el_val = L.build_load last_el_ptr "series_arry_next_element" build in
    let _ = L.build_store series_sizeMin series_size_ptr build in
    let _ = L.build_ret last_el_val build in
    StringMap.add (type_str typ) def m in
    List.fold_left series_pop_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

    (*expression generation*)
    let rec expr builder ((_, e) : sexpr) = match e with
      | SSliteral s   -> L.build_global_stringptr s "str" builder
      | SBliteral b -> L.const_int i1_t (if b then 1 else 0)
      | SIliteral i -> L.const_int i32_t i
      | SFliteral f -> L.const_float_of_string float_t f
      | SId s       -> L.build_load (lookup s) s builder
      | SNoexpr     -> L.const_int i32_t 0
      | SAssign (s, e) -> let e' = expr builder e in
                     ignore(L.build_store e' (lookup s) builder); e'
      (* | SAttrAssign (objname, attr, e) -> let e' = expr builder e in
      (match attr with
      "name" -> (L.build_call setplayername_func [|(L.build_load (lookup objname) objname builder) ; e'|] "setplayername" b
      | "score" -> (L.build_call setplayerscore_func [|(L.build_load (lookup objname) objname builder) ; e'|] "setplayersco
      ) *)
      | SPrintCall(e) ->
        (match fst e with
        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
        "printf" builder
        | 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
        "printf" builder
        | A.Float -> L.build_call printf_func [| L.const_in_bounds_gep float_format_str [|L.const_int i32_t 0; L.const_int
        "printf" builder
        | A.Bool -> L.build_call printf_func [| L.const_in_bounds_gep bool_format_str [|L.const_int i32_t 0; L.const_int i3
        "printf" builder
        | _ -> raise (Failure "Print of this type is not supported") (* Potentially need to support Class, Series, and None
        )
      | SPlayerClassCall(e) ->
        L.build_call playercall_func (Array.of_list (List.map (expr builder) (e))) "playercall" builder
      | SCardClassCall(e) ->
        L.build_call cardcall_func (Array.of_list (List.map (expr builder) (e))) "cardcall" builder
      | SAttrCall(objname, attr) ->
        (match attr with
        "name" -> L.build_call getplayername_func [|(L.build_load (lookup objname) objname builder)|] "getplayername" build
        | "score" -> L.build_call getplayerscore_func [|(L.build_load (lookup objname) objname builder)|] "getplayerscore"
        | "type" -> L.build_call getcardtype_func [|(L.build_load (lookup objname) objname builder)|] "getcardtype" builder
        | "faceup" -> L.build_call getcardfaceup_func [|(L.build_load (lookup objname) objname builder)|] "getcardfaceup" b
        | "value" -> L.build_call getcardvalue_func [|(L.build_load (lookup objname) objname builder)|] "getcardvalue" buil
        | _ -> raise (Failure "attribute is not supported")
        )
      | SActionCall(a, args) ->
        let (adef, main_func) = StringMap.find a action_decls_map in
        let llargs = List.rev (List.map (expr builder) (List.rev args)) in
        let result = (match main_func.satyp with
                      A.Void -> ""
                    | _ -> a ^ "_result")
        in L.build_call adef (Array.of_list llargs) result builder
        | SBinop ((A.Float,_ ) as e1, op, e2) ->
        let e1' = expr builder e1
        and e2' = expr builder e2 in
        (match op with
        A.Add     -> L.build_fadd
        | A.Sub     -> L.build_fsub
        | A.Mult    -> L.build_fmul
        | A.Div     -> L.build_fdiv
        | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
        | A.Neq     -> L.build_fcmp L.Fcmp.One
        | A.Less    -> L.build_fcmp L.Fcmp.Olt
        | A.Leq     -> L.build_fcmp L.Fcmp.Ole
        | A.Greater -> L.build_fcmp L.Fcmp.Ogt
        | A.Geq     -> L.build_fcmp L.Fcmp.Oge
        | A.And | A.Or ->
        raise (Failure "internal error: semant should have rejected and/or on float")
        ) e1' e2' "tmp" builder
        | SBinop (e1, op, e2) ->
        let e1' = expr builder e1
        and e2' = expr builder e2
        in
        (match op with
        A.Add     -> L.build_add
        | A.Sub     -> L.build_sub
        | A.Mult    -> L.build_mul
        | A.Div     -> L.build_sdiv
        | A.And     -> L.build_and
        | A.Or      -> L.build_or
        | A.Equal   -> L.build_icmp L.Icmp.Eq
        | A.Neq     -> L.build_icmp L.Icmp.Ne
        | A.Less    -> L.build_icmp L.Icmp.Slt
        | A.Leq     -> L.build_icmp L.Icmp.Sle
        | A.Greater -> L.build_icmp L.Icmp.Sgt
        | A.Geq     -> L.build_icmp L.Icmp.Sge
        ) e1' e2' "tmp" builder
      | SUnop(op, ((t, _) as e)) ->
        let e' = expr builder e in
        (match op with
          | A.Neg when t = A.Float -> L.build_fneg
          | A.Neg                  -> L.build_neg
          | A.Not                  -> L.build_not)
        e' "tmp" builder
      | SSeriesliteral (series_type, literals) ->
        let ltype = (ltype_of_typ series_type) in (*gets type of elements in arr *)
        let new_series_ptr = L.build_alloca (series_t ltype) "new_series_ptr" builder in
        let _ = init_series builder new_series_ptr series_type in
        let map_func literal =
            ignore(L.build_call (StringMap.find (type_str series_type) series_push) [| new_series_ptr; (expr builder litera
        in
        let _ = List.rev (List.map map_func literals) in
        L.build_load new_series_ptr "new_series" builder
      | SSeriesGet (series_type, id, e) ->
          L.build_call (StringMap.find (type_str series_type) series_get) [| (lookup id); (expr builder e) |] "series_get"
      | SSeriesSize (series_type, id) ->
          L.build_call ((StringMap.find (type_str series_type)) series_size) [| (lookup id) |] "series_size" builder
      | SSeriesPop (series_type, id) ->
          L.build_call ((StringMap.find (type_str series_type)) series_pop) [| (lookup id) |] "series_pop" builder
    in

