T. B. A. G.

a (t)ext (b)ased (a)dventure (g)ame language

Gregory Chen | glc2121
Yu Chun (Julie) Chien | yc2937
Brian Slakter | bjs2135
Maria van Keulen | mv2482
Iris Zhang | iz2140

Professor Edwards
Programming Languages and Translators
COMS4115 Fall 2015
1 Introduction
   1.1 Motivation and Background
   1.2 Language Description
   1.3 Style
2 Tutorial
   2.1 Simple Data Types and Syntax
   2.2 Simple Program Structure Example
   2.3 Complex Data Types and Program Structure Example
   2.4 Dealing with User Input
   2.5 Compiling and Running a TBAG program
   2.6 Important Reminder
3 Language Reference Manual
   3.1 Lexical Elements
      3.1.1 Identifiers
      3.1.2 Keywords
      3.1.3 Literals
         3.1.3.1 String Literals
         3.1.3.2 Integer Literals
         3.1.3.3 Boolean Literals
      3.1.4 Operators
      3.1.5 Delimiters
         3.1.5.1 Parentheses and Braces
         3.1.5.2 Commas
         3.1.5.3 Brackets
         3.1.5.4 Semicolon
         3.1.5.5 Curly Braces
         3.1.6 Periods
      3.1.6 Whitespace
   3.2 Data Types
      3.2.1 Primitive Data Types
         3.2.1.1 int
         3.2.1.2 string
         3.2.1.3 boolean
         3.2.1.4 void
      3.2.2 Non-Primitive Data Types
         3.2.2.1 Arrays
            3.2.2.1.1 Declaring Arrays
            3.2.2.1.2 Accessing and setting array elements
            3.2.2.1.3 Array length
         3.2.2.2 Rooms
            3.2.2.2.1 Defining rooms
            3.2.2.2.2 Initialization of data for rooms
            3.2.2.2.3 Setting adjacencies
3.2.2.4 Moving between Rooms
3.2.2.5 Identifying the currentRoom
3.2.2.6 Start location
3.2.2.3 NPCs and Items
   3.2.2.3.1 Defining NPCs and Items
   3.2.2.3.2 Initialization of data for NPCs and Items
3.3 Expressions and Operators
   3.3.1 Expressions
   3.3.2 Operators
3.4 Statements
   3.4.1 The if Statement
   3.4.2 The while Statement
3.5 Functions
   3.5.1 Function Definitions
   3.5.2 Calling Functions
3.6 Program Structure and Scope
   3.6.1 Program Structure
   3.6.2 Event Driven System
   3.6.3 Ending the Game
   3.6.4 Importing Libraries
   3.6.5 Scope
3.7 Built-in Functions
   3.7.1 The print function
   3.7.2 The getInputFromOptions function
   3.7.3 The getInputAdjacentRooms function
   3.7.4 The arrLen function
3.8 Context Free Grammar
4 Project Plan
   4.1 Process used for planning, specification, development and testing
   4.2 Style Guide
      4.2.1 OCaml
      4.2.2 Version Control (Git)
      4.2.3 Bash
      4.2.4 Java
   4.3 Project timeline
   4.4 Roles and Responsibilities
   4.5 Software development environment used
   4.6 Project log
      4.6.1 Events Log
      4.6.2 Git Commit History
      4.6.3 Git Merge with Master History
      4.6.4. Active Branches (as of December 21)
5 Architectural Design
5.1 Overview
5.2 Preprocessor
5.3 Scanner
5.4 Parser
5.5 Semantic Checker
5.6 Java Builder
5.7 Code Generator

6 Testing
6.1 Representative Language Programs
   6.1.1 GCD
      6.1.1.1 GCD Source
      6.1.1.2 GCD Target
   6.1.2 Mini-game
      6.1.2.1 Mini-game Source
      6.1.2.2 Mini-game Target

6.2 Testing Suite and Justification
   6.2.1 Success Tests
   6.2.2 Fail Tests
   6.2.3 Input Tests

6.3 Testing Automation and Scripts

7 Lessons Learned
7.1 Brian Slakter
7.2 Gregory Chen
7.3 Julie Chien
7.4 Iris Zhang
7.5 Maria van Keulen

8 Acknowledgements

9 Appendix
9.1 Source Code
   9.1.1 Preprocessor (importLibrary.py)
   9.1.2 Scanner (scanner.ml)
   9.1.3 Parser (parser.mly)
   9.1.4 AST (ast.ml)
   9.1.5 Semantic Checker (semantic_checker.ml)
   9.1.6 Java Builder (java_builder.ml)
   9.1.7 JAST (jast.ml)
   9.1.8 Code Generator (code_gen.ml)
   9.1.9 End-to-end Runner (tbag.ml)
   9.1.10 Compile and Run Script (run_tbag.sh)
   9.1.11 Test Script (run_tests.sh)
   9.1.12 Helper script for testing (java_tbag.sh)
   9.1.13 Makefile
   9.1.14 Standard TBAG library (stdlib.tbag)
9.1.13 Type conversion TBAG library (stdlib.tbag)
9.1.14 Driver functions (driver_functions.txt)
9.1.15 Test suite

9.2 Full Git Commit History
1 Introduction

1.1 Motivation and Background

Text-based adventure games were first introduced in the late 1970’s, and while gameplay technology and graphics have evolved considerably since then, these games still remain popular and new ones are continuously being developed. These games may differ on some details, but many of them operate on similar principles - there is a map that a player explores which may contain different rooms, and the rooms may contain various non-player-characters and items with which the player can interact. As the player explores the world, he or she triggers different events. Because of these commonalities among different games, there is no doubt similarity in the development process. We propose developing TBAG, a language that makes it easy to build game elements and describe event-driven game logic.

1.2 Language Description

TBAG is designed to make text-based game creation a simple process. Developers who are familiar with the typical structure of a text-based RPG (role playing game) will find the structure of a TBAG program intuitive. As with most games modeling a roleplay structure, users can define the structures “Room,” “Item” and “NPC” (non-player character) unique to their game, then populate those structures as they wish. Subsequent to the data setup portion of the program, the user can then create some global variables which they can reference later.

An event-driven system is the primary unique feature of our language. Event handling is expressed in our language via a series of predicates attached to handlers that will be executed if the associated predicate evaluates to true. This style of handling gameplay makes game development simple, as the actions to take simply depend on the truth values of the predicates. Within this model the user can use predefined functions to easily present choices to a player, change the state of the game, and implement logic such as moving between rooms with ease. The user can also define their own functions to be used within the event-driven component.

In the end product, the player sees the game evolving and their choices expanding or changing depending on their previous input. What drives the change of state behind the scenes is an event loop that is always checking the predicates and executing the handlers if their associated predicate evaluates to true. We envision that this language style will significantly speed up the time it takes for the user to create their own text-based adventure games.
1.3 Style

For the purposes of this report, we refer to the “user” as the programmer who uses the TBAG language. We refer to the “player” as the end user who would play a game written in the TBAG language.
2 Tutorial

2.1 Simple Data Types and Syntax

TBAG uses a simple C-like syntax with curly braces indicating scope, parentheses for expressions and argument passing, and semicolons ending statements. There are 3 primitive data types: ints, booleans, and strings. The complex data type for use in the TBAG language is an array. Below is a series of statements demonstrating how to use these data types:

```
int a = 0;
boolean honest = true;
string message = "hello";
int [10] i;
i[0] = a;
```

2.2 Simple Program Structure Example

Every TBAG game lives in a single .tbag file and must include at least one event handler. The example below is a simple GCD program written in TBAG:

```
int a = 8;
int b = 36;

a == b {
    print(a);
    endgame;
}

a > b {
    a = a - b;
}

a < b {
    b = b - a;
}
```
2.3 Complex Data Types and Program Structure Example

TBAG has the built-in structure of a Room, NPC, and Item. To use them, the user must first define each with fields. Subsequent instances of rooms, npcs, and items will only have access to those pre-defined fields. The “name” field is built into Rooms. Using rooms also requires that the user declares at least two rooms, an adjacency between them, and a start room. Items and NPCs are defined and declared similarly, with no such requirement for adjacency or start.

Some built-in functions and keywords allow for easy manipulation of Rooms in the event-driven model. The “currentRoom” keyword maps to the current room that the player finds themselves in. The “->” operator takes a room, and allows the Player to “go to” that room. The getInputFromOptions() function takes one or more strings that will display input options to the player. Finally, the “input” keyword is a reserved variable of type string that captures player choice. Here is an example program showing the more complicated program structure with a game using Rooms:

```java
room {
    string description;
    boolean visited;
}

room Home {
    name = "Home";
    description = "My House";
    visited = false;
}

room Work {
    name = "Work"
    description = "My Work";
    visited = false;
}

Home <-> Work; /* specify adjacency */

start {Work}

boolean madeItHome = false;

currentRoom == Work {
    currentRoom.visited = true;
    print("Currently in: ");
    print(currentRoom.name);
    getInputFromOptions("Home");
    ->input
}
currentRoom == Home {
    currentRoom.visited = true;
    print("Currently in: ");
    print(currentRoom.name);
    madeItHome = true;
}

madeItHome{
    print("Good job making it home!");
    endgame;
}

2.4 Dealing with User Input

The getInputFromOptions function can be used to get input from the player. The general structure for calling the getInputFromOptions is as follows:

```java
getInputFromOptions("Closet", "LivingRoom");
```

At runtime, this function will prompts the user to enter “Closet” or “LivingRoom”. When a player enters acceptable input, that input is saved into the reserved variable input. If the user wishes to simply present all adjacent rooms as options, they may use the getInputAdjacentRooms function, which requires one argument:

```java
getInputAdjacentRooms(Room myRoom);
```

This will present the rooms adjacent to the myRoom as options and save their choice to the reserved variable input.

2.5 Compiling and Running a TBAG program

To compile and run a .tbag file, simply feed the file name as input to the “run_tbag.sh” script, as follows:

```bash
./run_tbag hello_world.tbag
```
The file will be compiled and the executable will be run.

2.6 Important Reminder

TBAG is a language for game development, so most importantly, remember to have fun!
3 Language Reference Manual

3.1 Lexical Elements

3.1.1 Identifiers

Identifiers are strings used for naming variables, functions, and instances of convenience structs (rooms, items, and NPCs). These identifiers are case sensitive. They can consist of letters, digits, and underscores, but should always start with a letter. These rules are described by the definitions involving regular expressions below:

\[
\text{identifier} := (\text{letter}) (\text{letter} | \text{digit} | \text{underscore})* \\
\text{digit} := '0' - '9' \\
\text{letter} := \text{uppercase\_letter} | \text{lowercase\_letter} \\
\text{uppercase\_letter} := 'A' - 'Z' \\
\text{lowercase\_letter} := 'a' - 'z'
\]

3.1.2 Keywords

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

\[
\begin{align*}
\text{int} & \quad \text{room} & \quad \text{if} & \quad \text{true} & \quad \text{func} \\
\text{string} & \quad \text{item} & \quad \text{else} & \quad \text{false} & \quad \text{print} \\
\text{boolean} & \quad \text{start} & \quad \text{while} & \quad \text{AND} & \quad \text{endgame} \\
\text{void} & \quad \text{npc} & \quad \text{return} & \quad \text{OR} & \quad \text{currentRoom} \\
\text{neg} & \quad & \text{NOT} & \quad & \\
\end{align*}
\]

3.1.3 Literals

Literals are constant string, numeric, or boolean values, such as “hello”, 66, or false. Each literal is immutable and has a specific data type corresponding to one of the primitive types.
mentioned in the previous sentence. No type casting is allowed. Trying to assign a literal to a variable of mismatching type will cause an error.

3.1.3.1 String Literals

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

An escaped character is a character that immediately follow a backslash ‘\’. The backslash signals to the compiler to interpret the escaped character in a special way, according to the following table:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>Insert a double quote at this point.</td>
</tr>
<tr>
<td>\</td>
<td>Insert a backslash at this point.</td>
</tr>
<tr>
<td>\n</td>
<td>Insert a newline at this point.</td>
</tr>
<tr>
<td>\t</td>
<td>Insert a tab at this point.</td>
</tr>
<tr>
<td>\r</td>
<td>Insert a carriage return at this point.</td>
</tr>
<tr>
<td>\b</td>
<td>Insert a backspace at this point.</td>
</tr>
<tr>
<td>\f</td>
<td>Insert a formfeed at this point.</td>
</tr>
</tbody>
</table>

Examples: “happy”; “I’m a sentence.\n This sentence will start on a new line.”

3.1.3.2 Integer Literals

Integer literals are whole numbers represented by a sequence of one or more digits from 0-9. Integers are assumed to be in decimal (base 10) format. Precede an integer with “neg ” to denote a negative number. Absence of the “neg” keyword denotes a positive integer.

Examples: 42; neg 666

3.1.3.3 Boolean Literals

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

Examples (exhaustive list): true; false
3.1.4 Operators

Operators are tokens that are utilized for performing actions on different elements. Common operations performed by such operators are addition, subtraction, and other mathematical processes. These will be discussed further in section 3.

Example: +

3.1.5 Delimiters

Delimiters are special tokens that:
1. separate other tokens
2. tell the compiler how to interpret associated tokens.

3.1.5.1 Parentheses and Braces

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

3.1.5.2 Commas

Commas are used to separate function arguments.

3.1.5.3 Brackets

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

3.1.5.4 Semicolon

A semicolon is used to terminate a sequence of code.

3.1.5.5 Curly Braces

Curly braces are used to enclose function definitions, blocks of code (including predicate handlers), room/item/NPC struct definitions, room/item/NPC data, and the starting room. In general, blocks enclosed within curly braces do not need to be terminated with semicolons.
3.1.6 Periods

Periods are used for accessing fields of a room, item, or NPC.

3.1.6 Whitespace

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

3.2 Data Types

TBAG is statically typed. The types of all variables are known at compile time and cannot be changed.

3.2.1 Primitive Data Types

3.2.1.1 int

These are 32-bit signed integers that can range from \(-2,147,483,648\) to \(2,147,483,647\).

3.2.1.2 string

All text values will be of this type.

3.2.1.3 boolean

A truth value that can be either true or false.

3.2.1.4 void

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

3.2.2 Non-Primitive Data Types

3.2.2.1 Arrays

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

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

```c
int[5] myArray;
```

declares an array named myArray that can hold 5 integers.

3.2.2.1.2 Accessing and setting array elements

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

```c
myArray[1];
```

returns the element in myArray at index 1. Array elements can be set by accessing the element via the desired index in which to place the item and then assigning the desired value to the entry. For example:

```c
myArray[1] = 4;
```

sets the element in myArray at index 1 to 4.

3.2.2.1.3 Array length

To get an array’s length, simply call the arrLen function on the array. For example:

```c
int size = arrLen(my_array);
```

3.2.2.2 Rooms

A room is a structure that represents a location in the game that the player can go to or be in.
3.2.2.2.1 Defining rooms

The structure of a room can differ from program to program, but must be the same within one .tbag file. It is defined in a TBAG program by a “room” keyword followed by a block of declarations that contain that room’s fields and the types of those fields, similar to struct declarations in C. A “name” field with type string is automatically defined for you by the compiler. Example for declaring room structure:

```c
room {
    string description;
    boolean visited;
}
```

Note that if you wish to use rooms within your program, you must define a room structure. If you wish to have rooms with only name fields, you must still include a room definition, but you can leave it blank. For example:

```c
room {}
```

means that all the rooms in this particular file will only include the automatically defined name field.

3.2.2.2 Initialization of data for rooms

To define data for a particular room, use a “room” keyword followed by a block of assignments to the fields defined in the room definition. For example, given the above room definition, this is how to create a room with identifier home, name “home”, description “My House”, and visited = false:

```c
room home {
    name = "home";
    description = "My House";
    visited = false;
}
```

Repeat as needed to create additional rooms. Note that you could assign a value to the name field even though such a field was not declared in the example room definition in the previous
section. This is because a *name* field is automatically defined by the compiler for you. Note that if you wish to use rooms in your program, you must have at least two rooms per TBAG file.

### 3.2.2.2.3 Setting adjacencies

You can specify that two rooms are adjacent. This does not mean that the player can only access room1’s adjacent rooms if he/she is in room1. Declaring adjacencies are just a convenience for you so you can do things like present a room’s adjacencies to the player as options.

To specify that two rooms are adjacent, use the “<->” operator like so:

```plaintext
room1 <-> room2;
```

This will specify that room1 is adjacent to room2.

Note that if you wish to use rooms in your program, you must define at least one adjacency relationship.

### 3.2.2.4 Moving between Rooms

To move between rooms, use the goto function (->). For example:

```plaintext
->livingRoom
```

would move a player to the previously defined livingRoom.

### 3.2.2.5 Identifying the currentRoom

The currentRoom keyword identifies the location of the player in the game. You can use this variable to determine the current room, e.g.:

```plaintext
currentRoom == livingRoom
```

However, currentRoom should not be used to assign the location (the go-to -> operator should be used for that instead).
3.2.2.6 Start location

Start declarations are required when a user incorporates rooms into their program. This start declaration, which comes immediately after the adjacency declarations, sets the location (currentRoom) of the player to start the game. To specify this start room, utilize the following syntax:

```
start {Home}
```

where “Home” is a room that has been declared previously in the program.

3.2.2.3 NPCs and Items

NPC stands for non-player character. Like a room, this is also a structure with fields that you can define, modify, and access. Including NPCs is optional. Like rooms and NPCs, items are structs that you can configure for your convenience. The item datatype serves the same purpose as the NPC: to define structs that hold data. Even though items and NPCs work the same way, they are different data types because they are conceptually separate, allowing you to neatly categorize data in your program.

3.2.2.3.1 Defining NPCs and Items

Define the structure of an NPC and Item the same way you define the structure of a room. With NPCs and items, however, no name field is automatically defined for you so you must define that yourself. Example NPC and item definition:

```
npc {
    string name;
    boolean friendly;
}

item {
    string name;
    int usefulness;
}
```

The above block of code means that NPCs in the program will have a string-typed name field and a boolean-typed friendly field. Items will have a string-typed name field and an integer-typed usefulness field.
3.2.3.2 Initialization of data for NPCs and Items

Defining data for an NPC and Item works the same way as defining data for a room, using the “npc” and “item” keywords instead of the “room” keyword.

```plaintext
cmp home {
    name = "Greg";
    friendly = false;
}

text wheelbarrow {
    name = "wheelbarrow";
    usefulness = 10;
}
```

3.3 Expressions and Operators

3.3.1 Expressions

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

3.3.2 Operators

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Element Access</td>
<td>Left-to-Right</td>
</tr>
<tr>
<td>NOT</td>
<td>Logical Not</td>
<td>Right-to-Left</td>
</tr>
<tr>
<td>* /</td>
<td>Multiplication, division</td>
<td></td>
</tr>
<tr>
<td>+ -</td>
<td>Addition, subtraction</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>Inequality Operators: Less Than, Less Than Or Equal</td>
<td>Left-to-Right</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Inequality Operators: Greater Than, Greater Than Or Equal</td>
<td></td>
</tr>
<tr>
<td>== !=</td>
<td>Equal, Non-Equal, String Equal</td>
<td></td>
</tr>
<tr>
<td>~~</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.4 Statements

3.4.1 The if Statement

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

```
if (condition) {
    action1;
}
else {
    defaultaction;
}
```

“if” must be followed with “else,” although the else may have no statement associated with it:

```
if (condition) {
    action1;
} else {
}
```

3.4.2 The while Statement

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

```
while (condition) {
    action1;
}
```
3.5 Functions

3.5.1 Function Definitions

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

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

3.5.2 Calling Functions

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

```plaintext
sum(1, 2);
```

3.6 Program Structure and Scope

3.6.1 Program Structure

A TBAG program must live entirely within one source file.

The primary components of a TBAG program, in order, are
1. Import Library Statements
2. Room Definition
3. Room Declarations
4. Adjacency Declarations
5. Start Declaration  
6. NPC Definition  
7. NPC Declarations  
8. Item Definition  
9. Item Declarations  
10. Global Variable Declarations  
11. Event Driven Predicates and associated Handlers  
12. Function Declarations  

The language is set up in a way such that some of the primary components can be excluded and the program will still compile and run effectively. The 16 options for program structure are as follows, where each option lists the number associated with a primary component defined above:

- All Components: 2,3,4,5,6,7,8,9,10,11,12  
- No Items: 2,3,4,5,6,7,10,11,12  
- No NPCs: 2,3,4,5,8,9,10,11,12  
- No Rooms: 6,7,8,9,10,11,12  
- No Items, No NPCs: 2,3,4,5,10,11,12  
- No Rooms, No Items: 6,7,10,11,12  
- No Rooms, No NPCs: 8,9,10,11,12  
- No Rooms, No NPCs, No Items: 10,11,12  

And all of the above with and without import library statements (1).

This gives the user flexibility to use the language for all kinds of programs and games.

3.6.2 Event Driven System

The primary unique feature of the TBAG language is the event driven system, which is expressed using event handlers. Each handler consists of a predicate that evaluates to true or false, and an associated block of code that will run if the predicate evaluates to true. These event handlers will be continuously evaluated throughout gameplay in a loop. For example, in the handlers below:

```plaintext
currentRoom == livingRoom {  
    print("welcome");  
}

currentRoom == den {  
    print("this does not print");  
}
```
if the currentRoom were the livingRoom, the first predicate previous to the braces would be evaluated as true, and therefore the welcome message would be printed. However, the second predicate (currentRoom == den) would evaluated to false and therefore the message “this does not print” will not be printed. Event andlers are evaluated based on the order they are defined in the program.

3.6.3 Ending the Game

There is only one way to terminate a program, and that is by specifying a location for the game to end within a handler of a predicate block. To do so, simply include the statement “endgame” where you wish to terminate the program. For example

```c
  currentRoom == finalRoom {
    endgame;
  }
```

would end the game once the user enters the finalRoom.

3.6.4 Importing Libraries

Use the #import syntax at the top of a .tbag file to import TBAG libraries. A TBAG library is a file with extension .tbag that consists of a list of TBAG functions. To get access to a TBAG library’s functions, simply copy the library file to the lib/ folder if it’s not there already, then add the line “#import [libraryName]” to the top of any TBAG program that uses that library.

```c
  #import stdlib
  #import typeConversionLib
```

The above syntax imports the stdlib library and the typeConversionLib library to the including file.

3.6.5 Scope

Any declarations made within the program that are not within one the block of an if statement, a while statement, and a function definition are available for reference any point later in the program. Declarations made within blocks of an if statement, a while statement, or a function definition are only available for reference within that block. Declarations are never visible to any code that comes before it in the program.
3.7 Built-in Functions

3.7.1 The print function

The print function can be used to print out strings, integers, and booleans to the command line. The general structure for calling the print function is as follows:

```
print("welcome to the jungle");
print(666);
```

Anything within the parentheses will be printed; it must be of type string, integer, or boolean. Note that if a user wishes to print on a new line, a new line must be explicitly specified.

3.7.2 The getInputFromOptions function

The getInputFromOptions function can be used to get input from the user. The general structure for calling the getInputFromOptions is as follows:

```
getInputFromOptions("Closet", "LivingRoom");
```

At runtime, this function will prompts the user to enter “Closet” or “LivingRoom”. When an acceptable input is entered, that input will be saved into the reserved variable input.

3.7.3 The getInputAdjacentRooms function

If the programmer wishes to simply present all adjacent rooms as options, they may utilize the getInputAdjacentRooms function, which requires a single room argument:

```
getInputAdjacentRooms(currentRoom);
```

This will present the rooms adjacent to the current room as options and save their choice to the reserved variable input.
3.7.4 The arrLen function

The arrLen function can be used find out how many elements are in an array. The generalstructure for calling the arrLen function is as follows:

```c
int len;
int[3] a;
len = arrLen(a);     /* len is now 3 */
```

The sole argument of arrLen() is an array.
### 3.8 Context Free Grammar

```
program  →  rdef rdecl_list adecl_list start ndef ndecl_list idef idecl_list vdecl_list predicate_list
fdecl_list EOF  →  rdef rdecl_list adecl_list start ndef ndecl_list vdecl_list predicate_list fdecl_list EOF
               →  rdef rdecl_list adecl_list start idef idecl_list vdecl_list predicate_list fdecl_list EOF
               →  rdef rdecl_list adecl_list start vdecl_list predicate_list fdecl_list EOF
               →  ndecl ndecl_list idef idecl_list vdecl_list predicate_list fdecl_list EOF
               →  ndecl ndecl_list vdecl_list predicate_list fdecl_list EOF
               →  idef idecl_list vdecl_list predicate_list fdecl_list EOF
               →  vdecl_list predicate_list fdecl_list EOF

data_type  →  INT
               →  STRING
               →  VOID
               →  BOOLEAN