    let add_terminal builder instr =
    match L.block_terminator (L.insertion_block builder) with
        Some _ -> ()
      | None -> ignore (instr builder) in

    let rec stmt builder = function
      | SBlock stmt_list -> List.fold_left stmt builder stmt_list
      | SExpr e -> ignore(expr builder e); builder
      | SReturn e -> ignore(match adecl.satyp with
                                (* Special "return nothing" instr *)
                                A.Void -> L.build_ret_void builder
                                (* Build return statement *)
                              | _ -> L.build_ret (expr builder e) builder );
                              builder
      | SSeriesPush (id, e) ->
          ignore(L.build_call (StringMap.find (type_str (fst e)) series_push) [| (lookup id); (expr builder e) |] "" builde
      | SIf (predicate, then_stmt, else_stmt) ->
        let bool_val = expr builder predicate in
        let merge_bb = L.append_block context "merge" the_action in
        let build_br_merge = L.build_br merge_bb in (* partial function *)
        let then_bb = L.append_block context "then" the_action in
        add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
        build_br_merge;
        let else_bb = L.append_block context "else" the_action in
        add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
        build_br_merge;
        ignore(L.build_cond_br bool_val then_bb else_bb builder);
        L.builder_at_end context merge_bb
        | SWhile (predicate, body) ->
        let pred_bb = L.append_block context "while" the_action in
        ignore(L.build_br pred_bb builder);

        let body_bb = L.append_block context "while_body" the_action in
        add_terminal (stmt (L.builder_at_end context body_bb) body)
        (L.build_br pred_bb);
        let pred_builder = L.builder_at_end context pred_bb in
        let bool_val = expr pred_builder predicate in

        let merge_bb = L.append_block context "merge" the_action in
        ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
        L.builder_at_end context merge_bb
        | SFor (e1, e2, e3, body) -> stmt builder
        ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )
      | SNostmt -> builder
        in

    let builder = stmt builder (SBlock adecl.sabody) in
      add_terminal builder (match adecl.satyp with
              A.Void -> L.build_ret_void
            | A.Float -> L.build_ret (L.const_float float_t 0.0)
            | t -> L.build_ret (L.const_int (ltype_of_typ t) 0));
        ()
      in
    let _ = List.iter build_action_body action_decls in

  (*~~~~~~~~~~~~~~~~~~~~  main function generation top-level ~~~~~~~~~~~~~~~~~~~~~~~~~~~*)
  let build_main main_func =
          let fty = L.function_type i32_t[||] in
                let f = L.define_function "main" fty the_module in
                let builder = L.builder_at_end context (L.entry_block f) in

    let local_vars =
      let add_param m (t, n) p = L.set_value_name n p;
      let local = L.build_alloca (ltype_of_typ t) n builder in
        ignore(
          match t with
            A.Series series_type -> init_series builder local series_type
          | _ -> ()
        );
        ignore (L.build_store p local builder);
      StringMap.add n local m

      (* Allocate space for any locally declared variables and add the
        * resulting registers to our map *)
      and add_local m (t, n) =
      let local_var = L.build_alloca (ltype_of_typ t) n builder in
```

```
        ignore(
          match t with
            A.Series series_type -> init_series builder local_var series_type
          | _ -> ()
        );
        StringMap.add n local_var m
      in

      let params = List.fold_left2 add_param StringMap.empty main_func.smparams
          (Array.to_list (L.params f)) in (*not sure about this f *)
      List.fold_left add_local params main_func.smlocals
    in

  let lookup n = try StringMap.find n local_vars
    with Not_found -> StringMap.find n global_vars
  in
  (*for the series stuff, referenced past projects + clang *)
  let init_series builder series_ptr series_type =
    let size_ptr_ptr = L.build_struct_gep series_ptr 0 "series_size_ptr" builder in
        let size_ptr = L.build_alloca i32_t "series_size" builder in
        ignore(L.build_store (L.const_int i32_t 0) size_ptr builder);
        ignore(L.build_store size_ptr size_ptr_ptr builder);
    let series_ptr = L.build_struct_gep series_ptr 1 "series" builder in (* init array (series) *)
      (* fails when a = [] *)
      let p = L.build_array_alloca (ltype_of_typ series_type) (L.const_int i32_t 1028) "p" builder in
      ignore(L.build_store p series_ptr builder);
  in