pred_stmt  →  expr LBRACE vdecl_list stmt_list RBRACE
predicate_list  →  /* nothing */
               →  predicate_list pred_stmt
fdecl_list  →  /* nothing */
               →  fdecl_list fdecl
fdecl  →  FUNC data_type ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
formals_opt  →  /* nothing */
               →  formals_list
formal_list  →  data_type ID
               →  formal_list COMMA data_type ID
actuals_opt  →  /* nothing */
               →  actuals_list
actuals_list  →  expr
               →  actuals_list COMMA expr
vdecl_list  →  /* nothing */
               →  vdecl_list vdecl
vdecl  →  data_type LBRACK expr RBRACK ID SEMI
               →  data_type ID SEMI
               →  data_type ID ASSIGN expr SEMI
rdef  →  ROOM LBRACE vdecl_list RBRACE
rdecl_list  →  rdecl rdecl_list
rdecl  →  ROOM ID LBRACE stmt_list RBRACE
start  →  START LBRACE ID RBRACE
adecl_list  →  adecl
               →  adecl_list adecl
adecl  →  adj_list SEMI
adj_list  →  ID ADJ ID
ndef  →  NPC LBRACE vdecl_list RBRACE
ndecl_list  →  /* nothing */
ndecl  →  NPC ID LBRACE stmt_list RBRACE
idef  →  ITEM LBRACE vdecl_list RBRACE
idecl_list  →  /* nothing */
```
→ idecl_list idecl
idecl  →  ITEM ID LBRACE stmt_list RBRACE
stmt_list → /* nothing */
            → stmt_list stmt
stmt → expr SEMI
RETURN expr SEMI
LBRACE stmt_list RBRACE
IF LPAREN expr RPAREN stmt ELSE stmt
WHILE LPAREN expr RPAREN stmt
GOTO ID
expr → INT_LITERAL
NEG INT_LITERAL
STRING_LITERAL
END
BOOL_LITERAL
ID
expr PLUS expr
expr MINUS expr
times expr
expr TIMES expr
expr DIVIDE expr
expr EQ expr
expr STREQ expr
expr NEQ expr
expr LT expr
expr LEQ expr
expr GT expr
expr GEQ expr
expr AND expr
expr OR expr
NOT expr
ID ASSIGN expr
ID LBRACK expr RBRACK ASSIGN expr
ID LBRACK expr RBRACK
ID LPAREN actuals_opt RPAREN
LPAREN expr RPAREN
ID ACCESS ID
4 Project Plan

4.1 Process used for planning, specification, development and testing

Our team met once a week as a team with our T.A. Lixin to gauge progress and discuss next steps and milestones. We met on an as-needed basis at least once a week, with those who could attending meetings and contributing to parts as necessary. We used GroupMe to coordinate day-to-day logistics and Google Drive and Github as version control for code and working documents.

Generally we identified specific roadblocks towards the next concrete task ahead of time and assigned the least busy person to get a head start on that task. Then that person(s) would disseminate their code, progress made, and lessons learned to the rest of the group members. We made an effort to have discrete but concrete tasks for each member of the team at all times, rotating heavier tasks as necessary depending on schedules. In the case where a task required more than a single person to accomplish, we used pair programming so one person would type up the code and debug while the other would read through existing past projects or class resources for analogous examples.

On November 18, the project managers met to come up with a projected internal timeline for when components of the code base were due to keep everyone on track. Of course with so many components that were co-dependent, difficulties arose in finishing one before working on another. When roadblocks occurred, we met as a team to work through them together. As outlined in the timeline, most if not all projected dates were not met, but the planning that occurred prior to executing tasks helped immensely with workflow.

In early December, we changed TA’s and had to drastically alter the direction of our project, which pushed back deliverables significantly. After meeting with Professor Edwards, we added the event-driven component to our game which would make the TBAG language more algorithmically interesting and unique. As a result, code gen was finished mid-December and semantic checking was not finished until nearly a week later.

In theory, each team member was to thoroughly test their code against the test suite before making any pushes upstream to master. Due to the burden of working on the semantic checker, it fell to project manager Julie to write the success tests for a majority of the features. This first test suite was used during writing the code generator, then passed to semantic checking. Then semantic checking must pass the tests before merging back to the master code base. When semantic checking was passing most success cases, a full suite of fail tests were added to specifically check more fine-tuned semantic checking. The full test suite was not completed until December 21.
4.2 Style Guide

The following outlines our style guide for OCaml, version control, Bash, and Java.

4.2.1 OCaml

- snake_case
- 4 space indentation
- 76 character limit lines (wrap beyond)
- comments included if logic is at all confusing
- pattern matching: no pipe for first case, pipe for all remaining cases
- “then” should follow “if” in same line
- “in” should follow “let” in same line
- newlines between function definitions
- explicitly write out types for function arguments

4.2.2 Version Control (Git)

- Branch from master when implementing new features
- Commit often to allow for easy rollback of work if necessary
- Rebase from master before merge
- When ready to merge, open pull request
  - Have someone else review your pull request
- Merge back into master, ensure tests still passing
- Create new branch to incorporate new features, delete old

4.2.3 Bash

- separate actions into discrete statements where possible
- one line per statement
- one space between each token

4.2.4 Java

In regards to generating Java code, we tried to follow standard Java coding conventions.
### 4.3 Project timeline

<table>
<thead>
<tr>
<th>Date (actual)</th>
<th>Date (projected)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 16</td>
<td>Sep 14</td>
<td>Finalize team members</td>
</tr>
<tr>
<td>Sep 30</td>
<td>Sep 30</td>
<td>Submit project proposal</td>
</tr>
<tr>
<td>Oct 19</td>
<td>Oct 26</td>
<td>Produce LRM</td>
</tr>
<tr>
<td>Oct 24</td>
<td>Oct 20</td>
<td>Produce CFG</td>
</tr>
<tr>
<td>Oct 25</td>
<td>--</td>
<td>First commit, creation of project directory</td>
</tr>
<tr>
<td>Oct 26</td>
<td>Oct 26</td>
<td>Submit LRM and CFG</td>
</tr>
<tr>
<td>Nov 8</td>
<td>Nov 7</td>
<td>Develop Scanner and Parser (preliminary) for existing CFG</td>
</tr>
<tr>
<td>Nov 12</td>
<td>Nov 16</td>
<td>Hello World compiles, Hello World regression test complete</td>
</tr>
<tr>
<td>Dec 3</td>
<td>Dec 1</td>
<td>Switch TAs</td>
</tr>
<tr>
<td>Dec 8</td>
<td>Dec 9</td>
<td>Produce Java Target</td>
</tr>
<tr>
<td>Dec 9</td>
<td>Nov 23</td>
<td>Produce SAST (preliminary)</td>
</tr>
<tr>
<td></td>
<td>Dec 11</td>
<td>Produce JAST/Code_gen</td>
</tr>
<tr>
<td>Dec 20</td>
<td>Dec 11</td>
<td>Produce SAST/semantic checker</td>
</tr>
<tr>
<td>Dec 21</td>
<td>Dec 9</td>
<td>Produce Full Test Suite</td>
</tr>
<tr>
<td>Dec 21</td>
<td>Dec 11</td>
<td>Produce Full Sample Test Demo</td>
</tr>
<tr>
<td>Dec 21</td>
<td>Dec 21</td>
<td>Final Report</td>
</tr>
<tr>
<td>Dec 22</td>
<td>Dec 22</td>
<td>Final Presentation</td>
</tr>
</tbody>
</table>
4.4 Roles and Responsibilities

Our group had a flat hierarchical structure until mid-November, when we divided up roles with Julie as Project Manager, Brian as System Architect, Gregory as Language Guru, Maria as Tester, and Iris as Project Manager/Tester. However throughout the course of the project we had to switch to different tasks depending on differing needs at that moment. The final roles and deliverables contributed to by each member are as follows.

**Julie Chien**
*Project Manager, Tester*

- Project planning
- Preprocessor
- Language/feature design - event-driven system, built in functions, libraries
- Java prototypes for what compiled output should look like
- Success tests and gameplay/input tests
- Demo games

**Brian Slakter**
*System Architect, Language Guru*

- Scanner, parser
- Code gen, java builder
- Final Report

**Gregory Chen**
*System Architect, Language Guru*

- Designing the AST and CFG
- Scanner, parser
- Code gen, java builder

**Maria van Keulen**
*System Architect, Tester*

- Test suite script modeling Edwards’s for Micro C
- Semantic checker modeling Edwards’s slides
- Fail test suite

**Iris Zhang**
*System Architect, Project Manager*

- Semantic checker modeling Edwards’s slides
- Final Report
- Final Presentation
4.5 Software development environment used

- Languages: OCaml, Java (library)
- Programming Editor: Sublime, vim
- Version Control: Git, Github
- Documentation: Google Drive

4.6 Project log

4.6.1 Events Log

- September 16 - Met as group for the first time
- September 19 - Met as group with Edwards to discuss project idea
- September 20 - Met as group to discuss system architecture - finite state machine-based or action menu based
- September 21-30 - Develop project proposal
- October 19 - Brian writes LRM
- October 24 - Greg writes CFG
- November 8 - Greg writes Scanner and Parser for CFG
- November 12 - 17 - Brian writes basic code generation
- November 17 - Maria begins working on Test Suite
- November 17 - Julie begins working on Java Target Mockup
- November 17 - Iris begins working on Semantic Checker
- November 18 - Julie and Iris meet to discuss project plans, come up with project timeline and preliminary deadlines for team
- November 21 - Greg and Brian work on JAST
- November 23 - Greg and Brian met with Edwards, learned we were heading in the wrong direction and got suggestions for ways to improve our language and its complexity
- November 25 - Julie and Iris meet with Edwards to expanding our language idea, brainstormed new features
- November 27-28 - Julie developed ideas for event driven gameplay architecture, syntax, and implementation
- November 28 - Iris and Maria meet to work on semantic checker
- November 29 - Julie met with Edwards to get feedback on and further develop event driven gameplay design
- November 29 - Met as group to plan implementation of event-driven gameplay
- December 1 - Simple predicate / handler syntax
- December 3 - Team met with new TA David Arthur to get help with semantic analyzer and other implementation details
- December 3 - 14 - Language specific code generation (Brian and Greg)
December 10 - 21 - Semantic Checker (Iris and Maria)
December 14 - 16 - Julie writes success tests
December 16 - Standard library started
December 17 - 18 - LRM Revision
December 20 - Julie implemented #import to import external .tbag libraries
December 19 - 21 - Demo build, final report, bug fixes
December 22 - Presentation

4.6.2 Git Commit History

The team’s Git handles:
- Julie: jj-ian
- Maria: mvankeulen94
- Greg: gregorychen3
- Iris: iz2140
- Brian: bsakter

The full commit history is included as Appendix 2.
4.6.3 Git Merge with Master History

- Array improvements
  - #21 opened 10 days ago by gregorychen3

- Dot field access
  - #20 opened 10 days ago by gregorychen3

- Implemented dispAdj built in func. helloworld program tests.
  - #19 opened 12 days ago by gregorychen3

- Jj ian/java target
  - #18 opened 13 days ago by jj-ian

- Boilerplate code for java scanner now implemented.
  - #17 opened 13 days ago by gregorychen3

- More java target stuff, adding sub-.gitignore in Java Target folder s...
  - #16 opened 16 days ago by jj-ian

- Izhang/east
  - #15 opened 18 days ago by iz2140

- Bool literals
  - #14 opened 18 days ago by gregorychen3

- Predicate syntax
  - #13 opened 18 days ago by gregorychen3

- Boolean data type implemented
  - #12 opened 15 days ago by gregorychen3

- Now generating Npc.java and Item.java
  - #11 opened 19 days ago by gregorychen3

- Gregorychen3/print room fields
  - #10 opened 21 days ago by gregorychen3

- Gregorychen3/prettier print rooms
  - #9 opened 28 days ago by gregorychen3

- Pattern matching taken care of - should be good
  - #8 opened 29 days ago by tricker

- Jj ian/java target
  - #7 opened 29 days ago by jj-ian
### 4.6.4. Active Branches (as of December 21)

<table>
<thead>
<tr>
<th>Active branches</th>
<th>Pull requests</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sem_check_fail</code> Updated 13 minutes ago by mjvankouwen94</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td><code>jj-lan/tests</code> Updated 9 hours ago by jj-lan</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><code>sem_check_tests</code> Updated 21 hours ago by iz2140</td>
<td></td>
<td>38</td>
</tr>
<tr>
<td><code>izhang/sem_check_updated</code> Updated a day ago by mjvankouwen94</td>
<td></td>
<td>73</td>
</tr>
<tr>
<td><code>sem_check_new</code> Updated 3 days ago by mjvankouwen94</td>
<td></td>
<td>136</td>
</tr>
<tr>
<td><code>hookup</code> Updated 3 days ago by bsalakher</td>
<td></td>
<td>134</td>
</tr>
<tr>
<td><code>bs_sem_check</code> Updated 5 days ago by bsalakher</td>
<td></td>
<td>139</td>
</tr>
<tr>
<td><code>string_equals</code> Updated 10 days ago by ggregorychen3</td>
<td></td>
<td>141</td>
</tr>
<tr>
<td><code>izhang/sast</code> Updated 18 days ago by iz2140</td>
<td></td>
<td>191</td>
</tr>
<tr>
<td><code>bsalakher/pattern_matching_pret</code> Updated 29 days ago by bsalakher</td>
<td></td>
<td>236</td>
</tr>
<tr>
<td><code>bsalakher/driver-prettystart</code> Updated 29 days ago by bsalakher</td>
<td></td>
<td>244</td>
</tr>
<tr>
<td><code>bsalakher/sast_to_jast</code> Updated 29 days ago by bsalakher</td>
<td></td>
<td>255</td>
</tr>
<tr>
<td><code>bsalakher/improve-language-synt...</code> Updated a month ago by bsalakher</td>
<td></td>
<td>259</td>
</tr>
<tr>
<td><code>master_20151119</code> Updated a month ago by jj-lan</td>
<td></td>
<td>275</td>
</tr>
</tbody>
</table>
5 Architectural Design

5.1 Overview

The TBAG compiler takes a .tbag file and feeds it through a series of steps that eventually converts this source code to compiled Java code. The end-to-end process, including running the resulting executables, can be performed in one step using the “run_tbag.sh” script, passing the tbag file created by the user as the one input to this script. Below is a diagram depicting this process, the details of which we will discuss further in the next few sections.
5.2 Preprocessor

The preprocessor looks for #import [libraryName] statements in the .tbag file and copies the specified libraries into the functions section of a copy of the original .tbag file. The resultant file is then fed to the compiler.

Contributions from Julie.

5.3 Scanner

The scanner is responsible for reading in the tbag file and decomposing the full text of the source code into a series of prespecified tokens. The scanner continues to read input until it has recognized that a full token has been entered, and passes these token through to the parser. This reduces the work of the parser to simply understanding the structure of a program from tokens rather than a mass of source code.

Contributions from full team.

5.4 Parser

The parser receives tokens as input from the scanner. From these tokens, the parser will construct an Abstract Syntax Tree - a high-level representation of the program as a whole. Different subtrees within the overall AST will represent different components of the program, such as a function definition, variable declarations, etc. The grouping of tokens into these subtrees is defined by the CFG represented in the parser. The parser passes this constructed abstract syntax tree through to the next stage of compilation - the semantic checker.

Contributions from full team.

5.5 Semantic Checker

The semantic checker receives an abstract syntax tree as input from the parser. The semantic checker is a very important step in the compilation process, as it ensures that the program is correct beyond simple syntax check. The semantic checker walks through the syntax tree and confirms that the semantics of the program are correct. This step ensures that there are no type mismatch issues, references to undeclared variables or functions, and no issues with referencing elements out of scope. This step also checks the logic of the more TBAG-specific elements. For example, if a user defines a Room and its fields, the semantic checker will ensure that all these fields are initialized within
declarations of room instances. The compilation process will only continue if the semantic checker has traversed the entire AST without recognizing any semantic errors. Otherwise, an error will be thrown informing the user of their mistake, and further compilation will halt.

*Contributions from Iris, Maria.*

5.6 Java Builder

The java builder receives a semantically checked abstract syntax tree as input from the semantic checker. Because the general structure of the .tbag program is different from that of the corresponding java program, this step takes the different components of the TBAG program and rearranges them in a way such that the subsequent step of code generation is a simple process. For example, room, item, and NPC definitions are separated out as these will become separate java classes. All other code is grouped into the Driver.java class, from which the program will be run. The room, item, and NPC declarations will be grouped with the event-driven portion as these will both end up in the main method of the Driver class. The newly organized Java Abstract Syntax Tree is then passed through to the code_gen to produce java code.

*Contributions from Brian, Greg.*

5.7 Code Generator

The code generator receives a java abstract syntax tree as input from the java builder. This step will produce 4 distinct classes. The room definition logic is used to create the Room.java class, NPC definition logic is used to build the Npc.java class, and the item definition logic is used to create the Item.java class. All other portions of the program will be placed in the main class, Driver.java. The generation of code for Driver.java is broken down into 3 main parts: creation of global variables, creation of the main method, and creation of additional functions that may be called by statements within the main method. The main method is created by declaring instances of the user-defined rooms, adjacencies, start location, items, NPCs, and placing the event-driven logic into a while loop. Some additional library functions are also placed below the user-defined functions to allow for use of these functions in the main class as well. From this code generation step, we output these 4 java files that will then be compiled via the java compiler, and then the Driver may be executed to begin gameplay.

*Contributions from Brian, Greg.*
6 Testing

6.1 Representative Language Programs

6.1.1 GCD

6.1.1.1 GCD Source

```java
int a = 8;
int b = 36;

if (a == b) {
    print(a);
    endgame;
}

if (a > b) {
    a = a - b;
}

if (a < b) {
    b = b - a;
}
```

6.1.1.2 GCD Target

```java
import java.util.*;

public class Driver {

    public static Scanner scanner;
    public static Room currentRoom;
    public static String input = "";
    public static HashMap<String, Room> roomMap = new HashMap<String, Room>();
    public static int b = 8;
    public static int a = 36;

    public static void main(String[] args) {
        scanner = new Scanner(System.in);
        currentRoom = null;
        while (true) {
            if(a==b){
                System.out.print(a);
                break;
            }
            if(a>b){
                a = a-b;
            }
            if(a<b){
                b = b-a;
            }
        }
        scanner.close();
    }
```
// this is what happens when u do player->room
public static void movePlayerToRoom(Object room) {
    if (room instanceof Room) {
        currentRoom = (Room) room;
    } else {
        Room update = roomMap.get(room);
        currentRoom = update;
    }
}

// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid input.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.
// Arguments:
// String[] acceptableInputs -- the list of acceptable inputs
public static void promptForInput(String[] acceptableInputs) {
    System.out.println("Choose from one of the following options: ");
    for (String option : acceptableInputs) {
        System.out.print(option + " ");
    }
    System.out.println();
    // loop until player enters valid input
    input = scanner.nextLine();
    System.out.println("Input: " + input);
    while(!Arrays.asList(acceptableInputs).contains(input)) {
        System.out.println("Invalid Input. Try again.");
        input = scanner.nextLine();
        System.out.println("Input: " + input);
    }
    System.out.println();
    System.out.println();
}

// Gets all the adjacencies for the room entered as argument and displays these adjacencies to player.
// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid adjacency.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.
// pretty much exactly same as promptForInputs(), except it takes in a room as an argument instead of a list of strings
public static void getInpPutAdjacentRooms(Room room) {
    String[] acceptableInputs = new String[room.adjRooms.size()];
    int i = 0;
    for (Room r : room.adjRooms) {
        acceptableInputs[i] = r.name;
        i++;
    }
    System.out.println("Choose from one of the following options: ");
    for (String option : acceptableInputs) {
        System.out.print(option + " ");
    }
    System.out.println();
// loop until player enters valid input
input = scanner.nextLine();
System.out.println("Input: " + input);
while(!Arrays.asList(acceptableInputs).contains(input)) {
    System.out.println("Invalid Input. Try again.");
    input = scanner.nextLine();
    System.out.println("Input: " + input);
}
System.out.println();
System.out.println();

Note that the Java target code has been indented for readability; the actual compiler output is not so nicely indented.

6.1.2 Mini-game

6.1.2.1 Mini-game Source

```java
# import stdlib
# import typeConversionLib

room {};

room Closet { name = "Closet"; }
room Bedroom { name = "Bedroom"; }
room Wall { name = "Wall"; }
room Kitchen { name = "Kitchen"; }

Closet <-> Bedroom;
Closet <-> Wall;
Kitchen <-> Wall;
Kitchen <-> Bedroom;

start { Closet }
npc { string roomName; }
npc Cat { roomName = "Bedroom"; }
item { string roomName; }
item Cheese { roomName = "Kitchen"; }

boolean started = false;
boolean cheeseEaten = false;

NOT started {
    strPrintLine("You’re a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
```
6.1.2.2 Mini-game Target

```java
import java.util.*;

public class Driver {

    public static Scanner scanner;
    public static Room currentRoom;
    public static String input = "";
    public static HashMap<String, Room> roomMap = new HashMap<String, Room>();
    public static boolean cheeseEaten = false;
    public static boolean started = false;
    public static void main(String[] args) {
        scanner = new Scanner(System.in);
        Room Kitchen = new Room();
        roomMap.put("Kitchen", Kitchen);
        Kitchen.name = "Kitchen";

        Room Wall = new Room();
        roomMap.put("Wall", Wall);
        Wall.name = "Wall";

        Room Closet = new Room();
        roomMap.put("Closet", Closet);
        Closet.name = "Closet";

        Room Bedroom = new Room();
        roomMap.put("Bedroom", Bedroom);
        Bedroom.name = "Bedroom";

        Bedroom.setAdjacent(Kitchen);
        Wall.setAdjacent(Kitchen);
        Wall.setAdjacent(Closet);
    }
}
```
currentRoom = Closet;
Npc Cat = new Npc();
Cat.roomName = "Bedroom";

Item Cheese = new Item();
Cheese.roomName = "Kitchen";

while (true) {
    if(!started){
        strPrintLine("You're a mouse.");
        started = true;
    }
    if(true){
        printCurrentRoomInfo();
        getInputAdjacentRooms(currentRoom);
        movePlayerToRoom(input);
    }
    if(currentRoom.name.equals(Cat.roomName)){
        System.out.print("You got eaten by the cat.");
        break;
    }
    if(currentRoom.name.equals(Cheese.roomName)&&!cheeseEaten){
        System.out.print("Nice!! You ate the cheese!");
        cheeseEaten = true;
    }
}
scanned.close();
}

public static void intPrintLine(int a){
    System.out.print(a);
    System.out.print("\n");
}

public static void strPrintLine(String s){
    System.out.print(s);
    System.out.print("\n");
}

public static void boolPrintLine(boolean b){
    System.out.print(b);
    System.out.print("\n");
}

public static int intFromLetter(String letter){
    if (letter.equals("A")) {return 0;
    }else if (letter.equals("B") {return 1;
    }else if (letter.equals("C") {return 2;
    }else if (letter.equals("D") {return 3;
    }else if (letter.equals("E") {return 4;
    }else if (letter.equals("F") {return 5;
    }else if (letter.equals("G") {return 6;
    }else if (letter.equals("H") {return 7;
    }else if (letter.equals("I") {return 8;
    }else if (letter.equals("J") {return 9;
    }else if (letter.equals("K") {return 10;
    }else if (letter.equals("L") {return 11;
    }else if (letter.equals("M") {return 12;
    }else if (letter.equals("N") {return 13;
    }else if (letter.equals("O") {return 14;
    }else if (letter.equals("P") {return 15;
    }else if (letter.equals("Q") {return 16;
    }else if (letter.equals("R") {return 17;
public static String letterFromInt(int i) {
    if (i==0) { return "A";
    } else if (i==1) { return "B";
    } else if (i==2) { return "C";
    } else if (i==3) { return "D";
    } else if (i==4) { return "E";
    } else if (i==5) { return "F";
    } else if (i==6) { return "G";
    } else if (i==7) { return "H";
    } else if (i==8) { return "I";
    } else if (i==9) { return "J";
    } else if (i==10) { return "K";
    } else if (i==11) { return "L";
    } else if (i==12) { return "M";
    } else if (i==13) { return "N";
    } else if (i==14) { return "O";
    } else if (i==15) { return "P";
    } else if (i==16) { return "Q";
    } else if (i==17) { return "R";
    } else if (i==18) { return "S";
    } else if (i==19) { return "T";
    } else if (i==20) { return "U";
    } else if (i==21) { return "V";
    } else if (i==22) { return "W";
    } else if (i==23) { return "X";
    } else if (i==24) { return "Y";
    } else if (i==25) { return "Z";
    } else { return "-";
    }
}

public static void printCurrentRoomInfo() {
    System.out.println("Currently in: ");
    System.out.print(currentRoom.name);
    System.out.println("\n");
}

// this is what happens when u do player->room
public static void movePlayerToRoom(Object room) {
    if (room instanceof Room) {
        currentRoom = (Room) room;
    } else {
        Room update = roomMap.get(room);
        currentRoom = update;
    }
}

// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid input.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.

// Arguments:
// String[] acceptableInputs -- the list of acceptable inputs
public static void promptForInput(String[] acceptableInputs) {
    System.out.println("Choose from one of the following options: ");
    for (String option : acceptableInputs) {
        System.out.print(option + " ");
    }
    System.out.println();
    // loop until player enters valid input
    input = scanner.nextLine();
    System.out.println("Input: " + input);
    while(!Arrays.asList(acceptableInputs).contains(input)) {
        System.out.println("Invalid Input. Try again.");
        input = scanner.nextLine();
        System.out.println("Input: " + input);
    }
    System.out.println();
    System.out.println();
}

// Gets all the adjacencies for the room entered as argument and displays these adjacencies to player.
// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid adjacency.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.
// pretty much exactly same as promptForInputs(), except it takes in a room as an argument instead of a list of strings
public static void getInputAdjacentRooms(Room room) {
    String[] acceptableInputs = new String[room.adjRooms.size()];
    int i = 0;
    for (Room r : room.adjRooms) {
        acceptableInputs[i] = r.name;
        i++;
    }
    System.out.println("Choose from one of the following options: ");
    for (String option : acceptableInputs) {
        System.out.print(option + " ");
    }
    System.out.println();
    // loop until player enters valid input
    input = scanner.nextLine();
    System.out.println("Input: " + input);
    while(!Arrays.asList(acceptableInputs).contains(input)) {
        System.out.println("Invalid Input. Try again.");
        input = scanner.nextLine();
        System.out.println("Input: " + input);
    }
    System.out.println();
    System.out.println();
}
Note that the Java target code has been indented for readability; the actual compiler output is not so nicely indented.

6.2 Testing Suite and Justification

Our test cases are separated into the following two groups: success tests and fail tests. Success tests are used to ensure that compiled TBAG programs are correctly operational, given the language specifications. Additionally, success tests are used to ensure that the semantic checker does not mistakenly throw errors for semantically correct programs. Fail tests are used to ensure that the semantic checker properly identifies semantic errors in TBAG programs and prohibits semantically incorrect programs from compiling. We named success tests with the prefix "test_" and fail tests with "fail_". All tests and their .out files can be found in the test/ directory.

6.2.1 Success Tests

The success tests were chosen to check the functionality of standard language aspects such as functions, variables, and loops in addition to TBAG-specific aspects such as room/item/npc usage and event handlers. The test cases for the aspects of the language that are consistent with microC were inspired by the tests for microC. Most features were tested at least twice; once with handlers and once with functions (e.g. test_gcd_func.tbag and test_gcd.handler1.tbag). The success tests evaluate both simple and complex source programs in order to ensure that simple operations as well as full game playthroughs work as expected.

Contributions: Julie

<table>
<thead>
<tr>
<th>test_0npc_0item_2rooms.out</th>
<th>test_fib_func.out</th>
<th>test_if_func4.tbag</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_0npc_0item_2rooms.tbag</td>
<td>test_fib_func.tbag</td>
<td>test_if_handler3.out</td>
</tr>
<tr>
<td>test_0npc_1item_0rooms.out</td>
<td>test_func.out</td>
<td>test_if_handler3.tbag</td>
</tr>
<tr>
<td>test_0npc_1item_0rooms.tbag</td>
<td>test_func.tbag</td>
<td>test_local_var_func.out</td>
</tr>
<tr>
<td>test_0npc_1item_2rooms.out</td>
<td>test_func2.out</td>
<td>test_local_var_func.tbag</td>
</tr>
<tr>
<td>test_0npc_1item_2rooms.tbag</td>
<td>test_func2.tbag</td>
<td>test_if_func.out</td>
</tr>
<tr>
<td>test_1npc_0item_0rooms.out</td>
<td>test_gcd_func.out</td>
<td>test_if_func.tbag</td>
</tr>
<tr>
<td>test_1npc_0item_0rooms.tbag</td>
<td>test_gcd_func.tbag</td>
<td>test_if_func2.out</td>
</tr>
<tr>
<td>test_1npc_0item_2rooms.out</td>
<td>test_gcd_func2.out</td>
<td>test_if_func2.tbag</td>
</tr>
<tr>
<td>test_1npc_0item_2rooms.tbag</td>
<td>test_gcd_func2.tbag</td>
<td>test_if_func3.out</td>
</tr>
<tr>
<td>test_1npc_1item_0rooms.out</td>
<td>test_gcd_handler1.out</td>
<td>test_if_func3.tbag</td>
</tr>
<tr>
<td>test_1npc_1item_0rooms.tbag</td>
<td>test_gcd_handler1.tbag</td>
<td>test_if_func4.out</td>
</tr>
<tr>
<td>test_1npc_1item_2rooms.out</td>
<td>test_gcd_handler2.out</td>
<td>test_local_var_handler.out</td>
</tr>
<tr>
<td>test_1npc_1item_2rooms.tbag</td>
<td>test_gcd_handler2.tbag</td>
<td>test_local_var_handler.tbag</td>
</tr>
<tr>
<td>test_add.out</td>
<td>test_gcd_handler3.out</td>
<td>test_loop_event.out</td>
</tr>
<tr>
<td>test_add.tbag</td>
<td>test_gcd_handler3.tbag</td>
<td>test_loop_event.tbag</td>
</tr>
</tbody>
</table>
6.2.2 Fail Tests

The fail tests were chosen by reading through the semantic checker program and identifying all areas where the semantic checker needs to throw an error. Using the success tests as templates, we modified variable declarations and assignments, function calls and definitions, room/item/npc-related elements, operator usage, predicate statements, and other additional aspects to purposefully attempt throwing the expected errors. Successful compilations of programs that were expected to throw errors and unsuccessful compilations of semantically correct programs indicated issues with our semantic checker, which we proceeded to fix.

**Contributions: Maria, Iris**
6.2.3. Input Tests

Input tests were written to test player input and handling of player input. They also demonstrated fuller functionality of game logic, including moving through rooms, room adjacencies, standard library calls, and built-in functions. The .in files contain simulated player input.

Contribution: Julie

6.3 Testing Automation and Scripts

To automate the testing of our code, we used Professor Edwards’s MicroC test runner script as a template and modified the appropriate sections to fit our build flow. In the following description we will use “test script” to refer to the combination of our primary test runner script as well as any helper scripts that we wrote.

Our test script proceeds as follows:
- `run_tests.sh` script attempts to test all .tbag source files prefaced with "test_" and "fail_" in the subdirectory "tests". This subdirectory should be in the same directory as the test script, since a relative path is used to refer to the tests directory.

- To run a subset of the tests, the user can supply as command line arguments the names of the specific source files.

- When the script begins to perform a test for a particular source file, a message of the form "-n <basename>" is outputted to the terminal screen, where <basename> represents the name of the source file with the ".tbag" extension removed.

- If a source file successfully compiles, the test script proceeds to run the compiled Java output. If a source file fails to compile, the test script saves the compilation error that was produced.

A test is deemed successful if the expected output of the test matches up with the output that was produced by the script, which is saved into a file named <basename>.out. If the program successfully compiles, <basename>.out contains the runtime output of the Java program. If the program fails with a compiler error, <basename>.out contains the compiler error. The output comparison procedure is consistent with that of the original microC test script.

Since our language supports games with user input, we needed to modify the test script to pass input to the running Java program. If a user intends to test a program that accepts user input, the user must include the substring "_input" inside the file name. When the test script processes tests with source files of that format, it feeds a file named <basename>.in, which must be supplied by the user inside the tests subdirectory, into the running Java program. The <basename>.in file should contain all desired inputs to the program, with each input separated by a newline.

Finally, our test script also includes our predefined TBAG library functions in the source file that gets passed to the compiler. There are additional features included in Professor Edwards’s test script that we incorporated into our own test script for completeness, such as command line options and the usage of a log file, but we did not use these features during our testing.

*Contributions: Maria*
7 Lessons Learned

7.1 Brian Slakter

I learned a number of lessons through my work on this project. First of all, OCaml is awesome! I was a bit confused by the style at first, but I then realized how intuitive the idea of pattern matching is. When coding up solutions to interview questions involving lists, there are 3 cases that almost always require different actions - an empty list, a list with only one element, and a list with more than one element. Pattern matching makes thinking about these cases very intuitive, and forces you to break down the problem in this exact way. Second of all, stepping back to think about the process as a whole is a very important step that should take place early on in the project. We spent a lot of time in the weeds of our code, but understanding how the different components of the compiler fit together was not fully realized until later in the project, which caused us to make some changes. Finally, start early!

7.2 Gregory Chen

Starting early is definitely the most important take away, but it was difficult to know what exactly to be doing early on. This where I think it is important to be looking at previous years’ projects’ code. This helps with finding a high level direction early on, which is extremely important. For example, if I had known how different pieces of the compiler were ultimately going to communicate, things would have been smoother. Furthermore, it is important to pay close attention early on to the professor’s distinction between a translator and a true compiler. Another important lesson that is better learned early than late is to meet with the professor occasionally in addition to the assigned TA.

7.3 Julie Chien

1. Meet with Professor Edwards early on to make sure your language idea and implementation plan are actually good. If you’re like most of us, you probably don’t know enough about language design to successfully assess the quality of your idea. I was often surprised by what was considered a “good” idea and what wasn’t.

2. Related to #1 -- If you are thinking about making a major change to your project (such as your compiled language choice), run it by Professor Edwards first to make sure it won’t kill your project for reasons that you may not be able to perceive with your current knowledge.

3. If you’re using Git, work on new branches when you’re implementing features, then merge back into master when your feature is complete and passes all tests. Know how to rebase. I learned about rebasing in this project and it was super useful.

4. Everyone tells you to start early, but “early” is a subjective term so I’ll break it down concretely. It’s not really possible to start THAT early because you don’t learn enough to write most of your compiler until the last month or so of the semester. That is fine as long as you prepare accordingly. Start learning OCaml at least a week before the first homework is due.
After you turn in your LRM, get your scanner and parser out of the way immediately -- within 3-5 days. Codegen and Semcheck are the most difficult pieces of the project, so give yourself at least that whole last month to work on them. Start writing tests as soon as you have a functional compiler; it’ll save you time in the long run.

5. Be careful if you’re compiling to Java. If you’re doing this, make sure the language you’re designing isn’t too similar to Java.

7.4 Iris Zhang

The most important lesson: compilers should not be taken for granted. Thorough error checking and descriptive, useful error reporting is so important, and the majority of the gnarly work is done by the semantic checker so definitely get an early start on it. Built in functions are a huge pain to implement and I see why languages can have tens of thousands of lines for their compiler. If I took a wild guess I would say we wrote about 5-10% of a truly complete production-ready semantic checker. I will never again complain about a wordy fussy language. They impose order and logic on the AST, which makes everything down the pipeline easier to handle.

Get input from a lot of different sources (most importantly the professor) early on. Plan ahead. Even when those plans don’t work out, the insight you gain about workflow and how different people tend to work better together/apart is crucial to pulling through at the end.

Diminishing returns is a thing. Get enough sleep!

7.5 Maria van Keulen

In other CS classes, you may use compilers to catch certain errors in your source code. In this class, you use your language’s source code to catch errors in your compiler. Flesh out your regression test suite as soon as you can. During the process of writing our semantic checker and integrating it into the rest of the build procedure, our solid success test suite was super helpful in making sure we didn’t break anything along the way.

Also, write code in pairs. You may spend several hours investigating an error to no avail, only to have someone else look at it and identify the problem within a matter of minutes.

Finally, try not to panic if things don’t go according to plan. We overcame a number of obstacles through our willingness to take alternative routes when our original ones weren’t working.
8 Acknowledgements

We would like to thank our TA for going above and beyond to help us out with this project. He took us on late into the semester as an extra team, but treated us like we had been assigned to him from the beginning. We would also like to thank our professor for taking the time to meet with us weekly, especially late into the semester, to help get our team on the right track.
# Appendix

## 9.1 Source Code

### 9.1.1 Preprocessor (importLibrary.py)

```python
#!/usr/bin/env python

# looks for instances of "#import libraryName.tbag" in command line arg and copies library files into another file called prog_w_stdlib.tbag
# Author: Julie Chien
# 11/20/2015

import re
import fileinput
import shutil
import sys

tbagFileName = sys.argv[1]
tempFileName = 'prog_w_stdlib.tbag'

# copy file to temp file
shutil.copyfile(tbagFileName, tempFileName)

# search for lines starting with #import
linePattern = re.compile(r'#import (\w+)')
tbagFile = open(tbagFileName, 'r')
libraries = []

for line in tbagFile:
    matches = linePattern.findall(line)
    for libName in matches:
        libraries.append(libName)

if len(libraries) > 0:
    lineToReplace = "#import " + libraries[0]
    libTxtToPasteIn = ""

    for libName in libraries:
        libFileName = "lib/" + libName + ".tbag"
        with open(libFileName, 'r') as myfile:
            data=myfile.read()
        libTxtToPasteIn += data

    # if function is found in file, paste imported libraries before function block
    funcfound = False
    for line in fileinput.input(tempFileName, inplace=True):
        if funcfound == False:
```

54
if "func " in line:
    line = line.replace("func ", libTxtToPasteIn + "func ")
    funcfound = True
    print(line)

# if no function is found, paste imported libraries to end of file
if funcfound == False:
    with open(tempFileName, 'a') as f:
        f.write(libTxtToPasteIn)

for line in fileinput.input(tempFileName, inplace=True):
    if not re.search(linePattern, line):
        print(line)

9.1.1 Scanner (scanner.mll)

(* Authors: All *)

{ open Parser }

rule token = parse
    [\s|\t|\n|\r] { token lexbuf } (* Whitespace *)
    | "/*" { comment lexbuf }  (* comment lexbuf *)
    | func { FUNC }  (* comment lexbuf *)
    | "room" { ROOM }
    | "start" { START }
    | "endgame" { END }
    | "<->" { ADJ }
    | "->" { GOTO }
    | "npc" { NPC }
    | "item" { ITEM }
    | '{' { LPAREN }
    | '}' { RPAREN }
    | '{' { LBRACE }
    | '}' { RBRACE }
    | '[' { LBRACK }
    | ']' { RBRACK }
    | ',' { COMMA }
    | '+' { PLUS } (* operators*)
    | '-' { MINUS }
    | '*' { TIMES }
    | '/' { DIVIDE }
    | "==" { EQ }
    | "~==" { STREQ }
    | "!=" { NEQ }
    | '<' { LT }
    | "<=" { LEQ }
    | '>' { GT }
    | ">=" { GEQ }
    | '=' { ASSIGN }
    | ';' { SEMI }
9.1.2 Parser (parser.mly)

/* Authors: All */

{% open Ast %}

%token SEMI LPAREN RPAREN LBRACE RBRACE LBRACK RBRACK COMMA
%token FUNC ROOM ADJ GOTO ITEM NPC START END NEG
%token ASSIGN EQ STREQ NEQ LT LEQ GT GEQ AND OR NOT ACCESS
%token PLUS MINUS TIMES DIVIDE
%token IF ELSE WHILE RETURN
%token INT STRING VOID BOOLEAN
%token <int> INT_LITERAL
%token <string> STRING_LITERAL
%token <bool> BOOL_LITERAL
%token <string> ID
%token EOF

%right ASSIGN
%left OR
%left AND
%left EQ NEQ STREQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE
%right NOT
%left ACCESS

%start program
%type <Ast.program> program

%}{ close Ast %}
program:
  /* rooms, npcs, items */
  rdef rdecl_list adecl_list start ndef ndecl_list idecl_list vdecl_list
  predicate_list fdecl_list EOF
  { ($1, $2, $3, $4, $5, $6, $7, $8, $9, list.rev $10, list.rev $11) }
  | /* rooms, npcs, litems */
  rdef rdecl_list adecl_list start ndef ndecl_list vdecl_list
  predicate_list fdecl_list EOF
  { ($1, $2, $3, $4, $5, $6, [], [], $7, list.rev $8, list.rev $9) }
  | /* rooms, !npcs, items */
  rdef rdecl_list adecl_list start idecl_list vdecl_list
  predicate_list fdecl_list EOF
  { ($1, $2, $3, $4, [], [], $5, $6, $7, list.rev $8, list.rev $9) }
  | /* rooms, !npcs, !items */
  rdef rdecl_list adecl_list start vdecl_list predicate_list fdecl_list EOF
  { ($1, $2, $3, [], [], $5, $6, $7, list.rev $6, list.rev $7) }
  | /* !rooms, npcs, items */
  rdef ndecl_list idecl_list vdecl_list predicate_list fdecl_list EOF
  { ([], [], [], "null", $1, $2, $3, $4, $5, list.rev $6, list.rev $7) }
  | /* !rooms, !npcs, !items */
  rdef ndecl_list vdecl_list predicate_list fdecl_list EOF
  { ([], [], [], "null", $1, $2, [], [], $3, list.rev $4, list.rev $5) }
  | /* !rooms, !npcs, !items */
  idef idecl_list vdecl_list predicate_list fdecl_list EOF
  { ([], [], [], "null", [], [], $1, $2, $3, list.rev $4, list.rev $5) }
  | /* !rooms, !npcs, !items */
  vdecl_list predicate_list fdecl_list EOF
  { ([], [], [], "null", [], [], [], [], $1, list.rev $2, list.rev $3) }
  
data_type:
  INT
  | STRING { String }
  | VOID { Void }
  | BOOLEAN { Boolean }
  
pred_stmt:
  expr LBRACE vdecl_list stmt_list RBRACE
  { }
  | pred = $1;
  | locals = list.rev $3;
  | body = list.rev $4;
  | }
  
predicate_list:
  /* nothing */
  { [] }
  | predicate_list pred_stmt
  { $2 :: $1 }
  
fdecl_list:
  /* nothing */
  { [] }
  | fdecl_list fdecl
  { $2 :: $1 }
  
fdecl:
  FUNC data_type ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
  { }
  | returntype = $2;
  | fname = $3;
  | formals = $5;
locals = list.rev $8;
body = list.rev $9
}

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

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

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

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

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

vdecl:
  data_type LBRACK expr RBRACK ID SEMI
  | data_type ID SEMI
  | data_type ID ASSIGN expr SEMI
  { Arraydecl($1, $3, $5) }
  { Var($1, $2) }
  { VarInit($1, $2, $4) }

rdef:
  ROOM LBRACE vdecl_list RBRACE
  { $3 }

rdecl_list:
  rdecl rdecl
  { [$1; $2] }
  { $2 :: $1 }

rdecl:
  ROOM ID LBRACE stmt_list RBRACE
  { }
  { rname = $2;
    rbody = list.rev $4
  }

start:
  START LBRACE ID RBRACE
  { $3 }

adecn_list:
  adecl
  | adecl_list adecl
  { [$1] }
  { $2 :: $1 }

adecn:
  adj_list SEMI
  { List.rev $1 }

adj_list:
  ID ADJ ID
  { [$1; $3] }

ndef:
  NPC LBRACE vdecl_list RBRACE
  { $3 }

ndecl_list:
/* nothing */
| ndecl_list ndecl
   { [] }   { $2 :: $1 }

ndecl:
  NPC ID LBRACE stmt_list RBRACE
  { { nname = $2;
      nboby = list.rev $4
    } }

idef:
  ITEM LBRACE vdecl_list RBRACE
  { $3 }  $1

idecl_list:
  /* nothing */
  { [] }   { $2 :: $1 }

idecl:
  ITEM ID LBRACE stmt_list RBRACE
  { { iname = $2;
    iboby = list.rev $4
    } }

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

stmt:
  expr SEMI
  | RETURN expr SEMI
  | LBRACE stmt_list RBRACE
  | IF LPAREN expr RPAREN stmt ELSE stmt
  | WHILE LPAREN expr RPAREN stmt
  | GOTO ID
  { Expr($1) }   { Return($2) }   { Block(list.rev $2) }   { If($3, $5, $7) }   { While($3, $5) }   { Goto($2) }

/*
int_opt:
  INT_LITERAL
  { $1 }
*/

expr:
  INT_LITERAL
  | NEG INT_LITERAL
  | STRING_LITERAL
  | END
  | BOOL_LITERAL
  | ID
  | expr PLUS expr
  | expr MINUS expr
  | expr TIMES expr
  | expr DIVIDE expr
  | expr EQ expr
  | expr STREQ expr
  | expr NEQ expr
  | expr LT expr
  | expr LEQ expr
  | expr GT expr
  | expr GEQ expr
  | expr AND expr
  | expr OR expr
  { IntLiteral($1) }   { NegIntLiteral($2) }   { StrLiteral($1) }   { End }   { BoolLiteral($1) }   { Id($1) }   { Binop($1, Add, $3) }   { Binop($1, Sub, $3) }   { Binop($1, Mult, $3) }   { Binop($1, Div, $3) }   { Binop($1, Equal, $3) }   { Binop($1, StrEqual, $3) }   { Binop($1, Neq, $3) }   { Binop($1, Less, $3) }   { Binop($1, Leq, $3) }   { Binop($1, Greater, $3) }   { Binop($1, Geq, $3) }   { Binop($1, And, $3) }   { Binop($1, Or, $3) }
9.1.3 AST (ast.mli)

(* Authors: All *)

type op = Add | Sub | Mult | Div | Equal | StrEqual | Neq | Less | Leq | Greater | Geq | And | Or | Not

type variable_type =
  Int
  | String
  | Void
  | Array of variable_type * int
  | Boolean

type expr =
  IntLiteral of int
  | NegIntLiteral of int
  | StrLiteral of string
  | BoolLiteral of bool
  | Id of string
  | Assign of string * expr
  | ArrayAssign of string * expr * expr
  | ArrayAccess of string * expr
  | Binop of expr * op * expr
  | Boolneg of op * expr
  | Call of string * expr list
  | Access of string * string
  | End

type var_decl =
  Array_decl of variable_type * expr * string
  | Var of variable_type * string
  | VarInit of variable_type * string * expr

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

type room_def = var_decl list

type room_decl =
  {
    rname: string;
    rbody: stmt list;
  }
type start = string

type adj_decl = string list

type pred_stmt =
    {
        pred: expr;
        locals: var_decl list;
        body: stmt list;
    }

type func_decl =
    {
        returntype: variable_type;
        fname : string;
        formals : var_decl list;
        locals: var_decl list;
        body : stmt list;
    }

type npc_def = var_decl list

type npc_decl =
    {
        nname: string;
        nbody: stmt list;
    }

type item_def = var_decl list

type item_decl =
    {
        iname: string;
        ibody: stmt list;
    }

type basic_program = func_decl list

type simple_program = room_decl list *
    func_decl list

type room_program = room_def *
    room_decl list *
    func_decl list

type program = room_def *
    room_decl list *
    adj_decl list *
    start *
    npc_def *
    npc_decl list *
    item_def *
    item_decl list *
    var_decl list *
    pred_stmt list *
    func_decl list
9.1.4 Semantic Checker (semantic_checker.ml)

(* Authors: Iris, Maria *)

open Ast

(* environment *)

type symbol_table = {
  parent : symbol_table option;
  mutable variables : var_decl list;
}

type translation_environment = {
  scope : symbol_table;
  mutable return_type: variable_type;
  mutable current_func: func_decl option;
  mutable functions : func_decl list;
  mutable room_def: var_decl list;
  mutable rooms: room_decl list;
  mutable npc_def: var_decl list;
  mutablenpcs: npc_decl list;
  mutable item_def: var_decl list;
  mutable items: item_decl list;
  mutable pred_stmts : pred_stmt list;
}

(* helper functions *)

let check_type_not_void (v : Ast.variable_type) = match v with
  Ast.Int -> Ast.Int
| Ast.String -> Ast.String
| Ast.Boolean -> Ast.Boolean
| Ast.Array(v, i) -> Ast.Array(v, i)
| _ -> raise (Failure ("Invalid variable type used"))
(* vars can't be declared as "void" *)

let get_var_type_name var_decl =
  begin match var_decl with
    Array_decl(t, _, s) -> (t, s)
    | Var(t, s) -> (t, s)
    | VarInit(t, s, _) -> (t, s)
  end

let var_is_array var_decl =
  begin match var_decl with
    Array_decl(_, _, _) -> true
    | Var(_, _) -> false
    | VarInit(_, _, _) -> false
  end

let expr_is_strlit expr =
  begin match expr with
    StrLiteral(_) -> true
    | _ -> false
  end

(* type enforcement functions *)
let require_integers tlist str =
    let _ = List.map(  
        fun t -> match t with  
            Int -> true  
            _ -> raise (Failure(str))  
        ) tlist in  
    true

let require_strings tlist str =
    let _ = List.map(  
        fun t -> match t with  
            String -> true  
            _ -> raise (Failure(str))  
        ) tlist in  
    true

let require_voids tlist str =
    let _ = List.map(  
        fun t -> match t with  
            Void -> true  
            _ -> raise (Failure(str))  
        ) tlist in  
    true

let require_bools tlist str =
    let _ = List.map(  
        fun t -> match t with  
            Boolean -> true  
            _ -> raise (Failure(str))  
        ) tlist in  
    true

let require_eq tlist str =
    let typ = List.hd tlist in  
    let _ = List.map(  
        fun t -> if typ <> t then raise (Failure(str))  
        ) tlist in  
    true

(* find functions *)
let rec find_variable (scope : symbol_table) name =
    try  
        (* do match with the different types of variables in the List.find  
           * function *)
        List.find ( fun var_decl ->
            begin match var_decl with
                Array_decl(_, _, s) -> s = name  
                | Var(_, s) -> s = name  
                | VarInit(_, s, _) -> s = name
                end ) scope.variables
        with Not_found ->
            match scope.parent with
            | Some(parent) -> find_variable parent name
            | _ -> raise Not_found

let get_var_if_exists (scope : symbol_table) name =
    let var_decl = (try find_variable scope name with
        Not_found -> raise (Failure ("undeclared identifier" ^ name))) in var_decl
let find_room (env: translation_environment) (name) =
  try
    List.find (fun room_decl -> room_decl.rname = name) env.rooms
  with Not_found -> raise Not_found

let find_room_field (env: translation_environment) fieldName =
  let field_decl = (try (List.find ( fun var_decl -> begin match var_decl with
    Var(t, s) -> s = fieldName
    | _ -> raise (Failure "should never reach here")
    end ) env.room_def )
    with
      Not_found -> raise(Failure "room field referenced does not exist.
  )
    in let (typ, n) = get_var_type_name field_decl in
    (typ, n)

let find_npc (env: translation_environment) (name) =
  try
    List.find (fun npc_decl -> npc_decl.rname = name) env.npcs
  with Not_found -> raise Not_found

let find_npc_field (env: translation_environment) fieldName =
  let field_decl = (try (List.find ( fun var_decl -> begin match var_decl with
    Var(t, s) -> s = fieldName
    | _ -> raise (Failure "should never reach here")
    end ) env.npc_def )
    with
      Not_found -> raise(Failure "npc field referenced does not exist.
  )
    in let (typ, n) = get_var_type_name field_decl in
    (typ, n)

let find_item (env: translation_environment) (name) =
  try
    List.find (fun item_decl -> item_decl.iname = name) env.items
  with Not_found -> raise Not_found

let find_item_field (env: translation_environment) fieldName =
  let field_decl = (try (List.find ( fun var_decl -> begin match var_decl with
    Var(t, s) -> s = fieldName
    | _ -> raise (Failure "should never reach here")
    end ) env.item_def )
    with
      Not_found -> raise(Failure "item field referenced does not exist.
  )
    in let (typ, n) = get_var_type_name field_decl in
    (typ, n)

(* check that op matches with types of t1, t2 and return type of result *)
let get_binop_type op t1 t2 =
  begin match op with
    (Add | Sub | Mult | Div) ->
      let _ = require_integers [t1;t2] "Types to arithmetic operators +, -, *, /
      must both be Int" in
      Int
    | (Equal | Neq) ->
      let _ =
        (try require_integers[t1;t2] ""
          with _ -> try require_bools[t1;t2] ""
        )
    | _ -> raise(Failure "unknown operation")
  end

let get_binop_name op t1 t2 =
  begin match op with
    (Add | Sub | Mult | Div) ->
      let _ = require_integers [t1;t2] "Types to arithmetic operators +, -, *, /
      must both be Int" in
      get_id (Int, t1, t2)
    | (Equal | Neq) ->
      let _ =
        (try require_integers[t1;t2] ""
          with _ -> try require_bools[t1;t2] ""
        )
    | _ -> raise(Failure "unknown operation")
  end

let get_binop_desc op t1 t2 =
  begin match op with
    (Add | Sub | Mult | Div) ->
      let _ = require_integers [t1;t2] "Types to arithmetic operators +, -, *, /
      must both be Int" in
      get_id (Int, t1, t2)
    | (Equal | Neq) ->
      let _ =
        (try require_integers[t1;t2] ""
          with _ -> try require_bools[t1;t2] ""
        )
    | _ -> raise(Failure "unknown operation")
  end
let vdecl = (try
  find_variable env.scope vname
with Not_found ->
    let _ = (try find_room env vname with
             Not_found -> raise (Failure ("undeclared identifier " ^
                          vname))) in
    Var(Ast.Void, vname)) in
let (typ, vname) = get_var_type_name vdecl
in (Ast.Id(vname), typ)
| Ast.Binop(e1, op, e2) ->
  let (e1, t1) = check_expr env e1
  and (e2, t2) = check_expr env e2 in
  (Ast.Binop(e1, op, e2), get_binop_type op t1 t2)
| Ast.Assign(name, expr) ->
  let vdecl = get_var_if_exists env.scope name in
  let (vtyp, name) = get_var_type_name vdecl in
  let (expr, etyp) = check_expr env expr in
  if not (var_is_array vdecl) then
    let _ = require_eq [vtyp;etyp] "Type mismatch in assignment statement"
  in (Ast.Assign(name, expr), vtyp)
  else raise (Failure "Left hand side of assignment statement must
               be a non-array variable")
| Ast.ArrayAssign(name, expr1, expr2) ->
  let vdecl = get_var_if_exists env.scope name in
  let (typ, name) = get_var_type_name vdecl in
  let (pos, postyp) = check_expr env expr1 in
  let (expr, exprtyp) = check_expr env expr2 in
  let _ = require_integers [postyp] "Positional array access specifier must be
an Integer" in
  if var_is_array vdecl then
    let _ = require_eq [typ;exprtyp] "Right hand side of assignment statement does
not match type of array" in
(Ast.ArrayAssign(name, pos, expr), typ)
else raise (Failure "Left hand side of array assignment must be an array")
| Ast.ArrayAccess(name, expr) ->
  let vdecl = get_var_if_exists env.scope name in
  let (typ, name) = get_var_type_name vdecl in
  let (pos, postyp) = check_expr env expr in
  let _ = require_integers [postyp] "Positional array access specifier must be an Integer" in
  if var_is_array vdecl then (Ast.ArrayAccess(name, expr), typ)
else raise (Failure "Array access must be used on an array type")
| Ast.BoolNeg(op, expr) ->
  let (expr, typ) = check_expr env expr in
  let _ = require_bools [typ] "Type to unary boolean NOT operator must be boolean" in
  (Ast.BoolNeg(op, expr), typ)
| Ast.Call(fname, expr_list) ->
  if fname = "arrLen" && List.length expr_list = 1 then
    let arr_name =
      let e = List.hd expr_list in
      begin match e with
        Ast.Id(vname) -> vname
      | _ -> raise (Failure("arrLen expects an array argument"))
    end in
    let arr_decl = get_var_if_exists env.scope arr_name in
    if var_is_array arr_decl then
      (Ast.Call(fname, expr_list), Ast.Int)
    else raise (Failure "arrLen expects an argument")
  else if fname = "getInputFromOptions" && List.length expr_list >= 1 then
    let _ = List.map(
      fun e -> if not (expr_is_strlit e) then raise (Failure("getInputFromOptions expects one or more string arguments"))
    ) expr_list in
    (Ast.Call(fname, expr_list), Ast.Void)
  else if fname = "getInputAdjacentRooms" && List.length expr_list = 1 then
    let rname =
      begin match List.hd expr_list with
        Ast.Id(r) -> r
      | _ -> raise (Failure("getInputAdjacentRooms expects a room argument"))
    end in
    (Ast.Call(fname, expr_list), Ast.Void)
else
  let fdecl = (try find_function_with_exprs env fname expr_list
    with Not_found -> begin match env.current_func with
      Some(current_func) ->
        if (current_func.fname = fname &&
          check_matching_args env current_func.formals
          expr_list) then current_func
        else raise (Failure("Function " ^ fname ^ " does not exist with the given parameters."))
      end
    | Not_found -> raise (Failure("Function " ^ fname ^ " does not exist with the given parameters."))
    end)
let check_matching_decls in
    Ast.Call(fname, expr_list), typ)
| Ast.End -> (Ast.End, Ast.Int) (* This type is meaningless *)
| Ast.Access(name, field) ->
    try let _ = find_room env name in
    let (ftyp, fname) = find_room_field env field in
    (Ast.Access(name, field), ftyp)
    with Not_found ->
    try let _ = find_npc env name in
    let (ftyp, fname) = find_npc_field env field in
    (Ast.Access(name, field), ftyp)
    with Not_found ->
    try let _ = find_item env name in
    let (ftyp, fname) = find_item_field env field in
    (Ast.Access(name, field), ftyp)
    with Not_found ->
    raise(Failure("Trying to access field " ^ field ^ ", which does not
exist for that structure.
(* check formal arg list with expr list of called function *)
and check_matching_args_helper (env: translation_environment) ref_vars target_exprs =
    let result = true in
    let _ = (try List.map2 (fun r t -> let (rtyp, rname) = get_var_type_name r in
            let (txpr, ttyp) = check_expr env t in
            try require_eq [ttyp;rtyp] "";
            with _ -> raise Not_found) ref_vars target_exprs
        with Invalid_argument(_) -> raise Not_found) in result

and check_matching_args (env: translation_environment) ref_vars target_exprs =
    let result = (try check_matching_args_helper env ref_vars target_exprs with
    Not_found -> false) in result

and find_function_with_exprs (env: translation_environment) name expr_list =
    try
        List.find( fun func_decl -> func_decl.fname = name &&
            check_matching_args env func_decl.formals expr_list) env.functions
        with Not_found -> raise Not_found
    end

let check_matching_decls_helper (env: translation_environment) ref_vars target_decls =
    let result = true in
    let _ = (try List.map2 (fun r t -> let (rtyp, rname) = get_var_type_name r in
            let (ttyp, tname) = get_var_type_name t in
            try require_eq [ttyp;rtyp] "";
            with _ -> raise Not_found) ref_vars target_decls
        with Invalid_argument(_) -> raise Not_found) in result

let check_matching_decls (env: translation_environment) ref_vars target_decls =
    let result = (try check_matching_decls_helper env ref_vars target_decls with
    Not_found -> false) in result

let find_function_with_DECLS (env: translation_environment) name decl_list =
    try
        List.find( fun func_decl -> func_decl.fname = name &&
            check_matching_decls env func_decl.formals decl_list) env.functions
        with Not_found -> raise Not_found
    end
(* Stmt checking*)
let rec check_stmt env = function
  Block(stmt_list) -> Block(check_stmt env stmt_list)
  | Expr(expr) -> let (expr, _) = check_expr env expr in Expr(expr)
  | Return(expr) -> let (expr, typ) = check_expr env expr in
                      let _ = require_eq [typ;env.return_type] "Return type of expression does not match type of function" in
                      Return(expr)
  | If(expr, stmt1, stmt2) ->
                      let (expr, typ) = check_expr env expr in
                      let _ = require_bools [typ] "Expression in if statement conditional must be of type Boolean" in
                      If(expr, check_stmt env stmt1, check_stmt env stmt2)
  | While(expr, stmt) ->
                      let (expr, typ) = check_expr env expr in
                      let _ = require_bools [typ] "Expression in while statement conditional must be of type Boolean" in
                      While(expr, check_stmt env stmt)
  | Goto(rname) ->
                      let rdecl = try find_room env rname with
                      Not_found -> raise(Failure "Goto parameter name not a valid room.")
                      in Goto(rdecl.rname)

and check_stmts (env: translation_environment) stmt_list =
  let stmt_list = List.map (fun s -> check_stmt env s) stmt_list in
  stmt_list

(* Variable checking, both global and local *)
let check_var_decl (env: translation_environment) vdecl =
  let (typ, vname) = get_var_type_name vdecl in
  try let _ = find_variable env.scope vname in raise(Failure ("Variable with name " ^
    vname ^
    " exists.")")
  with Not_found ->
    (* add this var to the variables list of this environment *)
    (* also check that the expr type matches up with the type of the var * *)
    (* check that type is valid*)
    match vdecl with
    Array_decl (typ, expr, name) ->
      let (expr, exprtyp) = check_expr env expr in
      let _ = require_integers [exprtyp] "Array size must be integer" in
      let typ = check_type_not_void typ in
       (env.scope.variables <- Array_decl (typ,expr,name)::env.scope.variables;
        Array_decl typ,expr,name)
    Var (typ, name) -> let typ = check_type_not_void typ in
      env.scope.variables <- Var(typ, name)::env.scope.variables; Var(typ, name)
    VarInit (typ, name, expr) -> let typ = check_type_not_void typ in
      let (expr, exprtyp) = check_expr env expr in
      let _ = require_eq [exprtyp;typ] "Type mismatch in variable initialization" in
       (env.scope.variables <- VarInit (exprtyp,name,expr)::env.scope.variables;
        VarInit (exprtyp, name,expr))

let check_var_decls (env: translation_environment) var_decls =
  let var_decls = List.map(fun vdecl -> check_var_decl env vdecl) var_decls in
  var_decls

let rec check_stmts env stmt_list =
  let stmt_list = List.map (fun s -> check_stmt env s) stmt_list in
  stmt_list

(* Variable checking, both global and local *)
let check_var_decl (env: translation_environment) vdecl =
  let (typ, vname) = get_var_type_name vdecl in
  try let _ = find_variable env.scope vname in raise(Failure ("Variable with name " ^
    vname ^
    " exists.")")
  with Not_found ->
    (* add this var to the variables list of this environment *)
    (* also check that the expr type matches up with the type of the var * *)
    (* check that type is valid*)
    match vdecl with
    Array_decl (typ, expr, name) ->
      let (expr, exprtyp) = check_expr env expr in
      let _ = require_integers [exprtyp] "Array size must be integer" in
      let typ = check_type_not_void typ in
       (env.scope.variables <- Array_decl (typ,expr,name)::env.scope.variables;
        Array_decl typ,expr,name)
    Var (typ, name) -> let typ = check_type_not_void typ in
      env.scope.variables <- Var(typ, name)::env.scope.variables; Var(typ, name)
    VarInit (typ, name, expr) -> let typ = check_type_not_void typ in
      let (expr, exprtyp) = check_expr env expr in
      let _ = require_eq [exprtyp;typ] "Type mismatch in variable initialization" in
       (env.scope.variables <- VarInit (exprtyp,name,expr)::env.scope.variables;
        VarInit (exprtyp, name,expr))

let check_var_decls (env: translation_environment) var_decls =
  let var_decls = List.map(fun vdecl -> check_var_decl env vdecl) var_decls in
  var_decls

(* Function checking*)

let check_funcDecl (env: translation_environment) funcDecl =
  try
    let _ = find_function_with_decls env funcDecl.fname funcDecl.formals in
    raise(Failure("Function with name " ^ funcDecl.fname ^ " and given
argument types exists"))
  with Not_found ->
    let scope' = { parent = Some(env.scope); variables = [] } in
    let env' = { env with scope = scope'; return_type =
      funcDecl.return_type; current_func = Some(funcDecl) } in
    let fformals = check_var_decls env' funcDecl.formals in
    let flocals = check_var_decls env' funcDecl.locals in
    let fbody = funcDecl.body in
    let fReturnType = funcDecl.return_type in
    let new_funcDecl = { funcDecl with body = fbody; locals = flocals;
      formals = fformals; returnType = fReturnType; } in
    env.functions <- new_funcDecl::env.functions ; new_funcDecl

let check_func_decls env func_decls =
  let func_decls = List.map (fun f -> check_funcDecl env f) func_decls in
  func_decls

(* Room checking*)

let process_room_field (field: Ast.var_decl) (env: translation_environment) = match field with
  Ast.Var(typ, name) ->
    let t = check_type_not_void_typ in
    if (List.exists (fun var_decl -> begin match var_decl with
      Var(_, s) -> s = name
      | _ -> raise (Failure "should never reach here")
        end ) env.room_def )
    then
      raise (Failure "room fields names cannot repeat.")
    else
      env.room_def <- Ast.Var(t, name):: env.room_def; (* side effect add room
field to room_table *)
    Ast.Var(t, name) (*return this*)
    | _ -> raise (Failure "room field not correct format. declare a type and name.")

let process_room_decl_body (env: translation_environment) (rfa: Ast.stmt) = begin match rfa with
  Ast.Expr(roomAssign) -> begin match roomAssign with (* check that the expr is in the
form of an assign*)
    Ast.AssIGN(fieldname, expr) ->
      let rdecl =
        (try list.find (fun rdecl -> begin match rdecl with
          Array_decl(_, _, s) -> "0" = field
          | Var(t, s) -> s = field
          | VarInit(_, s, _) -> "0" = field end) env.room_def
        with
          Not_found -> raise (Failure "field name in room decl does not
exist.") ) in
      let (room_decl1_typ, _) = get_var_type_name rdecl in
      (*CHECKING FOR ROOM DECL BODY EXPR RETURN TYPE HERE*)
      let (_, typ) = check_expr env expr in
      let _ = require_eq [typ;room_decl1_typ] "room decl body does not match field
      type" in
      rfa (*return this*)
      | _ -> raise (Failure "room assignment not correct format.")
    end
  end

69
let check_room_decl (env: translation_environment) (room: Ast.room_decl) =
    let name = room.rname in
    let body = room.rbody in (* body is a list of stmts*)
    try let _ = find_room env name in raise(Failure ("Room with name " ^ name ^ " already exists.")
    with Not_found ->
    let checked_body = List.map ( fun unchecked -> process_room_decl_body env unchecked) body in
    (* check that number of fields in room_def match with number of fields in room decl*)
    let num_stmts = List.length(checked_body) in
    if (num_stmts <> List.length(env.room_def)) then
        raise (Failure "number of room decl fields do not match definition.")
    else
        (* add the room_decls to the scope*)
        let checked_room_decl = { rname = name; rbody = checked_body } in
        env.rooms <- checked_room_decl::env.rooms;
        checked_room_decl (* return this *)

let check_room_decls (env: translation_environment) rooms =
    let checked_room_decls = List.map (fun unchecked -> check_room_decl env unchecked) rooms in
    checked_room_decls

let check_room_def (env: translation_environment) (r: Ast.room_def) =
    try
        let checked_fields = List.map ( fun room_field -> process_room_field room_field env) r in
        checked_fields
        with
            | _ -> raise (Failure "room defs didn’t check out")
    (* NPC checking*)

let process_npc_field (field: Ast.var_decl) (env: translation_environment) = match field with
    Ast_Var(typ, name) ->
        let t = check_type_not_void typ in
        if (List.exists ( fun var_decl -> begin match var_decl with
            Var(_, s) -> s = name
            |_ -> raise (Failure "should never reach here")
          end ) env.npc_def )
          then raise (Failure "npc fields names cannot repeat.")
        else
            env.npc_def <- Ast.Var(t, name):: env.npc_def; (* side effect add room field to room_table *)
            Ast_Var(t, name) (*return this*)
            | _ -> raise (Failure "npc field not correct format. declare a type and name.")

let process_npc_decl_body (env: translation_environment) (nfa: Ast.stmt) = begin match nfa with
    Ast.Expr(npcAssign) -> begin match npcAssign with (* check that the expr is in the form of an assign*)
        Ast.Assign(fieldname, expr) ->
            let ndecl =
(try list.find(fun ndecl -> begin match ndecl with
  Array_decl(_, _, s) -> "\theta" = fieldname
  | Var(t, s) -> s = fieldname
  | VarInit(_, s, _) -> "\theta" = fieldname end) env.npc_def
with
Not_found-> raise (Failure "field name in npc decl does not exist.") in
let (npc_decl_typ, _) = get_var_type_name ndecl in
let (_, typ) = check_expr env expr in
let _ = require_eq [typ; npc_decl_typ] "npc decl body does not match field
type" in
  nfa (*return this*)
  | _ -> raise (Failure "npc assignment not correct format.") end
| _ -> raise (Failure "npc assignment not correct format.") end

let check_npc_decl (env: translation_environment) (npc: Ast.npc_decl) =
  let name = npc.nname in
  let body = npc.nbody in (* body is a list of stmts*)
  try let _ = find_npc env name in raise(Failure ("NPC with name " ^ name ^ " already
  exists.") )
  with Not_found ->
    let checked_body = List.map ( fun unchecked -> process_npc_decl_body env unchecked)
    body in
    (* check that number of fields in room_def match with number of fields in room
    decl*)
    let num_stmts = List.length(checked_body) in
    if (num_stmts <> List.length(env.npc_def)) then
      raise (Failure "number of npc decl fields do not match definition.")
    else
      (* add the room defs to the scope*)
      let checked_npc_decl = { nname = name; nbody = checked_body } in
      env.npcs <- checked_npc_decl::env.npcs;
      checked_npc_decl (* return this *)

let check_npc_decls (env: translation_environment) npcs =
  let checked_npc_decls = List.map ( fun unchecked -> check_npc_decl env unchecked)
  npcs in
  checked_npc_decls

let check_npc_def (env: translation_environment) (n: Ast.npc_def) =
  let checked_fields = List.map ( fun npc_field -> process_npc_field npc_field env) n
  in
  checked_fields

(* Item checking*)
let process_item_field (field: Ast.var_decl) (env: translation_environment) = match field
with
  Ast.Var(typ, name) ->
    let t = check_type_not_void typ in
    if (List.exists ( fun var_decl -> begin match var_decl with
      Var(_, s) -> s = name
      | _ -> raise (Failure "should never reach here")
    end ) env.item_def)
    then
      raise (Failure "item fields names cannot repeat.")
    else
      env.item_def <- Ast.Var(t, name):: env.item_def; (* side effect add item
field to item_table *)
    Ast.Var(t, name) (*return this*)