  let series_push : L.llvalue StringMap.t =
    let series_push_ty m typ =
      let series_push_def = L.define_function ("series_push" ^ (type_str typ)) (L.function_type void_t [| L.pointer_type (s
      let build = L.builder_at_end context (L.entry_block series_push_def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param series_push_def 0) series_ptr build);
      let valPtr = L.build_alloca (ltype_of_typ typ) "val_alloc" build in
      ignore(L.build_store (L.param series_push_def 1) valPtr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
      let series_load_2 = L.build_load series_ptr_2 "series_load_2" build in
      let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
      let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
      let series_size = L.build_load series_size_ptr "series_size" build in
      let next_el_ptr = L.build_gep series_load_2 [| series_size |] "series_next_el_ptr" build in
      let next_size = L.build_add series_size (L.const_int i32_t 1) "next_size" build in
      ignore(L.build_store next_size series_size_ptr build);
      ignore(L.build_store (L.build_load valPtr "val" build) next_el_ptr build);
      ignore(L.build_ret_void build);
      StringMap.add (type_str typ) series_push_def m in
      List.fold_left series_push_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in (*change orde

  let series_get : L.llvalue StringMap.t =
    let series_get_ty m typ =
      let def = L.define_function ("series_get" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (se
      let build = L.builder_at_end context (L.entry_block def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param def 0) series_ptr build);
      let idx_ptr = L.build_alloca i32_t "idx_alloc" build in
      ignore(L.build_store (L.param def 1) idx_ptr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_ptr_2 = L.build_struct_gep series_load 1 "series_ptr_2" build in
      let series_load_2 = L.build_load series_ptr_2 "array_load" build in
      let idx = L.build_load idx_ptr "idx_load" build in
      let series_el_ptr = L.build_gep series_load_2 [| idx |] "series_el_ptr" build in
      let element_val = L.build_load series_el_ptr "series_el_ptr" build in
      ignore(L.build_ret element_val build);
    StringMap.add (type_str typ) def m in
  List.fold_left series_get_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

  let series_size : L.llvalue StringMap.t =
    let series_size_ty m typ =
      let series_size_def = L.define_function ("series_size" ^ (type_str typ)) (L.function_type i32_t [| L.pointer_type (se
      let build = L.builder_at_end context (L.entry_block series_size_def) in
      let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
      ignore(L.build_store (L.param series_size_def 0) series_ptr build);
      let series_load = L.build_load series_ptr "series_load" build in
      let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
      let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
      let series_size = L.build_load series_size_ptr "series_size" build in
      ignore(L.build_ret series_size build);
    StringMap.add (type_str typ) series_size_def m in
  List.fold_left series_size_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

  let series_pop : L.llvalue StringMap.t =
  let series_pop_ty m typ =
  let def = L.define_function ("series_pop" ^ (type_str typ)) (L.function_type (ltype_of_typ typ) [| L.pointer_type (series
  let build = L.builder_at_end context (L.entry_block def) in
  let series_ptr = L.build_alloca (L.pointer_type (series_t (ltype_of_typ typ))) "series_ptr_alloc" build in
  ignore(L.build_store (L.param def 0) series_ptr build);
  let series_load = L.build_load series_ptr "series_load" build in
  let series_arr_ptr = L.build_struct_gep series_load 1 "series_arr_ptr" build in
  let series_arr_load = L.build_load series_arr_ptr "series_arr_load" build in
  let series_size_ptr_ptr = L.build_struct_gep series_load 0 "series_size_ptr_ptr" build in
  let series_size_ptr = L.build_load series_size_ptr_ptr "series_size_ptr" build in
  let series_size = L.build_load series_size_ptr "series_size" build in
  let series_sizeMin = L.build_sub series_size (L.const_int i32_t 1) "dec_size" build in
  let last_el_ptr = L.build_gep series_arr_load [| series_sizeMin |] "series_next_el_ptr" build in
  let last_el_val = L.build_load last_el_ptr "series_arry_next_element" build in
  let _ = L.build_store series_sizeMin series_size_ptr build in
  let _ = L.build_ret last_el_val build in
  StringMap.add (type_str typ) def m in
  List.fold_left series_pop_ty StringMap.empty [ A.Bool; A.Int; A.Float; A.String; A.Card; A.Player ] in

  (*expression generation*)
  let rec expr builder ((_, e) : sexpr) = match e with
```

```
      | SSliteral s   -> L.build_global_stringptr s "str" builder
      | SBliteral b -> L.const_int i1_t (if b then 1 else 0)
      | SIliteral i -> L.const_int i32_t i
      | SFliteral f -> L.const_float_of_string float_t f
      | SId s       -> L.build_load (lookup s) s builder
      | SNoexpr     -> L.const_int i32_t 0
      | SAssign (s, e) -> let e' = expr builder e in
                      ignore(L.build_store e' (lookup s) builder); e'
      | SPrintCall(e) ->
        (match fst e with
        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
        "printf" builder
        | 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_i
        "printf" builder
        | A.Float -> L.build_call printf_func [| L.const_in_bounds_gep float_format_str [|L.const_int i32_t 0; L.const_int i3
        "printf" builder
        | 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_
        "printf" builder
        | _ -> raise (Failure "Print of this type is not supported") (* Potentially need to support Class, Series, and None c
        )
        (* | SAttrAssign (objname, attr, e) -> let e' = expr builder e in
        (match attr with
        "name" -> (L.build_call setplayername_func [|(L.build_load (lookup objname) objname builder) ; e'|] "setplayername" b
        | "score" -> (L.build_call setplayerscore_func [|(L.build_load (lookup objname) objname builder) ; e'|] "setplayersco
        ) *)
        | SPlayerClassCall(e) ->
        L.build_call playercall_func (Array.of_list (List.map (expr builder) (e))) "playercall" builder
        | SCardClassCall(e) ->
        L.build_call cardcall_func (Array.of_list (List.map (expr builder) (e))) "cardcall" builder
        | SAttrCall(objname, attr) ->
          (match attr with
          "name" -> L.build_call getplayername_func [|(L.build_load (lookup objname) objname builder)|] "getplayername" build
          | "score" -> L.build_call getplayerscore_func [|(L.build_load (lookup objname) objname builder)|] "getplayerscore"
          | "type" -> L.build_call getcardtype_func [|(L.build_load (lookup objname) objname builder)|] "getcardtype" builder
          | "faceup" -> L.build_call getcardfaceup_func [|(L.build_load (lookup objname) objname builder)|] "getcardfaceup" b
          | "value" -> L.build_call getcardvalue_func [|(L.build_load (lookup objname) objname builder)|] "getcardvalue" buil
          | _ -> raise (Failure "attribute is not supported")
          )
        | SActionCall(a, args) ->
        let (adef, main_func) = StringMap.find a action_decls_map in
        let llargs = List.rev (List.map (expr builder) (List.rev args)) in
        let result = (match main_func.satyp with
                      A.Void -> ""
                    | _ -> a ^ "_result")
        in L.build_call adef (Array.of_list llargs) result builder
        | SBinop ((A.Float,_ ) as e1, op, e2) ->
          let e1' = expr builder e1
          and e2' = expr builder e2 in
          (match op with
          A.Add     -> L.build_fadd
          | A.Sub     -> L.build_fsub
          | A.Mult    -> L.build_fmul
          | A.Div     -> L.build_fdiv
          | A.Equal   -> L.build_fcmp L.Fcmp.Oeq
          | A.Neq     -> L.build_fcmp L.Fcmp.One
          | A.Less    -> L.build_fcmp L.Fcmp.Olt
          | A.Leq     -> L.build_fcmp L.Fcmp.Ole
          | A.Greater -> L.build_fcmp L.Fcmp.Ogt
          | A.Geq     -> L.build_fcmp L.Fcmp.Oge
          | A.And | A.Or ->
          raise (Failure "internal error: semant should have rejected and/or on float")
          ) e1' e2' "tmp" builder
        | SBinop (e1, op, e2) ->
          let e1' = expr builder e1
          and e2' = expr builder e2
          in
          (match op with
          A.Add     -> L.build_add
          | A.Sub     -> L.build_sub
          | A.Mult    -> L.build_mul
          | A.Div     -> L.build_sdiv
          | A.And     -> L.build_and
          | A.Or      -> L.build_or
          | A.Equal   -> L.build_icmp L.Icmp.Eq
          | A.Neq     -> L.build_icmp L.Icmp.Ne
          | A.Less    -> L.build_icmp L.Icmp.Slt
          | A.Leq     -> L.build_icmp L.Icmp.Sle
          | A.Greater -> L.build_icmp L.Icmp.Sgt
          | A.Geq     -> L.build_icmp L.Icmp.Sge
          ) e1' e2' "tmp" builder
        | SUnop(op, ((t, _) as e)) ->
          let e' = expr builder e in
          (match op with
            | A.Neg when t = A.Float -> L.build_fneg
            | A.Neg                  -> L.build_neg
            | A.Not                  -> L.build_not)
          e' "tmp" builder
        | SSeriesliteral (series_type, literals) ->
          let ltype = (ltype_of_typ series_type) in (*gets type of elements in arr *)
          let new_series_ptr = L.build_alloca (series_t ltype) "new_series_ptr" builder in
          let _ = init_series builder new_series_ptr series_type in
          let map_func literal =
              ignore(L.build_call (StringMap.find (type_str series_type) series_push) [| new_series_ptr; (expr builder litera
          in
          let _ = List.rev (List.map map_func literals) in
          L.build_load new_series_ptr "new_series" builder
        | SSeriesGet (series_type, id, e) ->
            L.build_call (StringMap.find (type_str series_type) series_get) [| (lookup id); (expr builder e) |] "series_get"
        | SSeriesSize (series_type, id) ->
            L.build_call ((StringMap.find (type_str series_type)) series_size) [| (lookup id) |] "series_size" builder
        | SSeriesPop (series_type, id) ->
            L.build_call ((StringMap.find (type_str series_type)) series_pop) [| (lookup id) |] "series_pop" builder
    in

```