    | _ -> raise (Failure "item field not correct format. declare a type and name.")

let process_item_decl_body (env: translation_environment) (ifa: Ast.stmt) = begin match ifa with
    Ast.Expr(itemAssign) -> begin match itemAssign with (* check that the expr is in the form of an assign*)
        Ast.Assign(fieldname, expr) ->
            let idecl =
                (try List.find(fun idecl -> begin match idecl with
                        Array_decl(_, _, s) -> "\0" = fieldname
                        Var(t, s) -> s = fieldname
                        VarInit(_, s, _) -> "\0" = fieldname end) env.item_def
                    with Not_found -> raise (Failure "field name in npc decl does not exist.")
                ) in
            let (item_decl_typ, _) = get_var_type_name idecl in
            let (_, typ) = check_expr env expr in
            let _ = require_eq[typ;item_decl_typ] "item decl body does not match field type"
        in
        ifa (*return this*)
        | _ -> raise (Failure "item assignment not correct format.") end
    | _ -> raise (Failure "item assignment not correct format.") end

let check_item_decl (env: translation_environment) (item: Ast.item_decl) =
    let name = item.name in
    let body = item.body in (* body is a list of stmts*)
    try let _ = find_item env name in raise(Failure ("Item with name " ^ name ^ " already exists.")
        with Not_found ->
            let checked_body = List.map ( fun unchecked -> process_item_decl_body env unchecked)
        body in
        (* check that number of fields in room_def match with number of fields in room decl*)
        let num_stmts = List.length(checked_body) in
        if (num_stmts <> List.length(env.item_def)) then
            raise (Failure "number of item decl fields do not match definition.")
        else
            (* add the room_decls to the scope*)
            let checked_item_decl = { iname = name; ibody = checked_body } in
            env.items <- checked_item_decl::env.items;
            checked_item_decl (* return this *)
    end

let check_item_decls (env: translation_environment) items =
    let checked_item_decls = List.map ( fun unchecked -> check_item_decl env unchecked)
        items in
    checked_item_decls

let check_item_def (env: translation_environment) (i: Ast.item_def) =
    let checked_fields = List.map ( fun item_field -> process_item_field item_field env)
        i in
    checked_fields

(* Predicate checking *)
let check_pred_stmt (env: translation_environment) pstm =
    (* check that the expr is a boolean expr, check all var decls, check all
    * stmts in body *)
let scope' = { parent = Some(env.scope); variables = [] } in
let env' = { env with scope = scope'; functions =
env.functions; room_def = env.room_def; pred_stmts =
env.pred_stmts; rooms = env.rooms } in
let (checked_pred, typ) = check_expr env stmt.pred in
let _ = require_bools [typ] "Expression in predicate statement conditional must be of type Boolean" in
let checked_locals = check_var_decls env' stmt.locals in
let checked_body = check_stmts env' stmt.body in
let new_stmt = { pred = checked_pred; locals = checked_locals; body =
checked_body; } in
env.pred_stmts <- new_stmt::env.pred_stmts ; new_stmt

let check_pred_stmts (env: translation_environment) stmts =
let new_stmts = list.map (fun s -> check_pred_stmt env s) stmts in
new_stmts

(* Adjacency checking *)
let find_adjacency (env: translation_environment) adj =
if (List.exists (fun rdecl -> rdecl.rname = (List.nth adj 0)) env.rooms &&
List.exists (fun rdecl -> rdecl.rname = (List.nth adj 1)) env.rooms) then
adj
else raise (Failure "One of rooms in adjacency list not declared")

let check_adj_decls (env: translation_environment) decls =
try
let checked_adj = list.map (fun decl -> find_adjacency env decl) decls in
checked_adj
with
| _ -> raise (Failure "adjacencies didn't check out")

(* Entrance point that transforms Ast into semantically correct Ast *)
let check_program (p : Ast.program) =
let var_int = { ftype = Void; fname = "print"; formals =
[Var(Ast.Int, "arg")]; locals = []; body = [] } in
let var_bool = { ftype = Void; fname = "print"; formals =
[Var(Ast.Bool, "arg")]; locals = []; body = [] } in
let var_str = { ftype = Void; fname = "print"; formals =
[Var(Ast.String, "arg")]; locals = []; body = [] } in
let print_funcs = [print_int; print_bool; print_str] in
(* adding name type String as default field in room_def*)
let name_field = Ast.Var(String, "name") in
(* adding currentRoom as a global variable*)
let current_room = { rname = "currentRoom"; rbody = [] } in
let input = Var(Ast.String, "input") in
let dummy_room = { rname = "input"; rbody = [] } in
let dummy_npc = { rname = "input"; nbodies = [] } in
let dummy_item = { rname = "input"; ibody = [] } in
let symbol_table = { parent = None; variables = [input]; } in
let env = { scope = symbol_table; return_type =
Ast.Int; functions = print_funcs; room_def = [name_field]; rooms = [current_room;
dummy_room]; npc_def = []; npcs = [dummy_npc];
item_def = []; items = [dummy_item]; pred_stmts = []; current_func = None } in
let room_def, room_decls, decls, start, npc_def, npc_decls, item_def,
item_decls, var_decls, pred_stmts, funcs = p in
let checked_room_def = check_room_def env room_def in
let checked_room_decls = check_room_decls env room_decls in
let checked_npc_def = check_npc_def env npc_def in
let checked_npc_decls = check_npc_decls env npc_decls in
9.1.5 Java Builder (java_builder.ml)

(* Authors: Brian, Greg *)

open Jast
open Ast
open Printf

(* http://langref.org/fantom+ocaml+erlang/files/reading/read-into-string *)
let load_file f =
  let ic = open_in f in
  let n = in_channel_length ic in
  let s = Bytes.create n in
  really_input ic s @ n;
  close_in ic;
  (s)

let build_main (preds, rooms, adjacencies, start, npcs, items) =
  { predicates = preds; rdecls = rooms; adecls = adjacencies; start = start; ndecls = npcs; idecls = items;}

let build_driver (vars, preds, functions, rooms, adjacencies, start, npcs, items) =
  let main = build_main (preds, rooms, adjacencies, start, npcs, items) in
  let default_funcs = load_file("java_lib/driver_functions.txt") in
  (vars, main, functions, default_funcs)

let rearrange (program) =
  let (room_def, room_decl_list, adj_decl_list, start, npc_def, npc_decl_list, item_def, item_decl_list, vdecl_list, predicate_list, func_decl_list) =
    program in
  let driver = build_driver (vdecl_list, predicate_list, func_decl_list, room_decl_list, adj_decl_list, start, npc_decl_list, item_decl_list) in
  (driver, room_def, npc_def, item_def)

9.1.6 JAST (jast.ml)

(* Authors: Greg, Brian *)
open Ast

type main_method =
{
    predicates: predStmt list;
    rdecls: room_decl list;
    adecls: adj_decl list;
    start: start;
    ndecls: npc_decl list;
    idecls: item_decl list;
}

type other_classes = room_def * item_def * npc_def

type driver_class = var_decl list * main_method * func_decl list

type program = driver_class * other_classes

9.1.7 Code Generator (code_gen.ml)

(* Authors: Brian, Greg *)

open Printf
open Jast
open Ast

let driver_file = "Driver.java"
let room_file = "Room.java"
let npc_file = "Npc.java"
let item_file = "Item.java"

let rec data_type = function
  | String -> "String"
  | Int -> "int"
  | Void -> "void"
  | Array(var_type, size) -> data_type var_type ^ "[" ^ string_of_int size ^ "]""
  | Boolean -> "boolean"

let operator = function
  | Add -> "+
  | Sub -> "-"
  | Mult -> "*"
  | Div -> "/"
  | Equal -> "=="
  | Neq -> "!="
  | Less -> "<"
  | Leq -> "<="
  | Greater -> ">
  | Geq -> ">="
let check_str_eq = function
  StrEqual -> true
  _ -> false

let rec expression = function
  StrLiteral(str) -> str
  | IntLiteral(i) -> string_of_int i
  | NegIntLiteral(i) -> "-" ^ string_of_int i
  | BoolLiteral(boolean) -> string_of_bool boolean
  | Id(id) -> id
  | Access(id, field) -> id ^^ "." ^ field
  | Assign(id, expr) -> id ^ "=" "(" expression expr
  | ArrayAssign(id, loc, expr) -> id ^ "[" ^ (expression loc ^ "] = " ^ expression expr
  (expression expr)
  | ArrayAccess(id, loc) -> id ^ "(" ^ (expression loc ^ "]")
  | Binop(expr1, op, expr2) -> if check_str_eq op then ((expression expr1) ^ (operator op) ^ (expression expr2)) ^ ")" else ((expression expr1) ^ (operator op) ^ (expression expr2))
  (expression expr2)
  | Boolean(op, expr) -> ((operator op) ^ (expression expr))
  | Call(fname, arg) ->
    let rec expr_list = function
    [ ] -> ""
    [ ] -> (expression solo)
    | hd::tl -> ((expression hd ^ "," ^ (expr_list tl))
    in {
      if fname = "getInputFromOptions" then
        "promptForInput(new String[]{" ^ expr_list arg ^ "})"
      else if fname = "getInputAdjacentRooms" then
        "getInputAdjacentRooms(currentRoom)"
      else if fname = "print" then
        ("System.out.print" ^ "(" ^ expr_list arg ^ ")")
      else if fname = "arrlen" then
        ((expr_list arg) ^ ".length")
      else fname ^ "(" ^ expr_list arg ^ ")"
    }
    | End -> "break"

let expression_with_semi (expr) = ((expression expr) ^ ";\n")

let rec statement_list = function
[ ] -> ""
| hd::tl ->
  let rec statement = function
  Block(stmt_list) -> "{" ^ (statement_list stmt_list) ^ "}"
  | Expr(expr) -> (expression_with_semi expr)
  | Return(expr) -> ("return " ^ expression_with_semi expr
  | If(expr, stmt1, stmt2) -> "if (" ^ (expression expr) ^ ") " ^
    (statement stmt1) ^ "else" ^
    (statement stmt2)
  | While(expr, stmt) -> "while (" ^ (expression expr) ^ ") " ^
    (statement stmt)
let rec forms_list = function
    [] -> ""
    [solo] -> formal solo
    | hd::tl -> ((formal hd) ^ "," ^ (formals_list tl))

let local = function
    ArrayDecl(var_type, expr, str) -> ((data_type var_type) ^ "[]" ^ str ^ "= new " ^
    (data_type var_type) ^ "[" ^ (expression
    ^ "]")
    | Var(var_type, str) -> ((data_type var_type) ^ "" ^ str)
    | VarInit(var_type, str, expr) -> ((data_type var_type) ^ "" ^ str ^
    " = " ^ (expression expr))

let rec locals_list = function
    [] -> ""
    | hd::tl -> ((local hd) ^ ";\n" ^ (locals_list tl))

let vdecl = function
    ArrayDecl(var_type, expr, str) -> ((data_type var_type) ^ "[]" ^
    str ^ "= new " ^ (data_type var_type) ^ "["
    ^ (expression expr) ^ "]";
    | Var(vtype, id) -> (data_type vtype) ^ "" ^ id ^ ";\n"
    | VarInit(vtype, id, expr) -> (data_type vtype) ^ "" ^ id ^ " = " ^
    expression_with_semi expr

let rec vdecl_list = function
    [] -> ""
    | hd::tl -> "\t" ^ (vdecl hd) ^ (vdecl_list tl)

let global_vdecl = function
    ArrayDecl(var_type, expr, str) ->
        ("static " ^ (data_type var_type) ^ "[]" ^
        str ^ "= new " ^ (data_type
        var_type) ^ "[" ^ (expression
        expr) ^ "]";
        "public static " ^ (data_type vtype) ^ " " ^
        id ^ ";\n"
    | VarInit(vtype, id, expr) -> "public static " ^ (data_type vtype) ^ " " ^
        id ^
        " = " ^ expression_with_semi expr

let rec global_vdecl_list = function
    [] -> ""
    | hd::tl -> "\t" ^ (global_vdecl hd) ^ (global_vdecl_list tl)

let funcdecl f =
    ("public static " ^ (data_type f.returntype) ^ " " ^ f.fname ^ "(" ^
    (formals_list f.formals) ^ ")"){\n" ^ (locals_list f.locals) ^
    (statement_list f.body) ^ "\t\n"
}

let rec funcdecl_list = function
    [] -> ""
    | hd::tl -> "\t" ^ (funcdecl hd) ^ "\t" ^ (funcdecl_list tl)) ^ "\n"
let rec room_props_list proplist prefix = match proplist with
  [] -> ""
  | hd::tl -> prefix ^ "." ^ (statement_list [hd]) ^ (room_props_list tl prefix)

let room_decl r =
  "Room " ^ r.rname ^ " = new Room();\"troomMap.put("" ^ r.rname ^""\", "
^ r.rname ^ ");\"n" ^ (room_props_list r.rbody r.rname)

let rec room_decl_list = function
  [] -> ""
  | hd::tl -> "\t\t" ^ ((room_decl hd) ^ "\n" ^ (room_decl_list tl))

let adj_decl = function
  [] -> ""
  | hd::tl -> hd ^.setAdjacent(" ^ (List.hd tl) ");"

let rec adj_decl_list = function
  [] -> ""
  | hd::tl -> "\t\t" ^ ((adj_decl hd) ^ "\t" ^ (adj_decl_list tl)) ^ "\n"

let start_decl s =
  "\t\tcurrentRoom = " ^ s ^ ";\n"

let pred_stmt s =
  "if(" ^ (expression s.pred) ^ ")\n\t" ^ vdecl_list s.locals ^ statement_list s.body ^ 
")"

let rec pred_stmt_list = function
  [] -> ""
  | hd::tl -> "\t" ^ ((pred_stmt hd) ^ "\n\t" ^ (pred_stmt_list tl)) ^ "\n"

let rec npc_props_list proplist prefix = match proplist with
  [] -> ""
  | hd::tl -> prefix ^ "." ^ (statement_list [hd]) ^ (npc_props_list tl prefix)

let npc_decl n =
  "Npc " ^ n.name ^ " = new Npc();\n\tt\n\t\t" ^ (npc_props_list n.body n.name)

let rec npc_decl_list = function
  [] -> ""
  | hd::tl -> "\t\t" ^ ((npc_decl hd) ^ "\n\t" ^ (npc_decl_list tl))

let rec item_props_list proplist prefix = match proplist with
  [] -> ""
  | hd::tl -> prefix ^ "." ^ (statement_list [hd]) ^ (item_props_list tl prefix)

let item_decl i =
  "Item " ^ i.name ^ " = new Item();\n\tt\n\t\t" ^ (item_props_list i.body i.name)

let rec item_decl_list = function
  [] -> ""
  | hd::tl -> "\t\t" ^ ((item_decl hd) ^ "\n" ^ (item_decl_list tl))

let default Globals =
  "
  public static Scanner scanner;
  public static Room currentRoom;
  public static String input = \"\\n\";"
public static HashMap<String, Room> roomMap = new HashMap<String, Room>();

let driver_code (driver_class) =
  let (vars, main, fdecls, lib_funcs) = driver_class in
  "import java.util.*;\n\npublic class Driver {
  \n  \n  default_globals

  global_vdecl_list vars

  \n  public static void main(String[] args) {
    \n    scanner = new Scanner(System.in);
    \n    roomDecl_list main.rdecls

    main.start

    npcDecl_list main.ndecls

    itemDecl_list main.idecls

    \n    while (true) {
      \n      scanner.close();
      \n      funcDecl_list fdecls

      lib_funcs

      }

    }

    let room_constructor = "\n\npublic Room() {
  \n  adjRooms = new ArrayList<Room>();
  \n  }\n  \n"

let room_adj_functions = "\n\npublic void setAdjacent(Room room) {
  adjRooms.add(room);
  room.adjRooms.add(this);
  \n  }

\n\npublic boolean isAdjacent(Room room) {
  return adjRooms.contains(room);
  \n  }

let room_adj_field = "\n\npublic ArrayList<Room> adjRooms;
  \n"

let room_code (room_def) =
  "import java.util.*;\n\npublic class Room {

  String name;

  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n  \n
9.1.8 End-to-end Runner (tbag.ml)

(* Authors: Brian *)
open Printf

let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
  let checked_program = Semantic_checker.check_program program in
  let jast_program = Java_builder.rearrange checked_program in
  Code_gen.pretty_print jast_program;

9.1.9 Compile and Run Script (run_tbag.sh)

#!/bin/bash
# Authors: Brian
python scripts/importLibrary.py $1

./tbag < prog_w_stdlib.tbag
rm prog_w_stdlib.tbag
javac Driver.java
javac Room.java
java Driver

9.1.9 Test Script (run_tests.sh)

#!/bin/sh
# Inspired by Professor
# Authors: Maria
# Inspired by Professor Edwards’s microC script
# TBAG="/java_tbag.sh"

# Set time limit for all operations
ulimit -t 30

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

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

80
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" > &2  
  diff -b "$1" "$2" "$3" > &2 || {  
    SignalError "$1 differs"  
    echo "FAILED $1 differs from $2" > &2  
  }
}

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

Check() {  
  error=0  
  basename=`echo $1 | sed 's/.*\//\/[\w\-]*$/'`  
  reffile=`echo $1 | sed 's/.*\//\/[\w\-]*$/'`  
  basendir=`echo $1 | sed 's/.*\//\/[\w\-]*$/'`  
  echo -n "$basename..."  
  echo 1>&2  
  echo "##### Testing $basename" 1>&2
  
  generatedfiles=""  
  generatedfiles="$generatedfiles $basename.out" &
  Run "$TBAG $1" "$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

81
while getopt kdpsh c; do
    case $c in
        k) # Keep intermediate files
            keep=1
            ;
        h) # Help
            Usage
            ;
        esac
    done

shift `expr $OPTIND - 1`

if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/fail_*.*tbag tests/test_*.*tbag"
fi

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

exit $globalerror

9.1.10 Helper script for testing (java_tbag.sh)

#!/bin/sh
# Contributions from Maria, Julie

basename=`echo $1 | sed 's/.*\\\//s/./tbag/\//'`
inputtestdirectory="tests/"
driverfile="Driver.java"

# clean up all existing .java files
rm -f Driver.java Item.java Npc.java Room.java *.*class

82
# add tbag standard library to the end of the file
python scripts/importLibrary.py $1

./tbag < prog_w_stdbib.tbag > ${basename}_compiler_output.txt 2>&1
rm prog_w_stdbib.tbag

if [ -f $driverfile ]; then
    javac Driver.java

    if [[ ${basename} = *"_input"* ]]
    then
        java Driver < "$inputtestsdirectory"${basename}.in
    else
        java Driver
    fi
else
    cat ${basename}_compiler_output.txt
fi

rm ${basename}_compiler_output.txt
rm -f Driver.java Item.java Npc.java Room.java *.class

9.1.11 Makefile
# authors: all

default: compiler

compiler: scanner parser semantic_checker java_builder code_gen tbagger
    ocamlc -o tbag scanner.cmo parser.cmo java_builder.cmo code_gen.cmo semantic_checker.cmo tbag.cmo

tbagger:
    ocamlc -c tbag.ml;

code_gen:
    ocamlc -c code_gen.ml

java_builder: jast
    ocamlc -c java_builder.ml
scanner: parser
  ocamllex scanner.mll; ocamlc -o scanner scanner.ml

parser: jast
  ocamlyacc parser.mly; ocamlc -c parser.mli; ocamlc -c parser.ml

jast: ast
  ocamlc -c jast.mli

semantic_checker: ast scanner
  ocamlc -c semantic_checker.ml

ast:
  ocamlc -c ast.mli

.PHONY: clean
clean:
  rm -f scanner.ml parser.ml parser.mli *.cmo *.cmi scanner a.out *.cmx
  scannertraced* *.o *.class tbag *.java

9.1.12 Standard TBAG library (stdlib.tbag)
/* Authors: Julie */

func void intPrintLine(int a) {
    print(a);
    print("\n");
}

func void strPrintLine(string s) {
    print(s);
    print("\n");
}

func void boolPrintLine(boolean b) {
    print(b);
    print("\n");
}

9.1.13 Type conversion TBAG library (stdlib.tbag)

/* Authors: Julie */

func int intFromLetter(string letter) {
    if (letter == "A") { return 0; } else {}
    if (letter == "B") { return 1; } else {}
    if (letter == "C") { return 2; } else {}
    if (letter == "D") { return 3; } else {}
    if (letter == "E") { return 4; } else {}
    if (letter == "F") { return 5; } else {}
    if (letter == "G") { return 6; } else {}
    if (letter == "H") { return 7; } else {}
    if (letter == "I") { return 8; } else {}
    if (letter == "J") { return 9; } else {}
    if (letter == "K") { return 10; } else {}
    if (letter == "L") { return 11; } else {}
    if (letter == "M") { return 12; } else {}
    if (letter == "N") { return 13; } else {}
    if (letter == "O") { return 14; } else {}
    if (letter == "P") { return 15; } else {}
    if (letter == "Q") { return 16; } else {}
    if (letter == "R") { return 17; } else {}
    if (letter == "S") { return 18; } else {}
    if (letter == "T") { return 19; } else {}
}
if (letter == "U") { return 20; } else {
    if (letter == "V") { return 21; } else {
        if (letter == "W") { return 22; } else {
            if (letter == "X") { return 23; } else {
                if (letter == "Y") { return 24; } else {
                    if (letter == "Z") { return 25; } else {
                        return neg 1;
                    }
                }
            }
        }
    }
}

func string letterFromInt(int i) {
    if (i == 0) { return "A"; } else {
        if (i == 1) { return "B"; } else {
            if (i == 2) { return "C"; } else {
                if (i == 3) { return "D"; } else {
                    if (i == 4) { return "E"; } else {
                        if (i == 5) { return "F"; } else {
                            if (i == 6) { return "G"; } else {
                                if (i == 7) { return "H"; } else {
                                    if (i == 8) { return "I"; } else {
                                        if (i == 9) { return "J"; } else {
                                            if (i == 10) { return "K"; } else {
                                                if (i == 11) { return "L"; } else {
                                                    if (i == 12) { return "M"; } else {
                                                        if (i == 13) { return "N"; } else {
                                                            if (i == 14) { return "O"; } else {
                                                                if (i == 15) { return "P"; } else {
                                                                    if (i == 16) { return "Q"; } else {
                                                                        if (i == 17) { return "R"; } else {
                                                                            if (i == 18) { return "S"; } else {
                                                                                if (i == 19) { return "T"; } else {
                                                                                    if (i == 20) { return "U"; } else {
                                                                                        if (i == 21) { return "V"; } else {
                                                                                            if (i == 22) { return "W"; } else {
                                                                                                if (i == 23) { return "X"; } else {
                                                                                                                                                    if (i == 24) { return "Y"; } else {
                                                                                                                                                    if (i == 25) { return "Z"; } else {
                                                                                                                                                    return ";
                                                                                                }
                                                                                            }
                                                                                        }
                                                                                    }
                                                                                }
                                                                            }
                                                                        }
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

9.1.14 Driver functions (driver_functions.txt)
/ * Authors: Julie */

// this is what happens when u do player->room
public static void movePlayerToRoom(Object room) {
    if (room instanceof Room) {
        currentRoom = (Room) room;
    } else {
        Room update = roomMap.get(room);
        currentRoom = update;
    }
}

// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid input.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.

// Arguments:
// String[] acceptableInputs -- the list of acceptable inputs
public static void promptForInput(String[] acceptableInputs) {
    System.out.println("Choose from one of the following options:");
    for (String option : acceptableInputs) {
        System.out.print(option + " ");
    }
    System.out.println();
    // loop until player enters valid input
    input = scanner.nextLine();
    System.out.println("Input: " + input);
    while (!Arrays.asList(acceptableInputs).contains(input)) {
        System.out.println("Invalid Input. Try again.");
        input = scanner.nextLine();
        System.out.println("Input: " + input);
    }
    System.out.println();
    System.out.println();
}

// Gets all the adjacencies for the room entered as argument and displays these adjacencies to player.
// Prompts player for input and sets global var "input" to whatever player submitted, provided it's a valid adjacency.
// If invalid inputs are entered, it'll reprompt until player enters a valid input.
// pretty much exactly same as promptForInputs(), except it takes in a room as an argument instead of a list of strings
public static void getInputAdjacentRooms(Room room) {
    String[] acceptableInputs = new String[room.adjRooms.size()];
    int i = 0;
    for (Room r : room.adjRooms) {
        acceptableInputs[i] = r.name;
        i++;
    }

    System.out.println("Choose from one of the following options:");
    for (String option : acceptableInputs) {
        System.out.print(option + "   ");
    }
    System.out.println();

    // loop until player enters valid input
    input = scanner.nextLine();
    System.out.println("Input: " + input);
    while (!Arrays.asList(acceptableInputs).contains(input)) {
        System.out.println("Invalid Input. Try again.");
        input = scanner.nextLine();
        System.out.println("Input: " + input);
    }
    System.out.println();
    System.out.println();
}

9.1.15 Test suite

fail_arr_assign.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
func int testvar() {
    int [10] i;
    i ["hello"] = 3;
    return 0;
}

fail_arr_assign.out:

Fatal error: exception Failure("Positional array access specifier must be an
    Integer")

fail_arr_assign2.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    int [10] i;
    i [0] = "hello";
    return 0;
}

fail_arr_assign2.out:

Fatal error: exception Failure("Right hand side of assignment statement does
    not match type of array")

fail_arr_assign3.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {

int i;
i[0] = 4;
return 0;
}

fail_arr_assign3.out:

Fatal error: exception Failure("Left hand side of array assignment must be an array")

fail_arr_assign4.tbag:

/* Authors: Maria */
true {
testvar();
endgame;
}

func int testvar() {
int i;
int j = i[0];
return 0;
}

fail_arr_assign4.out:

Fatal error: exception Failure("Array access must be used on an array type")

fail_arr_decl.tbag:

/* Authors: Maria */
true {
testvar();
endgame;
}

func int testvar() {
int ["hello"] i;
return 0;
}

fail_arr_decl.out:
Fatal error: exception Failure("Array size must be integer")

fail_arr_decl2.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    void [10] i;
    return 0;
}

fail_arr_decl2.out:

Fatal error: exception Failure("Invalid variable type used")

fail_arr_len.tbag:

/* Authors: Maria */
true {
    int a;
    int len;
    len = arrLen(a);
    print(len);
    endgame;
}

fail_arr_len.out:

Fatal error: exception Failure("arrLen expects an array argument")

fail_arr_len2.tbag:

/* Authors: Maria */
true {
    int a;
    int b;
    int len;
len = arrLen(a, b);
print(len);
endgame;
}

fail_arr_len2.out:

Fatal error: exception Failure("Function arrLen does
not exist with the given parameters.")

fail_exist_var.tbag:

/* Authors: Maria */
int i = 0;

true {
    testvar();
    endgame;
}

func int testvar() {
    int i = 7;
    return 0;
}

fail_exist_var.out:

Fatal error: exception Failure("Variable with name i exists.")

fail_func_call.tbag:

/* Authors: Maria */
ture {
    int x = gcd("yo", 14);
    endgame;
}

func int gcd(int a, int b) {
    while (a != b) {
        if (a > b) { a = a - b; }
        else { b = b - a; }
    }
    return a;
}
fail_func_call.out:

Fatal error: exception Failure("Function gcd does not exist with the given parameters.")

fail_func_call2.tbag:

/* Authors: Maria */
true {
    int x = gcd(14);
    endgame;
}

func int gcd(int a, int b) {
    while (a != b) {
        if (a > b) { a = a - b; }
        else { b = b - a; }
    }
    return a;
}

fail_func_call2.out:

Fatal error: exception Failure("Function gcd does not exist with the given parameters.")

fail_func_call3.tbag:

/* Authors: Maria */
true {
    int x = gcd("yo", 14);
    endgame;
}

fail_func_call3.out:

Fatal error: exception Failure("Function gcd does not exist with the given parameters.")

fail_func_decl.tbag:
/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    int [10] i;
    return 0;
}

fail_func_decl1.out:

Fatal error: exception Failure("Function with name testvar and given argument types exists")

fail_func_decl2.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar(int j) {
    int [10] i;
    return 0;
}

func int testvar(int k) {
    int [10] i;
    return 0;
}

fail_func_decl2.out:

Fatal error: exception Failure("Function with name testvar and given
argument types exists")

fail_func_decl3.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar(int j) {
    int [10] i;
    return "hello";
}

fail_func_decl3.out:

Fatal error: exception Failure("Return type of expression does not match return type of function")

fail_func_var_decl.tbag:

/* Authors: Iris */
true {
    testvar();
    endgame;
}

func int testvar() {
    int i = 0;
    int i;
    return 0;
}

fail_func_var_decl.out:

Fatal error: exception Failure("Variable with name i exists.")

fail_gif.a.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

cmp {
    string name;
    string roomName;
    int hunger;
}

cmp Cat {
    name = "Tubbs";
    roomName = "Bedroom";
    hunger = 5;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(Outside);
    ->input
}

currentRoom.name == Cat.roomName {
    print("you got eaten by the cat.
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");}
fail_gifa.out:

Fatal error: exception Failure("undeclared identifier Outside")

fail_gifo.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Bedroom }

currentRoom == Closet {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    getInputFromOptions("Bedroom", 1);
    ->input
    print("\n");
    endgame;
}

fail_gifo.out:

Fatal error: exception Failure("getInputFromOptions expects one or more string arguments")

fail_id_func.tbag:

/* Authors: Maria */
true {
    test();
    endgame;
}
func int test() {
    int a = 8;
    print(b);
}

fail_id_func.out:
Fatal error: exception Failure("undeclared identifier b")

fail_if.tbag:
/* Authors: Maria */
true {
    testif();
    endgame;
}

func int testif() {
    if ("hello") {
        print("true ");
    } else {
        print("hi");
    return 0;
}

fail_if.out:
Fatal error: exception Failure("Expression in if statement conditional must be
    of type Boolean")

fail_item_decl.tbag:
/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

item {
    string name;
    string roomName;
    int hunger;
}

item Cat {
    name = "Tubbs";
    roomName = "Bedroom";
}

boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}

currentRoom.name == Cat.roomName {
    print("you got eaten by the cat.
");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}
fail_item_decl.out:

Fatal error: exception Failure("number of item decl fields do not match definition.")

fail_item_decl2.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

item {
    string name;
    string roomName;
    int hunger;
}

item Cat {
    name = "Tubbs";
    roomName = "Bedroom";
    swag = 9999;
}

boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}
true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    -> input
}

currentRoom.name ~~ Cat.roomName {
    print("you got eaten by the cat.");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

fail_item_decl2.out:

Fatal error: exception Failure("field name in npc decl does not exist.")

fail_item_def.tbag:

/* Authors: Maria */

room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

item {
    string name;
    string roomName;
    void hunger;
}
item Cat {
    name = "Tubbs";
    roomName = "Bedroom";
    hunger = 5;
}

boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}
currentRoom.name ~ Cat.roomName {
    print("you got eaten by the cat.\n");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

fail_item_def.out:

Fatal error: exception Failure("Invalid variable type used")

fail_notexist_id.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}
func int testvar() {
    int x = i;
    return 0;
}

fail_notexist_id.out:
Fatal error: exception Failure("undeclared identifier i")

fail_notexist_var.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    i = 7;
    return 0;
}

fail_notexist_var.out:
Fatal error: exception Failure("undeclared identifier i")

fail_npc_decl.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }
npc {
    string name;
    string roomName;
    int hunger;
}

npc Cat {
    name = "Tubbs";
    roomName = "Bedroom";
}

boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

ture {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}

currentRoom.name ~~ Cat.roomName {
    print("you got eaten by the cat.\n");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

fail_npc_decl.out:

Fatal error: exception Failure("number of npc decl fields do not match definition.")

fail_npc_decl2.tbag:
/* Authors: Maria */