```
let add_terminal builder instr =
  match L.block_terminator (L.insertion_block builder) with
    Some _ -> ()
    | None -> ignore (instr builder) in

let rec stmt builder = function
  | SBlock stmt_list -> List.fold_left stmt builder stmt_list
  | SExpr e -> ignore(expr builder e); builder
  | SReturn e -> ignore(match main_func.smtyp with (*Should be unused*)
                           (* Special "return nothing" instr *)
                           A.Void -> L.build_ret_void builder
                           (* Build return statement *)
                         | _ -> L.build_ret (expr builder e) builder );
                         builder
  | SSeriesPush (id, e) ->
      ignore(L.build_call (StringMap.find (type_str (fst e)) series_push) [| (lookup id); (expr builder e) |] "" builder)
  | SIf (predicate, then_stmt, else_stmt) ->
    let bool_val = expr builder predicate in
    let merge_bb = L.append_block context "merge" f in
    let build_br_merge = L.build_br merge_bb in (* partial function *)

    let then_bb = L.append_block context "then" f in
    add_terminal (stmt (L.builder_at_end context then_bb) then_stmt)
    build_br_merge;

    let else_bb = L.append_block context "else" f in
    add_terminal (stmt (L.builder_at_end context else_bb) else_stmt)
    build_br_merge;
    ignore(L.build_cond_br bool_val then_bb else_bb builder);
    L.builder_at_end context merge_bb
    | SWhile (predicate, body) ->
    let pred_bb = L.append_block context "while" f in
    ignore(L.build_br pred_bb builder);

    let body_bb = L.append_block context "while_body" f in
    add_terminal (stmt (L.builder_at_end context body_bb) body)
    (L.build_br pred_bb);

    let pred_builder = L.builder_at_end context pred_bb in
    let bool_val = expr pred_builder predicate in

    let merge_bb = L.append_block context "merge" f in
    ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
    L.builder_at_end context merge_bb
    | SFor (e1, e2, e3, body) -> stmt builder
    ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e3]) ] )
  | SNostmt -> builder
    (*| SForLit ( e1, e2, body) -> stmt builder
    ( SBlock [SExpr e1 ; SWhile (e2, SBlock [body ; SExpr e]) ] ) *)
    in

let builder = stmt builder (SBlock main_func.smbody) in
```

```
  let _ = L.build_ret (L.const_int i32_t 0) (builder) in
  () in

let _ = build_main main_decl in
the_module;
```

playercall.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef struct {
  int *array;
  size_t used;
  size_t size;
} Array;

void initArray(Array *a, size_t initialSize) {
  a->array = malloc(initialSize * sizeof(int));
  a->used = 0;
  a->size = initialSize;
}

void insertArray(Array *a, int element) {
  // a->used is the number of used entries, because a->array[a->used++] updates a->used only *after* the array has been acc
  // Therefore a->used can go up to a->size
  if (a->used == a->size) {
    a->size *= 2;
    a->array = realloc(a->array, a->size * sizeof(int));
  }
  a->array[a->used++] = element;
}

void freeArray(Array *a) {
  free(a->array);
  a->array = NULL;
  a->used = a->size = 0;
}

struct Player
{
  char *name;
  int score;
};

/*int main() {

  struct Player *p;
  p = player("Bob",1);

  printf("Pointer is: (%s, %d)", p->name, p->score);

  return 0;
}*/

struct Player *playercall(char* name, int score)
{
  struct Player *p;
  p = (struct Player *) malloc(sizeof(struct Player));
  p->name = name;
  p->score = score;
  return p;
}

char *getplayername(struct Player *player)
{
  return player->name;
}

int getplayerscore(struct Player *player)
{
  return player->score;
}

void *setplayername(struct Player *player, char* name)
{
  player->name = name;
  return (void *)player->name;
}

int setplayerscore(struct Player *player, int score)
{
  player->score = score;
  return player->score;
}

struct Card
{
  char *type;
  bool faceup;
  int value;
};

struct Card *cardcall(char* type, bool faceup, int value)
{
  struct Card *c;
  c = (struct Card *) malloc(sizeof(struct Card));
  c->type = type;
  c->faceup = faceup;
  c->value = value;
  return c;
}
```

```c

char *getcardtype(struct Card *card)
{
  return card->type;
}

bool getcardfaceup(struct Card *card)
{
  return card->faceup;
}

int getcardvalue(struct Card *card)
{
  return card->value;
}
```

AHOD.ml

```ocaml
(* Top-level of the MicroC compiler: scan & parse the input,
   check the resulting AST and generate an SAST from it, generate LLVM IR,
   and dump the module *)

   type action = Ast | Sast | LLVM_IR | Compile
   (* | Token  *)

   let () =
     let action = ref Compile in
     let set_action a () = action := a in
     let speclist = [
       ("-a", Arg.Unit (set_action Ast), "Print the AST");
       ("-s", Arg.Unit (set_action Sast), "Print the SAST");
       (* ("-t", Arg.Unit (set_action Token ), "Print Tokens"); *)
       ("-l", Arg.Unit (set_action LLVM_IR), "Print the generated LLVM IR");
       ("-c", Arg.Unit (set_action Compile),
         "Check and print the generated LLVM IR (default)");
     ] in
     let usage_msg = "usage: ./AHOD.native [-a|-s|-t|-l|-c] [test-helloworld.ah]" in
     let channel = ref stdin in
     Arg.parse speclist (fun filename -> channel := open_in filename) usage_msg;

     let lexbuf = Lexing.from_channel !channel in
     (* match action with
     | Token -> print_string
    | _ ->  *)
     let ast = Parse.program Scanner.token lexbuf
    in
      match !action with
       Ast -> ()
     | _ -> let sast = Semant.check ast in
       match !action with
         Ast     -> ()
       | Sast    -> ()
       | LLVM_IR -> ()
       | Compile -> let m = Codegen.translate sast
      in
     Llvm_analysis.assert_valid_module m;
     print_string (Llvm.string_of_llmodule m)
```