```c++

room {}  

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

npc {
    string name;
    string roomName;
    int hunger;
}

npc Cat {
    name = "Tubbs";
    roomName = "Bedroom";
    swag = 9999;
}

boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}

currentRoom.name == Cat.roomName {
    print("you got eaten by the cat.
    endgame;
```
func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

fail_npc_decl2.out:

Fatal error: exception Failure("field name in npc decl does not exist.")

fail_npc_def.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Closet }

npc {
    string name;
    string roomName;
    void hunger;
}

npc Cat {
    name = "Tubbs";
    roomName = "Bedroom";
    hunger = 5;
}

boolean started = false;
NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

true {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}
currentRoom.name ~~ Cat.roomName {
    print("you got eaten by the cat.
");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

fail_npc_def.out:

Fatal error: exception Failure("Invalid variable type used")

fail_ops.tbag:

/* Authors: Maria */
#import stdlib

ture {
    printLine("hello" + 2);
    endgame;
}

func int printLine(string b) {
    print(b);
    print("\n");
    return 0;
}
fail_ops.out:

Fatal error: exception Failure("Types to arithmetic operators +, -, *, / must both be Int")

fail_ops2.tbag:

/* Authors: Maria */
true {
    println(1 == "hello");
    endgame;
}

func int println(boolean b) {
    print(b);
    print("\n");
    return 0;
}

fail_ops2.out:

Fatal error: exception Failure("Types to equality operators ==, != must be the same and be integers, booleans, or rooms")

fail_ops3.tbag:

/* Authors: Maria */
true {
    println(true == "hello");
    endgame;
}

func int println(boolean b) {
    print(b);
    print("\n");
    return 0;
}

fail_ops3.out:
Fatal error: exception Failure("Types to equality operators ==, !=
must be the same and be integers, booleans, or
rooms")

fail_ops4.tbag:

/* Authors: Maria */
room {};

room Closet {
    name = "Closet";
}

room LivingRoom {
    name = "Living Room";
}

Closet <-> LivingRoom;

start { Closet }

currentRoom == "hello" {
    print("\n");
    endgame;
}

fail_ops4.out:

Fatal error: exception Failure("Types to equality operators ==, !=
must be the same and be integers, booleans, or
rooms")

fail_ops5.tbag:

/* Authors: Maria */
room {};

room Closet {
    name = "Closet";
}

room LivingRoom {
name = "Living Room";
}

Closet <-> LivingRoom;

start { Closet }

currentRoom == "hello" {
    print("\n");
    endgame;
}

fail_ops5.out:

Fatal error: exception Failure("Types to equality operators ==, != must be the same and be integers, booleans, or rooms")

fail_ops6.tbag:

/* Authors: Maria */
true {
    printLine(true > 1);
    endgame;
}

func int printLine(boolean b) {
    print(b);
    print("\n");
    return 0;
}

fail_ops6.out:

Fatal error: exception Failure("Types to integer comparison operators <, <=, >, >= must be integers")

fail_ops7.tbag:

/* Authors: Maria */
true {
    printLine("hello" ~ ~ 1);
    endgame;

func int printLine(boolean b) {
    print(b);
    print("\n");
    return 0;
}

fail_ops7.out:
Fatal error: exception Failure("Types to ~~ must both be String")

fail_ops8.tbag:
/* Authors: Maria */
true {
    printLine("hello" AND 1);
    endgame;
}

func int printLine(boolean b) {
    print(b);
    print("\n");
    return 0;
}

fail_ops8.out:
Fatal error: exception Failure("Types to binary boolean operators AND, OR must both be Boolean")

fail_ops9.tbag:
/* Authors: Maria */
true {
    printLine(NOT "hello");
    endgame;
}

func int printLine(boolean b) {
    print(b);
    print("\n");
    return 0;
fail_ops9.out:

Fatal error: exception Failure("Type to unary boolean NOT operator must be boolean")

fail_pred_expr.tbag:

/* Authors: Iris */
a {
    endgame;
}

func int test() {
    int a = 8;
}

fail_pred_expr.out:

Fatal error: exception Failure("undeclared identifier a")

fail_rec_func.tbag:

/* Authors: Maria */
true {
    print(fib(5));
    endgame;
}

func int fib(int x) {
    if (x < 2) { return 1; }
    else { return fib("hello") + fib(x-2); }
}

fail_rec_func.out:

Fatal error: exception Failure("Function fib does not exist with the given parameters.")

fail_room_decl.tbag:

112
/ * Authors: Iris */
  
  room {
    int num_cats;
  }

  room Closet {
    name = "Closet";
  }

  room Bedroom {
    name = "Bedroom";
  }

  room LivingRoom {
    name = "Living Room";
  }

  room Outside {
    name = "Outside";
  }

  Closet < - > Bedroom;
  Bedroom < - > LivingRoom;

  start { Bedroom }

  boolean madeItOutside = false;

  currentRoom == Bedroom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    get_input_from_options("Closet", "LivingRoom");
    ->input
    print("\n");
  }

  currentRoom == Closet {
    string testName = currentRoom.name;
    print("Currently in: ");
    print(testName);
  }
print("\n");
get_input_from_options("Bedroom");
->input
print("\n");

}  
currentRoom == LivingRoom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    get_input_from_options("Bedroom", "Outside");
    ->input
    print("\n");
}

currentRoom == Outside {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    print("\n");
    madeItOutside = true;
}

madeItOutside {
    print("Good job making it outside, lazybones.");
    print("\n");
    endgame;
}

fail_room_decl.out:

Fatal error: exception Failure("number of room decl fields do not match
definition.")

fail_room_decl2.tbag:

/* Authors: Maria */
room {
    int num_cats;
}

room Closet {
    name = "Closet";}
num_dogs = 4;
}

room Bedroom {
    name = "Bedroom";
    num_dogs = 4;
}

room LivingRoom {
    name = "Living Room";
    num_dogs = 4;
}

room Outside {
    name = "Outside";
    num_dogs = 4;
}

Closet <-> Bedroom;
Bedroom <-> LivingRoom;

start { Bedroom }

boolean madeItOutside = false;

currentRoom == Bedroom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    get_input_from_options("Closet", "LivingRoom");
    ->input
    print("\n");
}

currentRoom == Closet {
    string testName = currentRoom.name;
    print("Currently in: ");
    print(testName);
    print("\n");
    get_input_from_options("Bedroom");
    ->input
    print("\n");
}
currentRoom == LivingRoom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    get_input_from_options("Bedroom", "Outside");
    ->input
    print("\n");
}

currentRoom == Outside {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    print("\n");
    madeItOutside = true;
}

madeItOutside {
    print("Good job making it outside, lazybones.");
    print("\n");
    endgame;
}

fail_room_decl2.out:

Fatal error: exception Failure("field name in room decl does not exist.")

fail_room_def.tbag:

/* Authors: Maria */
room {
    string place;
    void nonsense;
}

room Test {
    name = "Test";
    place = "here";
    nonsense = huh;
}
room Test2 {
    name = "Test2";
    place = "there";
    nonsense = wha;
}

Test <-> Test2;

start { Test }

true {
    endgame;
}

fail_room_def.out:

Fatal error: exception Failure("room defs didn't check out")

fail_undef_room.tbag:

/* Authors: Maria */
room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

Closet <-> Bedroom;

start { Bedroom }

currentRoom == LivingRoom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    get_input_from_options("Bedroom", "Closet");
    ->input
    print("\n");
}
endgame;
}

fail_undef_room.out:

Fatal error: exception Failure("undeclared identifier LivingRoom")

fail_var_assign.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    int [10] i;
    i = 3;
    return 0;
}

fail_var_assign.out:

Fatal error: exception Failure("Left hand side of assignment statement must be a non-array variable")

fail_var_assign2.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    int i;
    string j = "hello";
    i = j;
    return 0;
}

fail_var_assign2.out:

118
Fatal error: exception Failure("Type mismatch in assignment statement")

fail_var_decl1.tbag:

/* Authors: Maria */
true {
  testvar();
  endgame;
}

func int testvar() {
  void i;
  return 0;
}

fail_var_decl1.out:

Fatal error: exception Failure("Invalid variable type used")

fail_var_decl2.tbag:

/* Authors: Maria */
true {
  testvar();
  endgame;
}

func int testvar() {
  void i = 2;
  return 0;
}

fail_var_decl2.out:

Fatal error: exception Failure("Invalid variable type used")

fail_var_init.tbag:

/* Authors: Maria */
true {
  testvar();
  endgame;
func int testvar() {
    int i = "hello";
    return 0;
}

fail_var_init.out:
Fatal error: exception Failure("Type mismatch in variable initialization")

fail_vdecl_exists.tbag:
/* Authors: Iris */
true{
    string a = "blah";
    string a = "huh";
}

fail_vdecl_exists.out:
Fatal error: exception Failure("Variable with name a exists.")

fail_vdecl_ref.tbag:
/* Authors: Iris */
true {
    a = "ha";
}

fail_vdecl_ref.out:
Fatal error: exception Failure("undecleared identifier a")

fail_void_arr.tbag:
/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {

}
void [10] i;
    return 0;
}

fail_void_arr.out:

Fatal error: exception Failure("Invalid variable type used")

fail_void_var.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    void i;
    return 0;
}

fail_void_var.out:

Fatal error: exception Failure("Invalid variable type used")

fail_void_var2.tbag:

/* Authors: Maria */
true {
    testvar();
    endgame;
}

func int testvar() {
    void i = 7;
    return 0;
}

fail_void_var2.out:

Fatal error: exception Failure("Invalid variable type used")

fail_while.tbag:
/ * Authors: Maria */
int i = 0;

trued { 
    whileTest();
    endgame;
}

func void whileTest() {
    while (i) {
        print(i);
        print("\n");
        i = i + 1;
    }
    print(666);
}

fail_while.out:

Fatal error: exception Failure("Expression in while statement conditional
must be
of type Boolean")

test_0npc_0item_2rooms.tbag:

/* Authors: Julie */
room {
    int temperature;
}

room Space {
    name = "space";
    temperature = neg 1000;
}

room Sun {
    name = "surface of the sun";
    temperature = 99999;
}

Space <-> Sun;
start { Space }

currentRoom == Space {
    print("you are in ");
    print(currentRoom.name);
    print(" . you're being launched into the surface of the sun\n");
    print(" currently the temperature is ");
    print(currentRoom.temperature);
    print("\n");
    ->Sun
}

currentRoom == Sun {
    print("Now you're on the ");
    print(currentRoom.name);
    print(" the temp is ");
    print(currentRoom.temperature);
    endgame;
}

test_npc_0item_2rooms.out:

you are in space. you're being launched into the surface of the sun
currently the temperature is -1000
Now you're on the surface of the sun the temp is 99999

test_npc_1item_0rooms.tbag:

/* Authors: Julie */

item {
    string name;
}

item Purse {
    name = "fluffy purse";
}

true {
    print("You have a ");
    print(Purse.name);
    endgame;
}

test_npc_1item_0rooms.out:
You have a fluffy purse
test_0npc_1item_2rooms.tbag:

/* Authors: Julie */
room {
    int temperature;
}

room Space {
    name = "space";
    temperature = neg 1000;
}

room Sun {
    name = "surface of the sun";
    temperature = 99999;
}

Space <-> Sun;

start { Space }

item {
    string name;
}

item Purse {
    name = "fluffy purse";
}

currentRoom == Space {    
    print("you are in ");
    print(currentRoom.name);
    print(" you're being launched into the surface of the sun\n");
    print(" currently the temperature is ");
    print(currentRoom.temperature);
    print(" \n");
    ->Sun
}

currentRoom == Sun {    
    print("Now you're on the ");
}
print(currentRoom.name);
print(" the temp is ");
print(currentRoom.temperature);
print(" . You have a ");
print(Purse.name);
endgame;
}

test_0npc_1item_2rooms.out:

you are in space. you're being launched into the surface of the sun
currently the temperature is -1000
Now you're on the surface of the sun the temp is 99999. You have a fluffy purse

test_1npc_0item_0rooms.tbag:

/* Authors: Julie */
npc {
    string name;
    string color;
}
npc Cat {
    name = "Tubbs";
    color = "white";
}
true {
    print("There's a cat here. Its name is ");
    print(Cat.name);
    print(" and it's ");
    print(Cat.color);

    endgame;
}

test_1npc_0item_0rooms.out:

There's a cat here. Its name is Tubbs and it's white

test_1npc_0item_2rooms.tbag:

/* Authors: Julie */
room {

125
int temperature;
}

room Space {
    name = "space";
    temperature = neg 1000;
}

room Sun {
    name = "surface of the sun";
    temperature = 99999;
}

Space <-> Sun;

start { Space }

npc {
    string name;
    string color;
}

cnpc Cat {
    name = "Tubbs";
    color = "white";
}

currentRoom == Space {
    print("you are in ");
    print(currentRoom.name);
    print(" you're being launched into the surface of the sun\n");
    print(" currently the temperature is ");
    print(currentRoom.temperature);
    print("\n");
    ->Sun
}

currentRoom == Sun {
    print("Now you're on the ");
    print(currentRoom.name);
    print("; the temp is ");
    print(currentRoom.temperature);
    print(". There's a cat here. Its name is ");
print(Cat.name);
print(" and it's ");
print(Cat.color);

dangelog;
}

test_1npc_0item_2rooms.out:
you are in space. you're being launched into the surface of the sun

currently the temperature is -1000
Now you're on the surface of the sun; the temp is 99999. There's a cat here. Its name is Tubbs and it's white

test_1npc_1item_0rooms.tbag:

/* Authors: Julie */
npc {
  string name;
  string color;
}

npc Cat {
  name = "Tubbs";
  color = "white";
}

item {
  string name;
}

item Purse {
  name = "fluffy purse";
}

ture {
  print("You have a ");
  print(Purse.name);
  print(" . There's a cat here. Its name is ");
  print(Cat.name);
  print(" and it's ");
  print(Cat.color);
  
dangelog;
test_1npc_1item_0rooms.out:

You have a fluffy purse. There's a cat here. Its name is Tubbs and it's white

test_1npc_1item_2rooms.tbag:

/* Authors: Julie */
room {
  int temperature;
}

room Space {
  name = "space";
  temperature = neg 1000;
}

room Sun {
  name = "surface of the sun";
  temperature = 99999;
}

Space <-> Sun;

start { Space }

npc {
  string name;
  string color;
}

cpy Cat {
  name = "Tubbs";
  color = "white";
}

item {
  string name;
}

item Purse {
  name = "fluffy purse";
}
currentRoom == Space {
    print("you are in ");
    print(currentRoom.name);
    print(". you're being launched into the surface of the sun\n");
    print("currently the temperature is ");
    print(currentRoom.temperature);
    print("\n");
    \rightarrow Sun
}

currentRoom == Sun {
    print("Now you're on the ");
    print(currentRoom.name);
    print("; the temp is ");
    print(currentRoom.temperature);
    print(". You have a ");
    print(Purse.name);
    print(". There's a cat here. Its name is ");
    print(Cat.name);
    print(" and it's ");
    print(Cat.color);

    endgame;
}

test_1npc_1item_2rooms.out:

you are in space. you're being launched into the surface of the sun
currently the temperature is -1000
Now you're on the surface of the sun; the temp is 99999. You have a fluffy purse. There's a cat here. Its name is Tubbs and it's white
test_add.tbag:

/* Authors: Julie */
true {
    print(666+24);
    endgame;
}

test_add.out:
test_arith1.tbag:

/* Authors: Julie */
true {
    print(420 + 8 * 69 - 5);
    endgame;
}

test_arith1.out:

967

test_arith2.tbag:

/* Authors: Julie */
true {
    print(20 - 8 / 2 + 5);
    endgame;
}

test_arith2.out:

21

test_arr_len_1.tbag:

/* Authors: Greg */
true {
    int[3] a;
    int len;
    len = arrLen(a);
    print(len);
    endgame;
}

test_arr_len_1.out:

3

test_array_decl_with_int_expr.tbag:

/* Authors: Greg */
true {
    int[1+1+1] a;
    a[2] = 2;
```c
print(a[2]);
endgame;
}

test_array_decl_with_int_expr.out:

2
test_array_in_func.tbag:

/* Authors: Greg */
true {
    testArr();
    endgame;
}

func void testArr() {
    int[1] a1;
    int[2] a2;
    a1[0] = 0;
    a2[0] = 0;
    a2[1] = 1;
    print(a1[0]);
    print(a2[0]);
    print(a2[1]);
}

test_array_in_func.out:

001
test_array_in_handler.tbag:

/* Authors: Greg */
true {
    int[1] a1;
    int[2] a2;
    a1[0] = 0;
    a2[0] = 0;
    a2[1] = 1;
    print(a1[0]);
    print(a2[0]);
    print(a2[1]);
    endgame;
}
```
test_array_in_handler.out:

001
test_call_stdlib_from_func.tbag:

/* Authors: Julie */
#import stdlib

test ( ) {
    test ( ) ;
    endgame ;
}

func void test ( ) {
    strPrintLine("hi");
}

test_call_stdlib_from_func.out:

hi
test_fib_event.tbag:

/* Authors: Julie */
int fibTerm = 6;
int currentTerm = 0;
int fib1 = 0;
int fib2 = 1;
int tmp = 0;

currentTerm < fibTerm {
    print ( fib2 );
    tmp = fib1;
    fib1 = fib2;
    fib2 = tmp + fib2;
    currentTerm = currentTerm + 1;
}

currentTerm >= fibTerm {
    endgame;
}
test_fib_event.out:

112358
test_fib_func.tbag:

/* Authors: Julie */
/* Based on Seven Weeks of Cat Monarchy, a game created by Fathom and Scuffy for the Ludum Dare 34 game jam. */
/* http://fathom.itch.io/seven-weeks-of-cat-monarchy */

true {
    print(fib(0));
    print(fib(1));
    print(fib(2));
    print(fib(3));
    print(fib(4));
    print(fib(5));
    endgame;
}

func int fib(int x) {
    if (x < 2) { return 1; }
    else { return fib(x-1) + fib(x-2); }
}

test_fib_func.out:

112358
test_func.tbag:

/* Authors: Julie */
true {
    int a;
    a = add(666, 3);
    print(a);
    endgame;
}

func int add(int a, int b) {
    return a + b;
}
test_func.out:

669
test_func2.tbag:

/* Authors: Julie */
true {
    printstuff(666, "hi", 69, "lol");
    endgame;
}

func int printstuff(int a, string b, int c, string d) {
    print(a);
    print(b);
    print(c);
    print(d);
    return 0;
}

test_func2.out:

666hi69lol
test_game_cat_kingdom_input.in:

East_Chamber
No
Great_Hall
West_Chamber
Yes
Great_Hall
West_Chamber
Yes
Great_Hall
East_Chamber
Yes
Great_Hall
Throne_Room
2
East_Chamber
Yes
Great_Hall
West_Chamber
Yes
Kitchen
Y
Z
Z
X
Y
Z
Z
Y
X
None
West_Chamber
Yes
Great_Hall
Throne_Room
1
East_Chamber
Yes
Great_Hall
Throne_Room
2
test_game_cat_kingdom_input.tbag:

/* Authors: Julie */
#import stdlib

room {} 

room Great_Hall { name = "Great_Hall"; } 
room Throne_Room { name = "Throne_Room"; } 
room East_Chamber { name = "East_Chamber"; } 
room West_Chamber { name = "West_Chamber"; } 
room Kitchen { name = "Kitchen"; }

Throne_Room <-> Great_Hall; West_Chamber <-> Great_Hall;
East_Chamber <-> Great_Hall; West_Chamber <-> Kitchen;

start { Great_Hall }

cpc { int id; string name; string roomName; string message;
    string goodResponse; string evilResponse;
    string goodResult; string badResult;
}
npc Tubbs {
    id = 0;
    name = "King Tubbs";
    roomName = "East_Chamber";
    message = "hi";
    goodResponse = ""; evilResponse = ""; goodResult = ""; badResult = "";
}

cmp Pickles {
    id = 1;
    name = "Duke Pickles";
    roomName = "West_Chamber";
    message = "Your greatness, welcome to the treasury. I am your financial advisor.\nI stand here amongst our hoard of cheese chunks to determine the general \"cheesiness\" of our monarchy.\nFeel free to drop in any time and ask how things are going, yes.\n";
    goodResponse = ""; evilResponse = ""; goodResult = ""; badResult = "";
}

cmp Marshmallow {
    id = 2;
    name = "Lady Marshmallow";
    roomName = "East_Chamber";
    message = "Hello, my new and temporary liege. I am your kingdom advisor.\nIt is my job to advise you on the health and happiness of your kingdom.\nMy own health and happiness is irrelevant.\nPlease, see me again some time.\n";
    goodResponse = ""; evilResponse = ""; goodResult = ""; badResult = "";
}

cmp Pumpkin {
    id = 3;
    name = "Pumpkin";
    roomName = "Throne_Room";
    message = "A cat baby is lost in the spooky forest!\nI know because it sent me this baby note, via forest squirrel!\n";
    goodResponse = "I will organize a search party!\n";
    evilResponse = "I will organize a snake party!\n";
goodResult = "They find a baby, and a dozen or so other babies.\n"
badResult = "You party with some snakes. What a night! The baby is never heard from again.\n";
}

npc Snowball {
    id = 4;
    name = "Snowball";
    roomName = "Throne_Room";
    message = "Hel... hello monarch!\nI'm...\nI'm very lonely.\nWould you mind if I just stood in here for a few minutes?\nI'm sorry. I can leave.\n"
    goodResponse = "Please, stay!\n"
    evilResponse = "LEAVE AT ONCE.\n"
    goodResult = "The cat leaves. You eat a royal pizza bagel.\n"
    badResult = "You both have pizza bagels for lunch.\n"
}

npc Patches {
    id = 5;
    name = "Patches";
    roomName = "Throne_Room";
    message = "Help! It's my son!\nHe is very sick!\nAnd stuck in a huge bear mouth!\n"
    goodResponse = "Guards, help this cat's son!\n"
    evilResponse = "Guards, help the bear eat this cat's son!\n"
    goodResult = "Your guards manage to save most of the son.\n"
    badResult = "Your guards lose a little bit more of themselves.\n"
}

item {
    int initialAmount;
}

item BowlX {
    initialAmount = 5;
}

item BowlY {

initialAmount = 7;
}

item BowlZ {
    initialAmount = 8;
}

/* to easily access the db */

string[10] catNames;
string[10] catRoomNames;
string[10] catMessages;
string[10] goodResponses;
string[10] evilResponses;
string[10] goodResults;
string[10] badResults;

string[5] requestTitleCard;

boolean started = false;
int week = 1;
boolean spokeToMarshmallow = false;
boolean spokeToPickles = false;
boolean rollOverWeek = true;
int stateOfKingdom = 5;
int cheeseCubeCount = 5;
int lengthSabbatical = 3;

boolean handleSubjects = false;

boolean dataInitialized = false;

boolean xyzPuzzleInProgress = false;
int bowlXAmount;
int bowlYAmount;
int bowlZAmount;

int xyzState = 1;

NOT dataInitialized {

138
Marshmallow.name;
Patches.name;

catRoomNames[0] = Tubbs.roomName; catRoomNames[1] = Pickles.roomName;
Pumpkin.roomName;
catRoomNames[4] = Snowball.roomName; catRoomNames[5] =
Patches.roomName;

Pumpkin.message;

goodResponses[0] = Tubbs.goodResponse; goodResponses[1] =
Pickles.goodResponse;
Pumpkin.goodResponse;
Patches.goodResponse;

evilResponses[0] = Tubbs.evilResponse; evilResponses[1] =
Pickles.evilResponse;
Pumpkin.evilResponse;
Patches.evilResponse;

goodResults[0] = Tubbs.goodResult; goodResults[1] = Pickles.goodResult;
goodResults[2] = Marshmallow.goodResult; goodResults[3] =
Pumpkin.goodResult;
goodResults[4] = Snowball.goodResult; goodResults[5] = Patches.goodResult;

badResults[0] = Tubbs.badResult; badResults[1] = Pickles.badResult;

requestTitleCard[1] = "YOUR CAT SUBJECTS HAVE SOME IMPORTANT
REQUESTS!\n\n"
requestTitleCard[2] = "GET READY FOR EVEN MORE REQUESTS FROM CAT
SUBJECTS!\n\n";
requestTitleCard[3] = "THE FINAL REQUESTS BEFORE YOUR REIGN IS AT AN
END!\n\n";
bowlXAmount = BowlX.initialAmount;
bowlYAmount = BowlY.initialAmount;
bowlZAmount = BowlZ.initialAmount;
dataInitialized = true;
}

} NOT started {
   strPrintLine("King Tubbs, the great monarch of the Cat Kingdom, has
recently discovered a sunbeam of sensational quality!");
   print("He has, understandably, requested a ");
   print(lengthSabbatical);
   strPrintLine(" week sabbatical.");
   strPrintLine("It is up to you (as an Official Visiting Noblecat) to
lead the kingdom during that time.");
   strPrintLine("Speak to your advisors and then make some important
decisions!");
   strPrintLine("At the end of seven weeks you can see exactly what sort
of ruler you have been.");
   strPrintLine("THE CAT MONARCHY AWAITS YOUR STEADY LEADERSHIP!\n\n");
   started = true;
}

rollOverWeek {
   print("------------------------ WEEK "); print(week);
   strPrintLine(" ------------------------\n\n");
   rollOverWeek = false;
}

currentRoom.name ~ Pickles.roomName {
   print("Ah, here is your financial advisor ");
   print(Pickles.name);
   strPrintLine("!");
   strPrintLine("Do you want to talk to him?");
   getInputFromOptions("Yes", "No");
}
currentRoom.name ~~ Pickles.roomName AND input ~~ "Yes" {
    print(Pickles.name);
    strPrintLine(" says: ");
    if (NOT spokeToPickles) {
        strPrintLine(Pickles.message);
        spokeToPickles = true;
    } else {
        print("We have ");
        print(cheeseCubeCount);
        print(" cheese cubes!

        ");
    }
}

currentRoom.name ~~ Marshmallow.roomName {
    print("In this room is your kingdom advisor ");
    print(Marshmallow.name);
    strPrintLine("!");
    strPrintLine("Do you want to talk to her?");
    getInpulFromOptions("Yes", "No");
}

currentRoom.name ~~ Marshmallow.roomName AND input ~~ "Yes" {
    print(Marshmallow.name);
    strPrintLine(" says: ");
    if (NOT spokeToMarshmallow) {
        strPrintLine(Marshmallow.message);
        spokeToMarshmallow = true;
    } else {
        if (stateOfKingdom < 5) {
            strPrintLine("There is 10% more crying in the kingdom today. An acceptable amount, I suppose.
        ");
        } else {
            strPrintLine("I think things are... fine?
        ");
        }
    }
}
NOT xyzPuzzleInProgress {
    printCurrentRoomInfo();
    getlnputAdjacentRooms(currentRoom);
} 

NOT xyzPuzzleInProgress AND NOT (input ~ Throne_Room.name) {  
  -> input 
} 

input ~ Throne_Room.name {  
  if (spokeToMarshmallow AND spokeToPickles) {  
    -> input 
  } else {  
    strPrintLine("Before heading to the throne room to make today's  
    BIG DECISIONS, you should consider talking to your advisors to the east and 
    west!");  
  } 
} 

currentRoom == Throne_Room {  
  print(requestTitleCard[week]);  
  handleSubjects = true;  
} 

handleSubjects {  
  int subjectID = arbitrarySubjectID();  
  print(catNames[subjectID]);  
  strPrintLine(" says: ");  
  strPrintLine(catMessages[subjectID]);  
  print("1. ");  
  print(goodResponses[subjectID]);  
  print("2. ");  
  strPrintLine(evilResponses[subjectID]);  
  
  getInputFromOptions("1", "2");  
  if (input == "1") {  
    strPrintLine(goodResults[subjectID]);  
    stateOfKingdom = stateOfKingdom + 1;  
    cheeseCubeCount = cheeseCubeCount - 1;  
  } else {  
    strPrintLine(badResults[subjectID]);  
    stateOfKingdom = stateOfKingdom - 1;  
    cheeseCubeCount = cheeseCubeCount + 1;  
  
  
  
  }
->Great_Hall

week = week + 1;
handleSubjects = false;
rollOverWeek = true;
}
currentRoom == Kitchen AND NOT xyzPuzzleInProgress {
    strPrintLine("You see three bowls of cat food, one labeled "X", one labeled "Y", and one labeled "Z".");
    xyzPuzzleInProgress = true;
}
xyzPuzzleInProgress {
    strPrintLine("Which bowl do you want to eat from?");
    getInputFromOptions("X", "Y", "Z", "None");
}
currentRoom == Kitchen AND xyzPuzzleInProgress AND input == "None" {
    xyzState = 1;
    xyzPuzzleInProgress = false;
}
xyzPuzzleInProgress AND input == "X" {
    if (bowlXAmount > 0) {
        bowlXAmount = bowlXAmount - 1;
        print("Bowl X has "); print(bowlXAmount); print(" foods left.

");
        xyzState = 2;
    } else {
        strPrintLine("There's no food left in this bowl.");
    }
}
xyzPuzzleInProgress AND input == "Y" {
    if (bowlYAmount > 0) {
        bowlYAmount = bowlYAmount - 1;
        print("Bowl Y has "); print(bowlYAmount); print(" foods left.

");
}
if (xyzState == 2) {
    xyzState = 3;
} else {
    if (xyzState == 5) {
        xyzState = 6;
    } else {
        xyzState = 1;
    }
} else {
    strPrintLine("There's no food left in this bowl.");
}

xyzPuzzleInProgress AND input == "Z" {
    if (bowlZAmount > 0) {
        bowlZAmount = bowlZAmount - 1;
        print("Bowl Z has "); print(bowlZAmount); print(" foods left.

    if (xyzState == 3) {
        xyzState = 4;
    } else {
        if (xyzState == 4) {
            xyzState = 5;
        } else {
            xyzState = 1;
        }
    } else {
        strPrintLine("There's no food left in this bowl.");
    }
}

xyzPuzzleInProgress AND xyzState == 6 {
    strPrintLine("YOU FOUND A SECRET TRAP DOOR!!! In it you find a stash of 9,999 cheese cubes!!");
    cheeseCubeCount = cheeseCubeCount + 9999;
    xyzState = 1;
}
week > lengthSabbatical { 
    strPrintLine("THE TRUE MONARCH HAS RETURNED FROM THE DIVINE 
SUNBEAM!\nYour three weeks are complete, and the rightful ruler has 
returned!\nLet us see how you did!\nProcessing... BEEP... BEEP BOP..."); 
    if (stateOfKingdom < 5) { 
        strPrintLine("Everyone is dead or dying! What a complete terror 
world you've made!"); 
    } else {} 
    if (stateOfKingdom >= 5) { 
        strPrintLine("Everyone seems super happy, for now!"); 
    } else {} 
    
    if (cheeseCubeCount < 5) { 
        strPrintLine("Wow, you also gave away all of the monarch's 
cheese money! The monarchy is done with, I guess!"); 
    } else {} 
    if (cheeseCubeCount >= 5) { 
        strPrintLine("You've also made an unspendably huge fortune!"); 
    } else {} 
    strPrintLine("I think that about wraps it up!"); 
    strPrintLine("Take care of yourself today!"); 
    strPrintLine("Thank you for playing, goodbye forever!"); 
    
    endgame; 
} 

func void printCurrentRoomInfo() { 
    print("You're in the "); 
    print(currentRoom.name); 
    strPrintLine(".\n"); 
} 

func int arbitrarySubjectID() { 
    if (week == 1) { return 3; } else {} 
    if (week == 2) { return 4; } else {} 
    if (week == 3) { return 5; } else {} 
    return neg 1; 
}
test_game_cat_kingdom_input.out:

King Tubbs, the great monarch of the Cat Kingdom, has recently discovered a sunbeam of sensational quality!
He has, understandably, requested a 3 week sabbatical.
It is up to you (as an Official Visiting Noblecat) to lead the kingdom during that time.
Speak to your advisors and then make some important decisions!
At the end of seven weeks you can see exactly what sort of ruler you have been.
THE CAT MONARCHY AWAITS YOUR STEADY LEADERSHIP!

----------------------- WEEK 1 -----------------------

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber     West_Chamber     Throne_Room
Input: East_Chamber

In this room is your kingdom advisor Lady Marshmallow!
Do you want to talk to her?
Choose from one of the following options:
Yes     No
Input: No

You're in the East_Chamber.

Choose from one of the following options:
Great_Hall
Input: Great_Hall

You're in the Great_Hall.

Choose from one of the following options:
Ah, here is your financial advisor Duke Pickles!
Do you want to talk to him?
Choose from one of the following options:
Yes     No
Input: Yes

Duke Pickles says:
Your greatness, welcome to the treasury. I am your financial advisor.
I stand here amongst our hoard of cheese chunks to determine the general "cheesiness" of our monarchy.
Feel free to drop in any time and ask how things are going, yes.

You're in the West_Chamber.

Choose from one of the following options:
Kitchen     Great_Hall
Input: Great_Hall

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber     West_Chamber     Throne_Room
Input: West_Chamber

Ah, here is your financial advisor Duke Pickles!
Do you want to talk to him?
Choose from one of the following options:
Yes     No
Input: Yes

Duke Pickles says:
We have 5 cheese cubes!

You're in the West_Chamber.
Choose from one of the following options:
Kitchen     Great_Hall
Input: Great_Hall

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber     West_Chamber     Throne_Room
Input: East_Chamber

In this room is your kingdom advisor Lady Marshmallow!
Do you want to talk to her?
Choose from one of the following options:
Yes     No
Input: Yes

Lady Marshmallow says:
Hello, my new and temporary liege. I am your kingdom advisor.
It is my job to advise you on the health and happiness of your kingdom.
My own health and happiness is irrelevant.
Please, see me again some time.

You're in the East_Chamber.

Choose from one of the following options:
Great_Hall
Input: Great_Hall

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber     West_Chamber     Throne_Room
Input: Throne_Room

YOUR CAT SUBJECTS HAVE SOME IMPORTANT REQUESTS!

Pumpkin says:
A cat baby is lost in the spooky forest!
I know because it sent me this baby note, via forest squirrel!

1. I will organize a search party!
2. I will organize a snake party!

Choose from one of the following options:
1 2
Input: 2

You party with some snakes. What a night! The baby is never heard from again.

----------------------- WEEK 2 -----------------------

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber West_Chamber Throne_Room
Input: East_Chamber

In this room is your kingdom advisor Lady Marshmallow!
Do you want to talk to her?
Choose from one of the following options:
Yes No
Input: Yes

Lady Marshmallow says:
There is 10% more crying in the kingdom today. An acceptable amount, I suppose.

You're in the East_Chamber.

Choose from one of the following options:
Great_Hall
Input: Great_Hall

You're in the Great_Hall.
Choose from one of the following options:
East_Chamber    West_Chamber    Throne_Room
Input: West_Chamber

Ah, here is your financial advisor Duke Pickles!
Do you want to talk to him?
Choose from one of the following options:
Yes    No
Input: Yes

Duke Pickles says:
We have 6 cheese cubes!

You're in the West_Chamber.

Choose from one of the following options:
Kitchen    Great_Hall
Input: Kitchen

You see three bowls of cat food, one labeled "X", one labeled "Y", and one labeled "Z".
Which bowl do you want to eat from?
Choose from one of the following options:
X    Y    Z    None
Input: Y

Bowl Y has 6 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X    Y    Z    None
Input: Z

Bowl Z has 7 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X    Y    Z    None
Input: Z

Bowl Z has 6 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X     Y     Z     None
Input: X

Bowl X has 4 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X     Y     Z     None
Input: Y

Bowl Y has 5 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X     Y     Z     None
Input: Z

Bowl Z has 5 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X     Y     Z     None
Input: Z

Bowl Z has 4 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X     Y     Z     None
Input: Y
Bowl Y has 4 foods left.

YOU FOUND A SECRET TRAP DOOR!! In it you find a stash of 9,999 cheese cubes!!
Which bowl do you want to eat from?
Choose from one of the following options:
X    Y    Z    None
Input: X

Bowl X has 3 foods left.

Which bowl do you want to eat from?
Choose from one of the following options:
X    Y    Z    None
Input: None

You're in the Kitchen.

Choose from one of the following options:
West_Chamber
Input: West_Chamber

Ah, here is your financial advisor Duke Pickles!
Do you want to talk to him?
Choose from one of the following options:
Yes    No
Input: Yes

Duke Pickles says:
We have 10005 cheese cubes!

You're in the West_Chamber.

Choose from one of the following options:
Kitchen    Great_Hall
Input: Great_Hall

You're in the Great_Hall.
Choose from one of the following options:
East_Chamber  West_Chamber  Throne_Room
Input: Throne_Room

GET READY FOR EVEN MORE REQUESTS FROM CAT SUBJECTS!

Snowball says:
Hel... hello monarch!
I'm...
I'm very lonely.
Would you mind if I just stood in here for a few minutes?
I'm sorry. I can leave.

1. Please, stay!
2. LEAVE AT ONCE.

Choose from one of the following options:
1  2
Input: 1

The cat leaves. You eat a royal pizza bagel.

---------------------- WEEK 3 ----------------------

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber  West_Chamber  Throne_Room
Input: East_Chamber

In this room is your kingdom advisor Lady Marshmallow!
Do you want to talk to her?
Choose from one of the following options:
Yes  No
Input: Yes

Lady Marshmallow says:
I think things are... fine?

You're in the East_Chamber.

Choose from one of the following options:
Great_Hall
Input: Great_Hall

You're in the Great_Hall.

Choose from one of the following options:
East_Chamber  West_Chamber  Throne_Room
Input: Throne_Room

THE FINAL REQUESTS BEFORE YOUR REIGN IS AT AN END!

Patches says:
Help! It's my son!
He is very sick!
And stuck in a huge bear mouth!

1. Guards, help this cat's son!
2. Guards, help the bear eat this cat's son!

Choose from one of the following options:
1  2
Input: 2

Your guards lose a little bit more of themselves.

THE TRUE MONARCH HAS RETURNED FROM THE DIVINE SUNBEAM!
Your three weeks are complete, and the rightful ruler has returned!
Let us see how you did!
Processing... BEEP... BEEP BOP...
Everyone is dead or dying! What a complete terror world you've made!
You've also made an unspendably huge fortune!
I think that about wraps it up!
Take care of yourself today!
Thank you for playing, goodbye forever!
test_game_go_outside_input.in:

LivingRoom
Outside

test_game_go_outside_input.tbag:

/* Authors: Greg, Julie */

room {} room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

room LivingRoom {
    name = "Living Room";
}

room Outside {
    name = "Outside";
}

Closet <-> Bedroom;
Bedroom <-> LivingRoom;

start { Bedroom }

boolean madeItOutside = false;

currentRoom == Bedroom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    getInputFromOptions("Closet", "LivingRoom");
    -input
    print("\n");
}
currentRoom == Closet {
    string testName = currentRoom.name;
    print("Currently in: ");
    print(testName);
    print("\n");
    getInputFromOptions("Bedroom");
    ->input
    print("\n");
}

currentRoom == LivingRoom {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    getInputFromOptions("Bedroom", "Outside");
    ->input
    print("\n");
}

currentRoom == Outside {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
    print("\n");
    madeItOutside = true;
}

madeItOutside {
    print("Good job making it outside, lazybones.");
    print("\n");
    endgame;
}

test_game_go_outside_input.out:

Currently in: Bedroom
Choose from one of the following options:
   Closet     LivingRoom
Input: LivingRoom
Currently in: Living Room
Choose from one of the following options:
Bedroom     Outside
Input: Outside

Currently in: Outside

Good job making it outside, lazybones.

test_game_hangman_input.in:

M
D
T
I
E
H
B
C
R
F
G
K
test_game_hangman_input.tbag:

/* Authors: Julie */
#import stdlib
#import typeConversionLib

int guesses = 0;
int wrongGuesses = 0;
int guessesAllowed = 6;

boolean started = false;
string[10] answer;
boolean[26] guessed;

boolean updateAndDisplay = false;
boolean foundAll;

157
NOT started {
    answer[0] = "H";
    answer[1] = "I";
    answer[2] = "T";
    answer[3] = "C";
    answer[4] = "H";
    answer[5] = "H";
    answer[6] = "I";
    answer[7] = "K";
    answer[8] = "E";
    answer[9] = "R";

    started = true;
}

true {
    print("Guess a letter.");
}

letterIsInAnswer(input) {
    updateAndDisplay = true;
}

NOT letterIsInAnswer(input) {
    wrongGuesses = wrongGuesses + 1;
    updateAndDisplay = true;
}

updateAndDisplay {
    guessed[intFromLetter(input)] = true;
    guesses = guesses + 1;
    printHangMan(wrongGuesses);
    foundAll = checkAndPrintStatus();
    updateAndDisplay = false;
}

158
foundAll {
    strPrintLine("WOO!!!! YOU WON!!!!!!!!");
    endgame;
}

wrongGuesses >= guessesAllowed {
    strPrintLine("You lost, how embarrassing!!!");
    endgame;
}

func boolean letterIsInAnswer(string letter) {
    int i = 0;
    boolean found = false;
    while (i < arrLen(answer) AND found == false) {
        if (answer[i] == letter) {
            found = true;
        } else {}
        i = i + 1;
    }
    return found;
}

func void printHangMan(int wg) {
    print("\n");
    print("\n");
    print("\n");
    if (wg == 0) {
        print( " _______   \n" );
        print( "|         |   \n" );
        print( "|         O  \n" );
        print( "|            \n" );
        print( "|            \n" );
        print( "|            \n" );
    } else {}  
    if (wg == 1) {
        print( " _______   \n" );
        print( "|         |   \n" );
        print( "|       0    \n" );
        print( "|            \n" );
        print( "|            \n" );
        print( "|            \n" );
    } else {}
if (wg == 2) {
    print(" _ _ _ _ _ _ _ _ _");
    print(" |          |");
    print(" |          O ");
    print(" |         /  ");
    print(" |               ");
    print(" |               ");
    print(" |               ");
}
else {}

if (wg == 3) {
    print(" _ _ _ _ _ _ _ _ _");
    print(" |          |");
    print(" |          O ");
    print(" |         / | ");
    print(" |               ");
    print(" |               ");
    print(" |               ");
}
else {}

if (wg == 4) {
    print(" _ _ _ _ _ _ _ _ _");
    print(" |          |");
    print(" |          O ");
    print(" |         / | |");
    print(" |               ");
    print(" |               ");
    print(" |               ");
}
else {}

if (wg == 5) {
    print(" _ _ _ _ _ _ _ _ _");
    print(" |          |");
    print(" |          O ");
    print(" |         / | |");
    print(" |         /    ");
    print(" |               ");
    print(" |               ");
}
if (wg == 6) {
    print(" _ _ _ _ _ _ _ _ _
")
    print(" |          |
")
    print(" |          O 
")
    print(" |         / | \
")
    print(" |         /  \
")
    print(" |               
")
    print(" |               
");
} else {
    print("\n");
    print("\n");
    print("\n");
}

func void printGuessedLetters() {
    int i = 0;
    print("Already Guessed: ");
    while (i < arrLen(guessed)) {
        if (guessed[i] == true) {
            print(letterFromInt(i));
            print(" ");
        } else {
            i = i + 1;
        }
    }
}

func boolean checkAndPrintStatus() {
    int i = 0;
    boolean foundall = true;
    while (i < arrLen(answer)) {
        if (guessed[intFromLetter(answer[i])]) {
            print(answer[i]);
            print(" ");
        } else {
            foundall = false;
            print("_");
            print(" ");
        }
    }
}

}
    i = i + 1;
}

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

printGuessedLetters();

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

return foundall;
}


test_game_hangman_input.out:

Guess a letter. Choose from one of the following options:
A     B     C     D     E     F     G     H     I     J     K     L     M     N     O
P     Q     R     S     T     U     V     W     X     Y     Z
Input: M

_________
|       |
|   0   |
|       |
|       |
|       |

- - - - - - - -
Already Guessed : M

162
Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: D

_______
|      |
|      O
|       /
|  
|   
|

- - - - - - - -

Already Guessed : D M

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: T

_______
|      |
|      O
|       /
|  
|   
|
_ _ T _ _ _ _ _ _

Already Guessed: D M T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: I

________
|   0   |
|      /|
|   O   |

_ _ _ _ _ _ _ _ 

|          |
|          |

_ I T _ _ _ I _ _

Already Guessed: D I M T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: E

________
|   0   |
|      O|

164
Already Guessed: D E I M T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: H

Already Guessed: D E H I M T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: B
H I T _ H H I _ E _

Already Guessed: B D E H I M T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Input: C

H I T C H H I _ E _

Already Guessed: B C D E H I M T

Guess a letter. Choose from one of the following options:
Input: R

________
|   |
|   |
|   |
|   |
|   |

H I T C H H I _ E R

Already Guessed: B C D E H I M R T

Guess a letter. Choose from one of the following options:

A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z

Input: F

________
|   |
|   |
|   |
|   |
|   |

H I T C H H I _ E R

167
Already Guessed: B C D E F H I M R T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: G

_________
|   |   |
|   0   |
| /\   |
|   /   |
|   |

H I T C H I _ E R

Already Guessed: B C D E F G H I M R T

Guess a letter. Choose from one of the following options:
A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z
Input: K

_________
|   |   |
|   0   |
| /\   |
|   /   |
|   |

168
H I T C H I K E R

Already Guessed : B C D E F G H I K M R T

WOO!!!! YOU WON!!!!!!!

test_game_mouse_cat_input.in:

Wall
Kitchen
Wall
Kitchen
Bedroom
test_game_mouse_cat_input.tbag:

/* Authors: Julie */
#import stdlib
#import typeConversionLib

room {}

room Closet {
    name = "Closet";
}

room Bedroom {
    name = "Bedroom";
}

room Wall {
    name = "Wall";
}

room Kitchen {
    name = "Kitchen";
}

Closet <-> Bedroom;
Closest Wall;
Kitchen Wall;
Kitchen Bedroom;

start { Closet }

npc {
    string roomName;
}
npc Cat {
    roomName = "Bedroom";
}
boolean started = false;

NOT started {
    strPrintLine("You're a mouse.");
    started = true;
}

ture {
    printCurrentRoomInfo();
    getInputAdjacentRooms(currentRoom);
    ->input
}
currentRoom.name == Cat.roomName {
    print("you got eaten by the cat.\n");
    endgame;
}

func void printCurrentRoomInfo() {
    print("Currently in: ");
    print(currentRoom.name);
    print("\n");
}

test_game_mouse_cat_input.out:

You're a mouse.
Currently in: Closet
Choose from one of the following options:
Wall      Bedroom
Input: Wall

Currently in: Wall
Choose from one of the following options:
Kitchen    Closet
Input: Kitchen

Currently in: Kitchen
Choose from one of the following options:
Bedroom    Wall
Input: Wall

Currently in: Wall
Choose from one of the following options:
Kitchen    Closet
Input: Kitchen

Currently in: Kitchen
Choose from one of the following options:
Bedroom    Wall
Input: Bedroom

you got eaten by the cat.

test_gcd_func.tbag:

/* Authors: Julie */
true {
    print(gcd(2,14));
    print("\n");
    print(gcd(3,15));
    print("\n");
    print(gcd(99,121));
    print("\n");
    endgame;
}
```c
func int gcd(int a, int b) {
    while (a != b) {
        if (a > b) { a = a - b; }
        else { b = b - a; }
    }
    return a;
}