testall.sh

```
#!/bin/sh

# Regression testing script for MicroC
# Step through a list of files
#  Compile, run, and check the output of each expected-to-work test
#  Compile and check the error of each expected-to-fail test

# Path to the LLVM interpreter
LLI="lli"
#LLI="/usr/local/opt/llvm/bin/lli"

# Path to the LLVM compiler
LLC="llc"

# Path to the C compiler
CC="cc"

# Path to the microc compiler.  Usually "./microc.native"
# Try "_build/microc.native" if ocamlbuild was unable to create a symbolic link.
AHOD="./AHOD.native"
#MICROC="_build/microc.native"

# Set time limit for all operations
ulimit -t 30

globallog=testall.log
rm -f $globallog
error=0
globalerror=0

keep=0

Usage() {
    echo "Usage: testall.sh [options] [.mc files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}

SignalError() {
    if [ $error -eq 0 ] ; then
        echo "FAILED"
        error=1
    fi
    echo "  $1"
}

# Compare <outfile> <reffile> <difffile>
# Compares the outfile with reffile.  Differences, if any, written to difffile
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
        SignalError "$1 differs"
        echo "FAILED $1 differs from $2" 1>&2
    }
}

# Run <args>
# Report the command, run it, and report any errors
Run() {
    echo $* 1>&2
    eval $* || {
        SignalError "$1 failed on $*"
        return 1
    }
}

# RunFail <args>
# Report the command, run it, and expect an error
RunFail() {
    echo $* 1>&2
    eval $* && {
        SignalError "failed: $* did not report an error"
        return 1
    }
    return 0
}

Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.ah//'`
    reffile=`echo $1 | sed 's/.ah$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.ll ${basename}.s ${basename}.exe ${basename}.out" &&
    Run "$AHOD" "$1" ">" "${basename}.ll" &&
    Run "$LLC" "-relocation-model=pic" "${basename}.ll" ">" "${basename}.s" &&
    Run "$CC" "-o" "${basename}.exe" "${basename}.s" "playercall.o" &&
    Run "./${basename}.exe" > "${basename}.out" &&
    Compare ${basename}.out ${reffile}.out ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
        if [ $keep -eq 0 ] ; then
            rm -f $generatedfiles
        fi
        echo "OK"
        echo "###### SUCCESS" 1>&2
    else
        echo "###### FAILED" 1>&2
        globalerror=$error
    fi
}

CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.ah//'`
    reffile=`echo $1 | sed 's/.ah$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."

    echo -n "$basename..."

    echo 1>&2
    echo "###### Testing $basename" 1>&2

    generatedfiles=""

    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    RunFail "$AHOD" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff

    # Report the status and clean up the generated files

    if [ $error -eq 0 ] ; then
        if [ $keep -eq 0 ] ; then
            rm -f $generatedfiles
        fi
        echo "OK"
        echo "###### SUCCESS" 1>&2
    else
        echo "###### FAILED" 1>&2
        globalerror=$error
    fi
}

```

```
while getopts kdpsh c; do
    case $c in
        k) # Keep intermediate files
            keep=1
            ;;
        h) # Help
            Usage
            ;;
    esac
done

shift `expr $OPTIND - 1`

LLIFail() {
  echo "Could not find the LLVM interpreter \"$LLI\"."
  echo "Check your LLVM installation and/or modify the LLI variable in testall.sh"
  exit 1
}

which "$LLI" >> $globallog || LLIFail

if [ ! -f playercall.o ]
then
    echo "Could not find playercall.o"
    echo "Try \"make playercall.o\""
    exit 1
fi

if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/test-*.ah tests/fail-*.ah"
fi

for file in $files
do
    case $file in
        *test-*)
            Check $file 2>> $globallog
            ;;
        *fail-*)
            CheckFail $file 2>> $globallog
            ;;
        *)
            echo "unknown file type $file"
            globalerror=1
            ;;
    esac
done

exit $globalerror
```

test-action.ah

```
int x
when do void PLAY():
{
  do PRINT("actions are cool")
}
main:
{
  do PLAY()
}
```

test-action2.ah

```
int x
when do void PLAY():
{
  do PRINT("actions are cool")
}
when do void PLAYTWO():
{
  do PRINT("two actions are cooler")
}
main:
{
  do PLAY()
  do PLAYTWO()
}
```

test-action3-while.ah

```
when do void PLAY():
{
    while 3>5:
    {
    do PRINT(0)
    }
    do PRINT(1)
}
main:
{
    do PLAY()
}
```

test-action4-for.ah

```
when do void PLAY():
{
    int i
    for (i = 0 ; i < 5; i = i + 1):
    {
    do PRINT(i)
    }
}

main:
{
    do PLAY()
}
```

test-aciton5-if.ah

```
when do void PLAY():
{
    if 3<5 :
    {
    do PRINT(1)
    }
}
main:
{
    do PLAY()
}
```

test-action6-ifelse.ah

```
when do void PLAY():
{
    if 3>5 :
    {
        do PRINT(1)
    }
    else:
    {
        do PRINT(0)
    }
}
main:
{
    do PLAY()
}
```