test_gcd_func.out:

2
3
11

test_gcd_func2.tbag:

/* Authors: Julie */
true {
    print(gcd(14,21));
    print("\n");
    print(gcd(8,36));
    print("\n");
    print(gcd(99,121));
    print("\n");
    endgame;
}

func int gcd(int a, int b) {
    while (a != b) {
        if (a > b) { a = a - b; }
        else { b = b - a; }
    }
    return a;
}

test_gcd_func2.out:

7
4
11
```
test_gcd_handler1.tbag:

/* Authors: Julie */
int a = 36;
int b = 8;

a == b {
    print(a);
    endgame;
}

a > b {
    a = a - b;
}

a < b {
    b = b - a;
}

test_gcd_handler1.out:

4

test_gcd_handler2.tbag:

/* Authors: Julie */
int a = 8;
int b = 36;

a == b {
    print(a);
    endgame;
}

a > b {
    a = a - b;
}

a < b {
\[
b = b - a;
\]

```c
int a = 2;
int b = 14;

a == b {
    print(a);
    endgame;
}

a > b {
    a = a - b;
}

a < b {
    b = b - a;
}
```

test_gcd_handler3.out:

2
test_gcd_handler4.tbag:

```c
/* Authors: Julie */
int a = 99;
int b = 121;

a == b {
    print(a);
    endgame;
}```
a > b {
    a = a - b;
}

a < b {
    b = b - a;
}

test_gcd_handler4.out:
11
test_global_array_in_handler.tbag:

/* Authors: Julie */
int[1] a1;
int[2] a2;

true {
    a1[0] = 0;
a2[0] = 0;
a2[1] = 1;
    print(a1[0]);
    print(a2[0]);
    print(a2[1]);
    endgame;
}

test_global_array_in_handler.out:
001
test_global_var_func.tbag:

/* Authors: Julie */
int a = 5;

true {
    testvar();
    endgame;
}
func int testvar() {
    print(a);
    return 0;
}

test_global_var_func.out:

5
test_global_var_handler.tbag:

/* Authors: Julie */
int a = 5;
true {
    print(a);
    endgame;
}

test_global_var_handler.out:

5
test_handler1.tbag:

/* Authors: Julie */
boolean pred1 = false;
boolean pred2 = false;
int a = 0;
int b = 0;
true {
    a = 666;
    pred1 = true;
}
pred2 {
    print(a + b);
    endgame;
}
pred1 {
    b = 3;
    pred2 = true;
test_handler1.out:

669
test_handler2.tbag:

/* Authors: Julie */
boolean pred1 = false;
boolean pred2 = true;
int int3 = 4;
int int4 = 9;

pred1 {
    print("four");
    pred1 = false;
    endgame;
}

NOT pred2 {
    print("two");
    int4 = int4 + 5;
    pred2 = true;
}

int3 < 6 {
    print(1);
    pred2 = false;
    int3 = 7;
}

int4 >= 10 {
    print(3);
    pred1 = true;
    int4 = 9;
}

test_handler2.out:
1two3four
test_helloworld.tbag:

/* Authors: Julie */
true {
    print("hello world\n");
    endgame;
}

test_helloworld.out:

hello world

test_helloworld_func.tbag:

/* Authors: Julie */
true {
    hello();
    endgame;
}

func int hello() {
    print("hello world function");
    return 0;
}

test_helloworld_func.out:

hello world function

test_if_func.tbag:

/* Authors: Julie */
true {
    testif();
    endgame;
}

func int testif() {
    if (true) {
        print("true ");
    }

else {
  print("hi");
  return 0;
}

test_if_func.out:

true hi

test_if_func2.tbag:

/* Authors: Julie */
true {
  testif();
  endgame;
}

func int testif() {
  if (true) {
    print(666);
  }
  else {
    print(" lol");
  }
  print("hi");
  return 0;
}

test_if_func2.out:

666hi

test_if_func3.tbag:

/* Authors: Julie */
true {
  testif();
  endgame;
}

func int testif() {
  if (false) {
    print("true ");
    print("false ");
  }
    print("true ");
    print("false ");
  }
  print("hi");
  return 0;
}

test_if_func3.out:

true false

test_if_func4.tbag:

/* Authors: Julie */
true {
  testif();
  endgame;
}

func int testif() {
  if (false) {
    print("true ");
    print("false ");
  }
  print("hi");
  return 0;
}
else {}
    print("hi");
    return 0;
}

test_if_func3.out:

hi

test_if_func4.tbag:

/* Authors: Julie */
true {
    testif();
    endgame;
}

func int testif() {
    if (false) {
        print(666);
    } else {
        print(" lol");
    }
    print("hi");
    return 0;
}

test_if_func4.out:

lolhi

test_if_handler3.tbag:

/* Authors: Julie */

true {
    if (true) {
        print(666);
    } else {
        print(" lol");
    }
}
print("hi");
endgame;
}

test_if_handler3.out:

666hi

test_local_var_func.tbag:

/* Authors: Julie */
true {
    testvar();
    endgame;
}

func int testvar() {
    int a = 5;
    print(a);
    return 0;
}

test_local_var_func.out:

5

test_local_var_handler.tbag:

/* Authors: Julie */
true {
    int a = 5;
    print(a);
    endgame;
}

test_local_var_handler.out:

5

test_loop_event.tbag:

/* Authors: Julie */
int i = 0;
i < 5 {
    print(i);
    i = i + 1;
}

i >= 5 {
    print(666);
    endgame;
}

test_loop_event.out:
01234666
test_loop_while_func.tbag:
/* Authors: Julie */
int i = 0;

true {
    whileTest();
    endgame;
}

func void whileTest() {
    while (i < 5) {
        print(i);
        print("\n");
        i = i + 1;
    }
    print(666);
}

test_loop_while_func.out:
0
1
2
3
4
666
test_loop_while_handler.tbag:

/* Authors: Julie */
int i = 0;

true {
    while (i < 5) {
        print(i);
        print("\n");
        i = i + 1;
    }
    print(666);
    endgame;
}

test_loop_while_handler.out:

0
1
2
3
4
666
test_ops.tbag:

/* Authors: Julie */
#import stdlib

true {
    intPrintLine(1 + 2);
    intPrintLine(1 - 2);
    intPrintLine(1 * 2);
    intPrintLine(100 / 2);
    intPrintLine(99);
    boolPrintLine(1 == 2);
    boolPrintLine(1 == 1);
    intPrintLine(99);
    boolPrintLine(1 != 2);
    boolPrintLine(1 != 1);
    intPrintLine(99);
boolPrintLine(1 < 2);
boolPrintLine(2 < 1);
intPrintLine(99);
boolPrintLine(1 <= 2);
boolPrintLine(1 <= 1);
boolPrintLine(2 <= 1);
intPrintLine(99);
boolPrintLine(1 > 2);
boolPrintLine(2 > 1);
intPrintLine(99);
boolPrintLine(1 >= 2);
boolPrintLine(1 >= 1);
boolPrintLine(2 >= 1);
endgame;
}

test_ops.out:

3
-1
2
50
99
false
true
99
true
false
99
true
false
99
true
false
99
false
true
99
false
true
true
true
test_room_data_w_blank_room_decl.tbag:

/* Authors: Julie */
room {}

room myRoom {
    name = "living room";
}

room myRoom1 {
    name = "kitchen";
}

myRoom <-> myRoom1;

start {myRoom}

ttrue {
    print("hi\n");
    print(myRoom.name);
    print("\n");
    endgame;
}

test_room_data_w_blank_room_decl.out:

hi
living room

test_stdlib.tbag:

/* Authors: Julie */
#import stdlib

ttrue {
    strPrintLine("hi");
    intPrintLine(666);
    boolPrintLine(true);
    boolPrintLine(false);
    endgame;
}
testStdStringOut:

hi
666
true
false
testStringLiteralsTbag:

/* Authors: Julie */
true {
    string a = "single quo'te";
    string newline = "\n";
    string c = "tab\ttab";
    string d = "backspac\be";
    string e = "carriage r\return";
    string f = "formfeed\fformfeed";
    string g = "escaped \"double quote\"";
    string h = "backslas\h";

    print(a);
    print(newline);
    print(c);
    print(newline);
    print(d);
    print(newline);
    print(e);
    print(newline);
    print(f);
    print(newline);
    print(g);
    print(newline);
    print(h);
    print(newline);
endgame;
}
testStringLiteralsOut:
single quo'te
test_subtract.tbag:

/* Authors: Julie */
true {
    print(69-7);
    endgame;
}

test_subtract.out:

62

9.2 Full Git Commit History

2950f75Maria van Keulen Tue Dec 22 20:39:14 2015 -0500 Added Iris's test author annotations
2840ddeMaria van Keulen Tue Dec 22 20:24:49 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
5905e4aMaria van Keulen Tue Dec 22 20:24:25 2015 -0500 Added test author annotations
7725c34gregorychen3 Tue Dec 22 20:17:50 2015 -0500 authors added to Makefile
70845aeMaria van Keulen Tue Dec 22 19:41:01 2015 -0500 Added authors this project never ends
e1179e8Maria van Keulen Tue Dec 22 17:18:50 2015 -0500 Last commit EVAR
039bf8bjj-ian Tue Dec 22 04:52:39 2015 -0500 Merge pull request #43 from jj-ian/jj-ian/tests
1ab3f85Julie Tue Dec 22 04:44:10 2015 -0500 updated preprocessing script so imports can go at top of file, updated tests to match, updated tests to account for java driver formatting changes
c6e8abcJulie Tue Dec 22 03:48:26 2015 -0500 finished cat kingdom demo
28dc283Julie Tue Dec 22 03:36:20 2015 -0500 deleted backup
8954afbJulie Tue Dec 22 03:35:36 2015 -0500 finished cat kingdom demo
a942df4Julie Tue Dec 22 02:34:37 2015 -0500 still working on cat kingdom demo
4e77df7Julie Mon Dec 21 18:25:31 2015 -0500 added more cat subjects data to cat kingdom demo
4fd8af2Julie Mon Dec 21 18:12:37 2015 -0500 working on cat kingdom demo, updated .gitignore to ignore Julie's scripts
Julie

Fixed item npc checking

Sun Dec 20 20:33:12 2015 -0500  started cat kingdom game
5ca5ddbdJulie  Tue Dec 22 03:48:26 2015 -0500 finished cat kingdom demo
b191cc0Julie  Tue Dec 22 03:36:20 2015 -0500 deleted backup
2ac5c8b4Julie  Tue Dec 22 03:35:36 2015 -0500 finished cat kingdom demo
699fe07Julie  Tue Dec 22 02:34:37 2015 -0500 still working on cat kingdom demo

73332cMaria van Keulen  Mon Dec 21 18:34:35 2015 -0500  Fixed
check_matching_dcls_helper, renamed func
8364b64Julie  Mon Dec 21 18:25:31 2015 -0500 added more cat subjects data to cat kingdom demo
0b92623Julie  Mon Dec 21 18:12:37 2015 -0500 working on cat kingdom demo, updated
.gitignore to ignore Julie's scripts
31a349cJulie  Mon Dec 21 13:56:29 2015 -0500 cat kingdom output, not valid yet
def58fcJulie  Mon Dec 21 03:07:13 2015 -0500 working on cat kingdom demo, made new
library, updated some tests
3eb502cJulie  Mon Dec 21 01:58:05 2015 -0500  started cat kingdom game
0ed6758mvankeulen94  Mon Dec 21 13:32:54 2015 -0500 Merge pull request #42 from jj-ian/scfix
aceb14fMaria van Keulen  Mon Dec 21 13:22:10 2015 -0500 Check GIFO num args, updated
usage msg in script
79bf83eMaria van Keulen  Mon Dec 21 10:24:22 2015 -0500 Final sem check fixes
2197cc3jj-ian Mon Dec 21 00:28:54 2015 -0500 Merge pull request #41 from jj-ian/jj-ian/tests
351b35cJulie  Mon Dec 21 00:21:34 2015 -0500 modified golden refs for some tests after i changed java library
ef96dfbJulie  Sun Dec 20 23:55:11 2015 -0500 Merge branch 'jj-ian/tests' of
https://github.com/jj-ian/tbag into jj-ian/tests
7fde1dfJulie  Sun Dec 20 23:52:33 2015 -0500 finished hangman test, modified java lib so options print on same line, updated stdlib.tbag
513aff3Julie  Sun Dec 20 23:52:33 2015 -0500 finished hangman test, modified java lib so options print on same line, updated stdlib.tbag
09ce71mvankeulen94  Sun Dec 20 23:01:00 2015 -0500 Merge pull request #40 from jj-ian/sem_check_fin
bd600a7Maria van Keulen  Sun Dec 20 22:54:03 2015 -0500 Added
item/npc/getinputfromadj checking
da6c970Maria van Keulen  Sun Dec 20 22:33:12 2015 -0500 Removed unnecessary comments
cd44d17Maria van Keulen  Sun Dec 20 22:27:24 2015 -0500 Finished type check updates
c9e3fdaMaria van Keulen  Sun Dec 20 21:37:35 2015 -0500 Fixed item/npc err checking
and more type checking
01a82b9Maria van Keulen  Sun Dec 20 21:15:31 2015 -0500 More type check fixes
2ce474fMaria van Keulen  Sun Dec 20 20:48:33 2015 -0500 Added type checking funcs
23a7590Maria van Keulen  Sun Dec 20 20:22:15 2015 -0500 check num args for getadj function
6222486jj-ian Sun Dec 20 22:01:11 2015 -0500 Merge pull request #39 from jj-ian/jj-ian/tests
fc7dce6Julie  Sun Dec 20 22:00:14 2015 -0500 updated library import script to be compatible w python 2 and 3

188
34d378cJulie Sun Dec 20 21:29:43 2015 -0500 fixed a few things in stdlib and scripts, still working on hangman game
a3e6b2b2jj-ian Sun Dec 20 20:34:35 2015 -0500 Merge pull request #38 from jj-ian/jj-ian/tests
042eb5aJulie Sun Dec 20 20:33:23 2015 -0500 hooked up #import to compiler and tests
script, updated tests to match
ed1ca88jj-ian Sun Dec 20 19:54:57 2015 -0500 Merge pull request #37 from jj-ian/jj-ian/tests
cb12b8fJulie Sun Dec 20 19:35:47 2015 -0500 done w/ import library script, testing it on various programs
a063f6bJulie Sun Dec 20 18:42:50 2015 -0500 working on #import, new test for functions and stdlib
984f96fJulie Sun Dec 20 02:24:14 2015 -0500 hangman
6cfa8ddJulie Sun Dec 20 02:09:34 2015 -0500 updated hangman test
b302126mvankeulen94 Sun Dec 20 19:37:52 2015 -0500 Merge pull request #36 from jj-ian/sem_check_f
ejj-ian/sem_check_f
7e0b200Maria van Keulen Sun Dec 20 19:13:47 2015 -0500 Fixed out file after rebase
0f0742aMaria van Keulen Sun Dec 20 17:54:14 2015 -0500 Updated fail tests to reflect alpha renaming
9afdae7Maria van Keulen Sun Dec 20 17:40:55 2015 -0500 Updated getInputFromOptions tests
7f009f0Maria van Keulen Sun Dec 20 17:20:36 2015 -0500 Added wrapper function for find_variable and tests
f3e4d3aMaria van Keulen Sun Dec 20 15:42:53 2015 -0500 Fixed rdecl bug and added more tests
c52a7a4Maria van Keulen Sun Dec 20 15:14:23 2015 -0500 Fixed bug in assigns, added more fail tests
93ed1e9Maria van Keulen Sun Dec 20 13:36:03 2015 -0500 Added fail tests, updated script to rm temp files
9d77d01bsslakter Sun Dec 20 18:56:23 2015 -0500 will they pass, lord i hope
b58a7eJulie Sun Dec 20 18:42:50 2015 -0500 working on #import, new test for functions and stdlib
bc32398Brian Slakter Sun Dec 20 17:47:03 2015 -0500 Merge pull request #34 from jj-ian/tests_passing
c66926bsslakter Sun Dec 20 17:17:34 2015 -0500 tests passing
04a5ebJulie Sun Dec 20 15:50:23 2015 -0500 Merge branch 'jj-ian/tests' of https://github.com/jj-ian/tbag into jj-ian/tests
41bb76eJulie Sun Dec 20 02:24:14 2015 -0500 hangman
8e03678Julie Sun Dec 20 02:09:34 2015 -0500 updated hangman test
7c16743Julie Sun Dec 20 15:42:04 2015 -0500 working on hangman test, adding to stdlib
6d76c64Iris Sun Dec 20 15:41:06 2015 -0500 adding getInputAdjacentRooms check
c92dfc6bslakter Sun Dec 20 15:17:27 2015 -0500 camel case, adjacency function
8797392Julie Sun Dec 20 13:41:35 2015 -0500 fixed bug in java library, updated tests to reflect compiler updates, updated hangman test
850395cIris Zhang Sun Dec 20 13:01:25 2015 -0500 Merge pull request #32 from jj-ian/sem_check_fail
Fix comment in parser that was throwing error

52ec962Iris Zhang Sat Dec 19 21:58:33 2015 -0500 Merge pull request #29 from jj-ian/sem_check_up

c598025Iris Sat Dec 19 21:19:33 2015 -0500 Add semchecking of items working with tests
d21d8beMaria van Keulen Sat Dec 19 21:09:24 2015 -0500 Fixed bug in array declaration, added tests

0167165Iris Sat Dec 19 20:59:00 2015 -0500 Semchecking NPCs working with tests

282d13cMaria van Keulen Sat Dec 19 20:24:48 2015 -0500 Checked not operand, checked recursive types
31922deMaria van Keulen    Sat Dec 19 19:47:03 2015 -0500    Added get_input_from_options
func checking
b43ee2cMaria van Keulen    Sat Dec 19 18:59:04 2015 -0500    Added arr_len semantic checking
c977814Iris    Sat Dec 19 18:53:24 2015 -0500    Added goto in stmt for semcheck... not sure if working with tests
45f157aIris    Sat Dec 19 17:13:42 2015 -0500    Semcheck Room Field access working with testgit add semantic_checker.ml
d37a547Iris    Sat Dec 19 17:11:02 2015 -0500    working on semchecking room field access
6f48d2aMaria van Keulen    Sat Dec 19 16:00:32 2015 -0500    Fixed find_room failure bug
b8f0487Maria van Keulen    Sat Dec 19 15:47:06 2015 -0500    Added special cases to eq/neq ops
f461fcfIris    Sat Dec 19 15:44:43 2015 -0500    Change return type of expr checking of rooms to void
2c27a07Maria van Keulen    Sat Dec 19 15:36:56 2015 -0500    Rmed unnecessary comments/functions
97e1e9aIris    Sat Dec 19 15:35:12 2015 -0500    Redo the way we sem check for Room == Room
8954914Maria van Keulen    Sat Dec 19 14:34:52 2015 -0500    Added sem_check back into build procedure
7e5836ccslakter    Sat Dec 19 17:48:08 2015 -0500    authors
a74bd7dbslakter    Sat Dec 19 17:46:12 2015 -0500    formatting
689d2bagregorychen3    Sat Dec 19 17:00:16 2015 -0500    Merge branch 'master' of https://github.com/jj-ian/tbag
d5a0096gregorychen3    Sat Dec 19 16:59:55 2015 -0500    code beautification (tabbing etc)
11e6644Julie    Sat Dec 19 16:58:33 2015 -0500    updated test script to include stdlib for all tbag files to test, updated stdlib, added more tests, updated tests to work w stdlib
6af66fcIris Zhang    Sat Dec 19 14:18:03 2015 -0500    Merge pull request #28 from jj-ian/sem_check_tests
d3e4000Iris    Sat Dec 19 14:14:21 2015 -0500    Adding one line that somehow got deleted in rebase
570494eMaria van Keulen    Sat Dec 19 11:08:21 2015 -0500    Added separate function finding for exprs/decls
6048ac4Maria van Keulen    Sat Dec 19 10:45:48 2015 -0500    Fixed bug in duplicate function checking
7549543Maria van Keulen    Sat Dec 19 10:24:01 2015 -0500    Added different print func for each type
b6b6494Maria van Keulen    Sat Dec 19 09:48:55 2015 -0500    In process of fixing find function mechanism
474866cMaria van Keulen    Sat Dec 19 08:46:47 2015 -0500    Moved stuff from symbol_table to translation_env
5913383Maria van Keulen    Sat Dec 19 08:29:02 2015 -0500    check_valid_type -> check return and var types
8147b7aIris    Sat Dec 19 02:29:09 2015 -0500    Add comment in sem_check for TODO for checking room names as valid expr
c04e1cIris    Sat Dec 19 02:22:32 2015 -0500    Add hacky fix for currentRoom in semcheck by adding it as a global variable
a864afdIris    Sat Dec 19 02:01:53 2015 -0500    Adj_decl sem_check working with test
378822cIris    Sat Dec 19 01:42:24 2015 -0500    Added adjacencies to sem_checker but not passing tests

191
Sat Dec 19 01:27:45 2015 -0500  room_decl body stmts now checking types, but not yet tested
2c300acIris Sat Dec 19 00:35:15 2015 -0500  ROOMDECL SEMANTIC CHECKING HALF WORKING~~
62bc4ceIris Fri Dec 18 23:31:15 2015 -0500  Room_decl added to sem_check compiling
4f155e6Maria van Keulen Fri Dec 18 23:03:37 2015 -0500  Rearranged binop checking
4b6f52aMaria van Keulen Fri Dec 18 21:48:02 2015 -0500  Added var existence fail tests
c38a127Maria van Keulen Fri Dec 18 21:31:37 2015 -0500  Updated run_tests to run fail tests
49a859fMaria van Keulen Fri Dec 18 21:30:16 2015 -0500  Rmd print statements, checked find_variable_in try
553cd73Iris Fri Dec 18 20:47:13 2015 -0500  working on room_decls sem check
b4c9cd9bIris Fri Dec 18 18:15:54 2015 -0500  added file with some functions to be used in semcheck
44acc26Maria van Keulen Fri Dec 18 16:10:17 2015 -0500  Updated function arg/param comparison
8814b97Iris Fri Dec 18 15:52:19 2015 -0500  Added recursive function check in semcheck
4e4ef2dIris Fri Dec 18 15:30:10 2015 -0500  Change room_def using list.fold_left to list.map
f4c6bd9Iris Fri Dec 18 15:24:20 2015 -0500  Fixed test_while_loop failure var decls before func decl checking
fa7b789Gregory Chen Sat Dec 19 13:58:39 2015 -0500  Merge pull request #27 from jj-ian/array_length
2dadfc0gregorychen3 Sat Dec 19 13:56:54 2015 -0500  tests included. they pass
52f80edgregorychen3 Sat Dec 19 13:56:07 2015 -0500  arr_len() implemented, new tests pass
43d8ba7jj-ian Fri Dec 18 18:00:54 2015 -0500  Merge pull request #26 from jj-ian/jj-ian/tests
cd43d1Julie Fri Dec 18 17:58:36 2015 -0500  Merge branch 'jj-ian/tests' of https://github.com/jj-ian/tbag into jj-ian/tests
1dbd575Julie Fri Dec 18 17:57:21 2015 -0500  more tests -- power set of npc, items, and room defs/decls
a6714beJulie Fri Dec 18 17:57:21 2015 -0500  more tests -- power set of npc, items, and room defs/decls
c813928gregorychen3 Fri Dec 18 17:28:47 2015 -0500  more array tests
fffff91gregorychen3 Fri Dec 18 17:19:43 2015 -0500  fixed codegen bug concerning array decls. array tests added.
2a6a313bslakter Fri Dec 18 16:17:40 2015 -0500  precedence and such
515222fbslakter Fri Dec 18 15:53:03 2015 -0500  tbag
80ed587bslakter Fri Dec 18 15:50:42 2015 -0500  updated precedence
ed0d14gregorychen3 Fri Dec 18 14:38:25 2015 -0500  tbag.ml modified to (temporarily) skip semantic check
dd7509fmvankeulen94 Fri Dec 18 14:25:00 2015 -0500  Merge pull request #25 from jj-ian/mvankeulen94/sem_check_updated
158d443Maria van Keulen Fri Dec 18 10:02:52 2015 -0500  script outputs compiler err
if no Driver.java
b6e48e0Maria van Keulen Fri Dec 18 10:02:03 2015 -0500  Added prelim checking for print function
1490edcMaria van Keulen Fri Dec 18 08:53:59 2015 -0500  java_tbag outputs compiler error messages if fail
Fixing semantic_checker after added new test, string literal

Added check_func_decl function

Added check internal checking

Renamed room_def semantic_checking

Added check_stmt

Fixed operator checking in check_expr

Added most of check_stmt function except Goto

Updated semantic checker, reamed argument type

Added updated version of semantic checker

Merge pull request #24 from jj-ian/jj-ian/tests

Merge branch 'jj-ian/tests' of https://github.com/jj-ian/tbag into jj-ian/tests

Updated string literal script to work w/ new compiler changes

Added new test, string literal

Updated string literal script to work w/ new compiler changes

Merge branch 'jj-ian/tests' of https://github.com/jj-ian/tbag into jj-ian/tests

Added new test, string literal

Finished modifying existing tests so they are successful with today's compiler changes

More tests fixed

Renamed room test

More tests fixed

Fixed four more tests

Merge branch 'master' of https://github.com/jj-ian/tbag

Some tests fixed

Added new test for a minimal program using rooms

Merge branch 'jj-ian/tests' of https://github.com/jj-ian/tbag into jj-ian/tests

Added new test, string literal

Stlif from julie

Pretty_printer.ml deleted
funcs now come at end of tbag file
rooms, npcs, and items all independently optional
code_gen, case sensitivity
npcs and items are optional now
added new test, string literal
all together now
Merge branch 'master' of
https://github.com/jj-ian/tbag
negative numbers
Merge branch 'master' of
https://github.com/jj-ian/tbag
funcs now come at end of tbag file
endgame implemented
Merge branch 'master' of
http://github.com/jj-ian/tbag
Merge pull request #23 from
jj-ian/jj-ian/tests
Merge branch 'jj-ian/tests' of
https://github.com/jj-ian/tbag into jj-ian/tests
more tests
more tests
Merge pull request #22 from
jj-ian/jj-ian/tests
fixed bug in input testing script, added
test for a small game that deals w/ inputs
A bunch of gcd tests
more tests, testing functions, handlers, loops
committing so i can pull
more tests, still working on
test_fib_event.tbag, that one isn't complete
deleted *.DS_Store, had added .DS_Store to
.gitignore in prev commit
added new tests - simple arithmetic
Updated test runner to
account for game IO
new test w/ simple print statement,
modified .gitignore to ignore .log and .diff files
Merge pull request #21 from
jj-ian/array_improvements
All assign and access can now be
used with expr for location.
array declarations now more robust.
can take int expr for size.
Gregory Chen

Merge pull request #20 from jj-ian/dot_field_access

added an extra dot-field notation

dot field notation now supported

added name field to room in go_outside game.

uncommented displayAdj() library function

2015-05-00 -0500

Gregory Chen

sun Mar 7 23:04:51 2015 -0500 Merge pull request #19 from jj-ian/disAdj_builtin_in_func

helloworld program tests.