test-action7-params.ah

```
when do string A(string x,int y):
{
    string z
    z = x
    do PRINT(y)
    return z
}
main:
{
    string w
    w = do A("Ahoy Matey, to AHOD", 3)
    do PRINT(w)
}
```

test-advanced.ah

```
when do series<Card> CREATEPILE():
{
    Card card1
    card1 = Card("R5", true, 5)
    return [card1]
}
main:
{
    series<Card> cards
    Card card
    cards = do CREATEPILE()
    card = cards[0]
    do PRINT(card.type)
    do PRINT(card.faceup)
}
```

test-assign-global.ah

```
int x
bool y
string z
main:
  {
  x = 4
  do PRINT(x)
  y = true
  do PRINT(y)
  z = "banana"
  do PRINT(z)
  }
```

test-assign-local.ah

```
main:
{
  int x
  x = 3
  do PRINT(x)
}
```

test-assign-local2.ah

```
int z
when do void PLAY():
{
  int x
  int y
  x = 3
  y = 4
  do PRINT(x)
  do PRINT(y)
}
main:
{
  int a
  a = 3
  z = 5
  do PRINT(z)
  do PRINT(a)
  do PLAY()
}
```

test-basic.ah

```
int a
main:
{
    float b
    b = 5.0
    if 3<5:
    {
        for (a = 0 ; a < 5; a = a + 1):
        {
            b = b + 1.0
            do PRINT(b)
        }
    }
}
```

test-binop.ah

```
main:
  {
  do PRINT(1 + 2)
  do PRINT(1 - 2)
  do PRINT(1 * 2)
  do PRINT(1.0 / 2.0)
  do PRINT(true)
  do PRINT(true and true)
  do PRINT(true and false)
  do PRINT(true and true)
  do PRINT(false and false)
  do PRINT(true or true)
  do PRINT(true or false)
  do PRINT(true or true)
  do PRINT(false or false)
  do PRINT(1 == 2)
  do PRINT(1 == 1)
  do PRINT(1 != 2)
  do PRINT(1 != 1)
  do PRINT(1 < 2)
  do PRINT(2 < 1)
  do PRINT(1 <= 2)
  do PRINT(3 <= 2)
  do PRINT(1 > 2)
  do PRINT(2 > 1)
  do PRINT(1 >= 2)
  do PRINT(3 >= 2)
  do PRINT(-3 + 1)
  do PRINT(-3 - 1)
  do PRINT(not true)
  }
```

test-bjsim.ah

```
int i
series<Card> deck
int score
Player player
when do void INIT():
{
    player = Player("Stephen", 0)
    do PRINT("Hello, and welcome to our Blackjack Simulator")
    do PRINT("First things first let us welcome our player for the evening")
    do PRINT(player.name)
    deck = [Card("R6", true,6)]
    deck = do CREATEDECK(deck)
}
when do string COMMENT(int x):
{
    string localquip
    if x > 6 :
    {
        localquip = "That is a high value card"
    }
    else:
    {
        localquip = "That is a low value card"
    }
    return localquip
}
when do series<Card> CREATEDECK(series<Card> deck):
{
    series<string> types
    series<int> values
    int value
    string type
    Card card
    types = ["R5", "R9", "B9", "B2", "R3"]
    values = [3, 9, 9, 2, 3]
    for (i = 0; i < types.size(); i = i + 1):
    {
        type = types[i]
        value = values[i]
        card = Card(type, true,value)
        deck.push(card)
    }
    return deck
}
main:
{
    Card card
    int currvalue
    string quip
    do INIT()
    do PRINT("There are 5 cards in the hand. A Hit Request")
    card = deck[deck.size()-1]
    score = 0
    do PRINT("Card:")
    do PRINT(card.type)
    do PRINT("Current Score of:")
    score = score + card.value
    do PRINT(score)
    deck.pop()
    do PRINT("Another hit request")
    card = deck[deck.size()-1]
    do PRINT("Card:")
    do PRINT(card.type)
    do PRINT("Current Score of:")
    score = score + card.value
    do PRINT(score)
    deck.pop()
    do PRINT("Looks like we stop here for tonight. Time to see what the next cards were")
    for (i = 0 ; i < deck.size(); i = i + 1):
    {
        card = deck[i]
        do PRINT("Card:")
        do PRINT(card.type)
        score = score + card.value
        quip = do COMMENT(score)
        do PRINT(quip)
        if score<21 :
        {
        do PRINT("Could have had a higher score of: ")
        do PRINT(score)
        }
        else:
        {
        do PRINT("Good that you stopped otherwise you would have a score of: ")
        do PRINT(score)
        }
    }
}
```

test-class1.ah

```
Player player
main:
{
  player = Player("bob", 0)
  do PRINT(player.name)
  do PRINT(player.score)
}
```

test-class2.ah

```
Card card
main:
{
  card = Card("R5", true, 5)
  do PRINT(card.type)
  do PRINT(card.faceup)
  do PRINT(card.value)
}
```

test-class3.ah

```
Card card
main:
{
  card = Card("R5", true, 5)
  do PRINT(card.type)
  do PRINT(card.faceup)
  do PRINT(card.value)
  card = Card("R5", false, 5)
  do PRINT(card.faceup)
}
```

test-class4.ah

```
Player player
main:
{
  string name
  int score
  name = "bob"
  score = 0
  player = Player(name, score)
  do PRINT(player.name)
  do PRINT(player.score)
  /*player.score = 1*/
  /*do PRINT(player.score)*/
}
```