6ca9291Gregory Chen

2015-05-00 -0500 Merge pull request #18 from jj-ian/jj-ian/java_target

cc39efJulie

Mon Dec 7 20:51:35 2015 -0500 compiled java files in java target

afac28cJulie

Mon Dec 7 20:40:36 2015 -0500 changed name of java target package from default to EventDrivenJavaTargetPackage

2df5586Julie

Mon Dec 7 20:33:14 2015 -0500 finished java target sample, refactored stuff to what generated code should look like, debugged sample, added mockup for cheat code support

5279729Julie

Sun Dec 6 23:53:27 2015 -0500 java target, working out some more control flow issues

cb977b2Julie

Sun Dec 6 23:21:50 2015 -0500 java target, working on unlocking behavior, added support for cheat code like things, refining how someone should write a tab program

e326c0Julie

Fri Dec 4 14:22:12 2015 -0500 java target sample, adding more handlers, dealing w/ room keys/locks

d4588aJulie

Fri Dec 4 14:12:11 2015 -0500 java target, still working on sample game with locks

9a2e35Julie

Fri Dec 4 13:55:23 2015 -0500 working on java target

95a1c7Julie

Mon Dec 7 15:31:06 2015 -0500 trivial changes for newline and indenting in code gen.

f3bfc70Gregory Chen

Mon Dec 7 15:27:17 2015 -0500 Merge pull request #17 from jj-ian/boilerplate_scanner
Gregory Chen

Iris Zhang  Preliminary predicate syntax implemented.

Julie Merge pull request #16 from Julie Thu Dec 3 21:29:17 2015 -0500Event driven java target, moved old java target to separate folder

Gregory Chen Work on semantic checker begin, vars and funcs

Maria van Keulen Still hacking away at semantic checker Wed Dec 2 13:56:53 2015 -0500Fix printed error for checking While stmt

Maria van Keulen Started work on statement checking Sat Nov 21 16:50:26 2015 -0500Fixed some sast refs, moved process_var_decl up

Maria van Keulen Added stmt check skeleton Mon Nov 30 00:13:15 2015 -0500Committed checked_body, added stmt check skeleton

Maria van Keulen Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Iris Zhang Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Jules Merge pull request #17 from Jules Thu Dec 3 22:03:52 2015 -0500Test script now compiles Driver.java

Maria van Keulen Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Iris Zhang Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Julie Merge pull request #16 from Julie Thu Dec 3 21:29:17 2015 -0500New event driven java target, moved old java target to separate folder

Gregory Chen Added boolean expressions woooo

Jules Merge pull request #17 from Jules Thu Dec 3 22:03:52 2015 -0500Test script now compiles Driver.java

Maria van Keulen Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Iris Zhang Merged branch 'izhang/sast' of https://github.com/jj-ian/tbag into izhang/sast

Julie Merge pull request #16 from Julie Thu Dec 3 21:29:17 2015 -0500New event driven java target, moved old java target to separate folder

Gregory Chen Added boolean expressions woooo
ad4d84cgregorychen3 Tue Dec 1 22:21:33 2015 -0500hello_world.tbag modified with testing for booleans. all works.
1c054f3Maria van Keulen Tue Dec 1 21:42:34 2015 -0500Compiles without errors!
24c0e09Gregory Chen Tue Dec 1 21:41:37 2015 -0500Merge pull request #12 from jj-ian/implment_booleans
0e51f00gregorychen3 Tue Dec 1 21:40:23 2015 -0500boolean data type implemented
f7e2c94Gregory Chen Tue Dec 1 14:10:29 2015 -0500Merge pull request #11 from jj-ian/regs
b817bb8gregorychen3 Tue Dec 1 14:06:51 2015 -0500now generating Npc.java and Item.java
618249agregorychen3 Tue Dec 1 12:05:38 2015 -0500trivial changes to revert to java target
2743f74Maria van Keulen Mon Nov 30 00:13:15 2015 -0500 Commted out checked_body, added stmt check skeleton
fb26948Maria van Keulen Sun Nov 29 22:50:28 2015 -0500 Fixed some sast refs, moved process_var_decl up
2d5d88acIris Sun Nov 29 21:16:20 2015 -0500 Working on more semantic analyzer, got it to compile until line 83.
03ce38dgregorychen3 Sun Nov 29 17:00:08 2015 -0500 started file code_gen.ml
7cca9a9bslakter Sun Nov 29 13:55:07 2015 -0500 new adj structure
1a6e3b1bslakter Sun Nov 29 13:39:49 2015 -0500 cast up
782b2fIris Wed Nov 25 19:14:45 2015 -0500 Working on semantic analyzer lots of syntax errors
c03a227gregorychen3 Wed Nov 25 12:47:37 2015 -0500 whoops left a few renamed files behind
1c150b3gregorychen3 Wed Nov 25 12:47:10 2015 -0500 renamed files, all compile and hello world works
cde8442gregorychen3 Wed Nov 25 12:33:36 2015 -0500 renamed jast to c_ast
f74d041bslakter Mon Nov 23 15:50:36 2015 -0500 can define room properties
2a6181fbslakter Mon Nov 23 12:57:36 2015 -0500 now can set adjacencies wooord
b38bf2bslakter Mon Nov 23 12:16:01 2015 -0500 using funcs already built, added constructor, can now instantiate rooms in driver
6d6bf4Gregory Chen Sun Nov 22 21:32:38 2015 -0500 Merge pull request #10 from jj-ian/implment_room_fields
26e5761gregorychen3 Sun Nov 22 21:27:58 2015 -0500 pretty printer now generates room class with fields
f14c3cc2gregorychen3 Sun Nov 22 21:13:47 2015 -0500 hello world program contains room def w/ string fields
93685afgregorychen3 Sun Nov 22 21:07:49 2015 -0500 pretty printer taking vdecls. type of vdecl not yet implemented
fb7066aIris Sun Nov 22 20:04:56 2015 -0500 Still hacking away at semantic checker
151cc41Gregory Chen Sun Nov 22 19:26:46 2015 -0500 Merge pull request #9 from jj-ian/implment_room_fields
df320eegregorychen3 Sun Nov 22 19:19:14 2015 -0500 trivial formatting and gitignore changes
6adb6cdgregorychen3 Sun Nov 22 18:55:53 2015 -0500 Room.java is being trivially generated by pretty_printer.ml
d3180e38rian Slakter Sat Nov 21 22:44:00 2015 -0500 Merge pull request #8 from jj-ian/bslakter/pattern_matching_pretty_print
68130c0bslakter Sat Nov 21 22:42:32 2015 -0500 pattern matching taken care of - should be good
bad6dc4Julie Sat Nov 21 20:39:31 2015 -0500 compiled java target files
753a25e jj-ian Sat Nov 21 20:31:15 2015 -0500 Merge pull request #7 from jj-ian/jj-ian/javatarget
0e0cc6b Julie Sat Nov 21 20:18:35 2015 -0500 some more cleanup and refactoring
31030bd Julie Sat Nov 21 20:13:55 2015 -0500 cleaning up and refactoring java code
8b1e88a Julie Sat Nov 21 20:10:48 2015 -0500 support for items, picking up items from rooms, items i/o
c0ce03d Julie Sat Nov 21 18:26:56 2015 -0500 setting up fake function pointer for equipping items
96165cf Julie Sat Nov 21 17:50:38 2015 -0500 items and stats
8d12671 Julie Sat Nov 21 16:26:46 2015 -0500 test commit
de1060d Julie Sat Nov 21 16:50:26 2015 -0500 Work on semantic_checker begin, vars and funcs
d6db1c4 Brian Slakter Sat Nov 21 17:39:22 2015 -0500 Merge pull request #6 from jj-ian/bslakter/driver-pretty-start
72ac8c3 bslakter Sat Nov 21 17:38:54 2015 -0500 pretty printing basics
525de38 bslakter Sat Nov 21 17:11:08 2015 -0500 hello world printing, rooms being declared in main method
07d81ce Gregory Chen Sat Nov 21 16:37:24 2015 -0500 Merge pull request #5 from jj-ian/gregs
c887da5 gregorychen3 Sat Nov 21 16:20:24 2015 -0500 test_hello_world.sh now works trivially with pretty_printer
62e5aaf gregorychen3 Sat Nov 21 15:59:09 2015 -0500 new hello world program is being received by pretty_printer
60988ec gregorychen3 Sat Nov 21 15:50:24 2015 -0500 altered helloword.java to pass new parser start symbol. passes jast also
04a90a5 bslakter Sat Nov 21 16:11:42 2015 -0500 committing up javatarget stuff
4b0d834 gregorychen3 Sat Nov 21 14:09:00 2015 -0500 Added trivial pretty_printer.ml, modified tbag.ml and Makefile accordingly.
00914c6 gregorychen3 Sat Nov 21 13:37:39 2015 -0500 fixed tabbing in jast.ml
3fcf0f gregorychen3 Sat Nov 21 13:34:16 2015 -0500 fixed tabbing java_builder.ml
802318a Brian Slakter Sat Nov 21 12:28:20 2015 -0500 Merge pull request #4 from jj-ian/bslakter/sast_to_jast
f12622f bslakter Sat Nov 21 12:02:30 2015 -0500 simplify sast
663e35f bslakter Sat Nov 21 11:49:55 2015 -0500 SAST TO JAST woot
4f15cab bslakter Sat Nov 21 11:27:02 2015 -0500 debugging, should be good tho
dc4fcc7 bslakter Sat Nov 21 10:03:09 2015 -0500 added jast, about to write sast to jast converter
d243b2d Brian Slakter Fri Nov 20 23:02:29 2015 -0500 Merge pull request #3 from jj-ian/bslakter/master
8e1b74d bslakter Fri Nov 20 23:06:01 2015 -0500 sast updated
4e55103 bslakter Fri Nov 20 23:03:08 2015 -0500 added all parts of program to grammar
6f77af6 bslakter Fri Nov 20 22:36:57 2015 -0500 tryin to figure out why rules never reduced
6c80d11 bslakter Fri Nov 20 21:41:11 2015 -0500 full language in progress
ae6f253 bslakter Fri Nov 20 21:28:03 2015 -0500 trying to clean
dd51c5b bslakter Thu Nov 19 18:52:04 2015 -0500 fixed - sast interface should be set
30c0858a Maria van Keulen Thu Nov 19 17:58:46 2015 -0500 Merge branch 'testchangebranch' into fixed_master
e6bf575 Maria van Keulen Thu Nov 19 17:48:57 2015 -0500 Added sample test output file
2ce9f7f Maria van Keulen Thu Nov 19 17:47:29 2015 -0500 Merge branch 'fixed_master'
of https://github.com/jj-ian/tbag into fixed_master
c6391ab Maria van Keulen Thu Nov 19 17:47:21 2015 -0500 Added output file
b53b18e bslakter Thu Nov 19 16:50:03 2015 -0500 in progress need to fix vdecl
d72d881 bslakter Thu Nov 19 16:31:51 2015 -0500 sast created, ast and parser updated
for variable decls
058809f Maria van Keulen Thu Nov 19 16:26:15 2015 -0500 Test suite now tests output
instead of source
b10aa61 bslakter Thu Nov 19 14:30:41 2015 -0500 updated git ignore
3281c0c bslakter Thu Nov 19 14:05:03 2015 -0500 adding in java target stuff
156a89e Maria van Keulen Mon Nov 16 22:53:11 2015 -0500 Updated test cases, removed
comments and echos
753e757 Maria van Keulen Mon Nov 16 22:46:48 2015 -0500 Merge branch 'bintestbranch'
78654d gregorychen3 Sun Nov 15 19:05:36 2015 -0500 added a msg string field to room
class in compile.ml. goodnight world
0fb86ff gregorychen3 Sun Nov 15 18:40:57 2015 -0500 added *.java to gitignore. more
formatting improvements to compile.ml
1c80210 gregorychen3 Sun Nov 15 17:39:10 2015 -0500 Did a bunch of code formatting
(tabbing mostly)
311b3a7 gregorychen3 Sun Nov 15 17:27:50 2015 -0500 Empty room decls now compile.
Formatting looks good in .java file.
80d00cf gregorychen3 Fri Nov 13 15:55:27 2015 -0500 removed now-unnecessary make.sh
91fa96a gregorychen3 Fri Nov 13 15:53:15 2015 -0500 Removed testing from Makefile. to
test, run a the script separate from the Makefile
f7305f6 Maria van Keulen Thu Nov 12 18:06:24 2015 -0500 run_tests more generic, added
more test files
ec16c26 bslakter Thu Nov 12 17:47:41 2015 -0500 parens working too
10e1266 bslakter Thu Nov 12 17:32:12 2015 -0500 more binops
f3945f d bslakter Thu Nov 12 17:26:37 2015 -0500 binops in progress w00tt
6bf8bd8 bslakter Thu Nov 12 17:20:25 2015 -0500 Merge branch 'master' of
https://github.com/jj-ian/tbag
f431342 bslakter Thu Nov 12 17:20:21 2015 -0500 functions working well - can utilize
ids
3b8510d Maria van Keulen Thu Nov 12 17:03:10 2015 -0500 Added run_tests.sh script
6ec48d7 Maria van Keulen Thu Nov 12 17:02:31 2015 -0500 Added test files
e863c8c bslakter Thu Nov 12 17:00:23 2015 -0500 about to add expr recognizing id
c26120a bslakter Thu Nov 12 16:32:50 2015 -0500 can do other functions too - about
to refactor
fc2ef4b bslakter Thu Nov 12 16:29:43 2015 -0500 Merge branch 'master' of
https://github.com/jj-ian/tbag
b2e18d a bslakter Thu Nov 12 16:29:38 2015 -0500 can write other functions
59a5243 gregorychen3 Thu Nov 12 16:23:28 2015 -0500 run.sh is now test_hello_world.sh
63adc2d gregorychen3 Thu Nov 12 16:01:39 2015 -0500 Added clean functionality to
Makefile. Deleted clean.sh
c407eb0 bslakter Thu Nov 12 15:54:28 2015 -0500 Merge branch 'master' of
https://github.com/jj-ian/tbag
9262567 bslakter Thu Nov 12 15:54:17 2015 -0500 makefile fixed
b0ccdd25 gregorychen3 Thu Nov 12 15:47:15 2015 -0500 updated .gitignore to ignore *.class
files
5e23220 bslakter Thu Nov 12 15:33:34 2015 -0500 separate makes
4733d67bslakter Thu Nov 12 15:27:30 2015 -0500 hello world added
4484156bslakter Thu Nov 12 15:26:44 2015 -0500 runner added
93cle77bslakter Thu Nov 12 15:26:00 2015 -0500 Make (it rain)
a3dd5c0bslakter Thu Nov 12 13:32:03 2015 -0500 just type make and we good
3f397b7bslakter Thu Nov 12 13:25:44 2015 -0500 HELLO WORLD SUCKASSSSS
66b942ebslakter Thu Nov 12 13:00:20 2015 -0500 almost hello worlding w000t
beetches
17cc8c7bslakter Thu Nov 12 12:51:10 2015 -0500 proper makefile now up - just type make
df4a2c2bslakter Thu Nov 12 12:43:33 2015 -0500 programs connected properly via make
script, commented out compilation issues - about to make proper makefile
0fc7dddbslakter Wed Nov 11 20:09:10 2015 -0500 missing something in ast but almost there
3ccf7b4bslakter Wed Nov 11 19:42:18 2015 -0500 pushing so greg can work on it too
f454664bslakter Wed Nov 11 19:29:46 2015 -0500 are we good to go omg
4bcf3a0Iris Wed Nov 11 19:26:08 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
cb301cfIris Wed Nov 11 19:26:04 2015 -0500 Adding Call function to parser and ast
38fffcdbslakter Wed Nov 11 19:25:29 2015 -0500 should be able to print
6557e98bslakter Wed Nov 11 18:02:30 2015 -0500 basic program added, tbag
e2711bdgregorychen3 Wed Nov 11 17:45:17 2015 -0500 Changed tbag.ml to get java code. added JavaCode.ml
553c7d7Iris Wed Nov 11 17:38:58 2015 -0500 Take out all dollar_sign stuff and fix make.
make.sh
f146784gregorychen3 Wed Nov 11 17:25:24 2015 -0500 Added code for tbag.ml. modified make.sh to compile tbag.ml
eee05d7eIris Sun Nov 8 16:26:22 2015 -0500 Accepting Brian's changes to array access
61d5f07Maria van Keulen Sun Nov 8 15:41:04 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
8a52aebMaria van Keulen Sun Nov 8 15:41:00 2015 -0500 Merge branch 'item_npc'
212e629bslakter Sun Nov 8 15:19:04 2015 -0500 array access done too
f1e12ecbslakter Sun Nov 8 14:57:38 2015 -0500 array decls and assignment woooot
1dcaae2Maria van Keulen Sun Nov 8 13:47:25 2015 -0500 Finished temp optional npc/item implementation
9061e3fbslakter Sun Nov 8 13:46:44 2015 -0500 adjacency declarations are good to go
5e8134dMaria van Keulen Sun Nov 8 11:21:18 2015 -0500 Added temporarily mandatory npc/item decls
7e28178Iris Sun Nov 8 04:18:50 2015 -0500 Adding array access to front end, WIP
6ee19bJulie Sat Nov 7 17:09:49 2015 -0500 variable declarations done
6799da8Julie Sat Nov 7 16:26:34 2015 -0500 more data type work, formal arguments, argument declarations
a9b213dJulie Sat Nov 7 14:52:19 2015 -0500 working on function decl in parser1, added some stuff to scanner, scanner now reports syntax error
2a31b66Julie Sat Nov 7 14:12:14 2015 -0500 removed scanner executable from git tracking
2aa0989Julie Sat Nov 7 14:09:54 2015 -0500 Merge branch 'julies_branch'
559fb83Julie Sat Nov 7 14:07:33 2015 -0500 Updated gitignore to ignore scanner executable
9ae2056Julie Sat Nov 7 14:06:00 2015 -0500 added support for string and int literals
837cf54Gregory Chen Sat Nov 7 11:53:53 2015 -0500 ast and parser now compile with changes to rdelc
Julie Sat Nov 7 01:47:31 2015 -0500 added types, but beware rdecls_list isn't compiling, committing to check out an earlier commit
ea84ac4Julie Sat Nov 7 01:10:16 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
5885774Julie Sat Nov 7 01:10:09 2015 -0500 deleted Other Stuff folder since it was copied to Other
f8a6bf8Gregory Chen Fri Nov 6 16:54:08 2015 -0500 Merge branch 'gregs_branch'
55bdecfGregory Chen Fri Nov 6 16:45:42 2015 -0500 syntax requires at least 2 rooms
19fd036Gregory Chen Fri Nov 6 16:40:45 2015 -0500 multiple room_decls implemented
5ad263Gregory Chen Fri Nov 6 15:44:26 2015 -0500 run_menhir starts menhir now.
380feGregory Chen Fri Nov 6 15:33:17 2015 -0500 fixed run_menhir.sh
7bd33fG Gregory Chen Fri Nov 6 14:57:58 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
7a06a32Iris Fri Nov 6 00:35:32 2015 -0500 Add operators to parser
9cf90b0mvankeulen94 Thu Nov 5 16:34:58 2015 -0500 Merge pull request #2 from jj-ian/while_return
f130c5aMaria van Keulen Thu Nov 5 16:31:45 2015 -0500 Added return construct
bc5a168Maria van Keulen Thu Nov 5 15:48:09 2015 -0500 Added while construct
f4c8311mvankeulen94 Thu Nov 5 10:54:23 2015 -0500 Merge pull request #1 from jj-ian/ifelse_branch
e03fc6aMaria van Keulen Wed Nov 4 21:50:46 2015 -0500 Fixed formatting of exprs and stmts
81e6e35Maria van Keulen Wed Nov 4 20:20:59 2015 -0500 Removed unnecessary semi declaration
f3f83fbbIris Wed Nov 4 14:51:01 2015 -0500 Add trivial print stmt to scanner
a3e11aIris Wed Nov 4 14:38:11 2015 -0500 Renamed Other stuff Other
2ac4e09Maria van Keulen Wed Nov 4 14:00:18 2015 -0500 Added initial code for if/elses
6d60f5Julie Tue Nov 3 20:12:52 2015 -0500 made scanner.mll print out tokens, rest of pattern match needs to be filled out. check the bottom of the scanner.mll file
6156971Julie Tue Nov 3 18:10:42 2015 -0500 fixed compile issue, updated make script to compile everything, updated clean script to clean everything, fixed bugs in ast and parser
b0a74f4Julie Tue Nov 3 16:58:59 2015 -0500 updated ast.mll and scripts, fixed the 'unbound type constructor Ast.program error'
b03eaafGregory Chen Tue Nov 3 12:30:09 2015 -0500 Merge branch 'master' of https://github.com/jj-ian/tbag
6b2401dJulie Tue Nov 3 01:11:50 2015 -0500 added more options to clean script. still working on fixing 'Error: Unbound type constructor Ast.program' error
7b0e3e5Julie Tue Nov 3 00:36:26 2015 -0500 debugging and adding more flags to scripts. added compile.Scanner.sh to compile just the scanner
631355eGregory Chen Fri Oct 30 22:29:33 2015 -0400 more indentations to parser
2cbfbb8Gregory Chen Fri Oct 30 22:20:31 2015 -0400 indentation changes to parser
7c167beGregory Chen Fri Oct 30 22:16:44 2015 -0400 minor changes to scanner.mll
d5fbdcd3Iris Fri Oct 30 22:03:21 2015 -0400 Add operators to parser and ast
db1734cGregory Chen Fri Oct 30 21:50:32 2015 -0400 fixed indenting in scanner.mll
f8db1c0Iris Fri Oct 30 21:46:23 2015 -0400 Merge branch 'master' of https://github.com/jj-ian/tbag
7c366fd6Iris Fri Oct 30 21:46:19 2015 -0400 Add operators to scanner
1fa0789Gregory Chen Fri Oct 30 21:33:41 2015 -0400 changed test.sh to run_menhir.sh
Display commands being executed. Started test.sh

File commands being executed. Started test.sh

Modified clean.sh and make.sh to display commands being executed. Started test.sh

Merge branch 'master' of
https://github.com/jj-ian/tbag

brian's regex sample code

Add trivial stuff to scanner from microC

Adding assign to scanner

Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 22:37:56 2015 -0400  added gitignore, deleted some files
E897Fa7Gregory Chen Sun Oct 25 22:35:16 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

d978e29Gregory Chen Sun Oct 25 22:24:25 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 22:20:50 2015 -0400  parser


Julie

SUN OCT 25 22:13:52 2015 -0400  fixed some things in ast, parser, scanner
A769524Gregory Chen Sun Oct 25 22:13:44 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 22:11:51 2015 -0400  deleted attempt at making an interpreter

Julie


Julie

SUN OCT 25 21:47:56 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 21:13:58 2015 -0400  Deleted unneccessary prog type from ast.ml

Julie

SUN OCT 25 20:46:33 2015 -0400  added sample code for calc and microc

Julie

SUN OCT 25 20:41:44 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 20:41:42 2015 -0400  commented out something
A65762María van Keulen Sun Oct 25 20:41:09 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 20:40:03 2015 -0400  Fixed compilation errors
47a66e6f4Gregory Chen Sun Oct 25 20:39:12 2015 -0400  Further organized microc_original
Ad6602bGregory Chen Sun Oct 25 20:37:08 2015 -0400  organized microc compiler sample code

Julie

SUN OCT 25 20:35:19 2015 -0400  Added room skeleton
E976c2aGregory Chen Sun Oct 25 20:32:07 2015 -0400  fixed a bug in ast.ml
6392448Julie Sun Oct 25 20:30:21 2015 -0400  ast

Julie

SUN OCT 25 20:26:16 2015 -0400  added tbagInterpreter.ml; adjusted make.sh and clean.sh accordingly
E34c822Gregory Chen Sun Oct 25 20:14:26 2015 -0400  added some more things to clean.sh
9606b44Gregory Chen Sun Oct 25 20:10:28 2015 -0400  Deleted some intermediate files
F8661aeGregory Chen Sun Oct 25 20:08:42 2015 -0400  added parser.ml to clean.sh

Julie

SUN OCT 25 20:08:13 2015 -0400  Merge branch 'master' of
https://github.com/jj-ian/tbag

Julie

SUN OCT 25 20:06:35 2015 -0400  ast
corrected a bug in make.sh
added a command to clean.sh
Merge branch 'master' of https://github.com/jj-ian/tbag
Added make.sh and clean.sh
Merge branch 'master' of https://github.com/jj-ian/tbag
Things added to scanner.
Merge branch 'master' of https://github.com/jj-ian/tbag
progress on ast and scanner
Merge branch 'master' of https://github.com/jj-ian/tbag
ast and parser
changed tbag compiler to tbag_compiler
working on parser and ast
added scanner.mll
ast
microc compiler
Initial commit