test-class5.ah

```
Card card1
Card card2
Card card3
series<Card> deck
Card type1
int i
main:
{
  string red5
  red5 = "R5"
  card1 = Card(red5, true, 5)
  card2 = Card("R6", false, 6)
  card3 = Card("R7", true, 7)
  deck = [card1, card2, card3]
  for  (i = 0; i < deck.size(); i = i + 1):
  {
          type1 = deck[i]
    do PRINT(type1.type)
    do PRINT(type1.faceup)
    do PRINT(type1.value)
  }
}
```

test-class6.ah

```
main:
{
  Card card
  card = Card("R5", true, 5)
  do PRINT(card.type)
  do PRINT(card.faceup)
  do PRINT(card.value)
}
```

test-comments.ah

```
int a /* nothing should be printed out */
when do void PLAY(): /*test*/
{/*test*/
  do PRINT("actions are cool")/*test*/
}/*test*/
main: /*comments everywhere */
  { /* woo */
    /* only single line comments, decieving huh*/
    do PLAY()
    a = 1 /* comment and code woah */
    /*test*/
  } /* woo */
```

test-demo1.ah

```
int i
series<Card> deck
when do series<Card> CREATEDECK(series<Card> deck):
{
    series<string> types
    string type
    Card card
    types = ["R0", "R1", "R2", "R3", "R4"]
    for (i = 0; i < types.size(); i = i + 1):
    {
        type = types[i]
        card = Card(type, true, 5)
        deck.push(card)
    }
    return deck
}
main:
{
    Card card
    /*card = Card("R0", true)*/
    /*deck = [card]*/
    deck = [Card("D0", true, 5)]
    deck = do CREATEDECK(deck)
    for (i = 0 ; i < 5; i = i + 1):
    {
        card = deck[i]
        do PRINT(card.type)
        do PRINT(card.faceup)
    }
}
```

test-for1.ah

```
int i
main:
{
    for (i = 0 ; i < 10; i = i + 1):
    {
    do PRINT(i)
    }
}
```

test-for2.ah

```
int i
series<int> a
main:
{
    a = [8,2,5]
    for  (i = 0; i < a.size(); i = i + 1):
    {
    do PRINT(a[i])
    }
    a.pop()
    for  (i = 0; i < a.size(); i = i + 1):
    {
    do PRINT(a[i])
    }
}
```

test-helloworld.ah

```
  main:
  {
          do PRINT("Hello world!")
          do PRINT(":)")
  }
```

test-if-else.ah

```
main:
{
    if 3>5 :
    {
        do PRINT("true")
    }
    else:
    {
        do PRINT("false")
    }
}
```

test-if.ah

```
main:
{
    if 3<5 :
    {
    do PRINT("true")
    }
}
```

test-return.ah

```
int w
string x
bool y
float z
when do int A():
{
  return 3
}
when do string B():
{
  return "some plt fun"
}
when do bool C():
{
  return true
}
when do float D():
{
  return 3.5
}
main:
{
  w = do A()
  do PRINT(w)
  x = do B()
  do PRINT(x)
  y = do C()
  do PRINT(y)
  z = do D()
  do PRINT(z)
}
```

test-series.ah

```
series<int> a
series<string> b
series<float> c
series<bool> d
main:
  {
    a = [1,3]
          do PRINT(a[1])
    b = ["banana", "apple"]
    do PRINT(b[1])
    c = [1.1]
    do PRINT(c[0])
    d = [true, false]
    do PRINT(d[1])
  }
```

test-series2.ah

```
series<int> a
series<string> b
main:
  {
    a = [2]
    a.push(1)
          do PRINT(a[1])
    a.push(3)
    do PRINT(a[2])
    do PRINT(a[2-1])
    b = ["stmt1", "stmt2"]
    b.push("stmt3")
    b.push("stmt4")
    do PRINT(b[3])
  }
```

test-series3.ah

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

test-series4.ah

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

test-series5-fr.ah

```
series<string> deck
series<string> hand
int i
main:
{
    /* This is a comment */
    deck = ["d","e","f","g"]
    hand = ["a","b","c"] /* this is another comment */
    hand.push(deck.pop()) /* deck.pop() gives 'e' */
    hand.push(deck.pop()) /* deck.pop() gives 'f' */

    /* hand = ["a", "b", "c", "f", "g"] */
    /* deck = ["d","e"] */

    do PRINT("hand is now:")
    for  (i = 0; i < hand.size(); i = i + 1):
    {
    do PRINT(hand[i])
    }
    do PRINT("deck is now:")
    for  (i = 0; i < deck.size(); i = i + 1):
    {
    do PRINT(deck[i])
    }
}
```

test-while-gcd.ah

```
int x
int y
main:
{
    x = 18
    y = 12
    while x!=y:
    {
    if x>y:
    {
        x = x - y
    }
    else:
    {
        y = y - x
    }
    }
    do PRINT(x)
}
```

test-while1.ah

```
main:
{
    while 3>5:
    {
    do PRINT("true")
    }
    do PRINT("false")
}
```

test-while2.ah

```
int x
main:
{
    x = 1
    while x<5:
    {
    x = x + 1
    }
    do PRINT(x)
}
```