0. Introduction

0.1 Bali Specifications

As discussed in lecture, we can specify a language in terms of its syntax (spelling and structural rules) and semantics (meaning). Although formal description methods exist for both kinds of specifications, we will only formalize the Bali syntax, as developed in Section 1. Bali’s semantics are informally defined in Section 2.

0.2 Notation

We use the following notation throughout this document:

- Specific language elements, like keywords, operators, and punctuation, are shown as bold.
- Other terminals are shown as bolditalic, such as names and integers, because they can have different values.
- Non-terminals in production rules are shown as plain red.
- A collection of elements with an ellipsis (...) indicates at least one element is in the entire production, possibly more. For example, int name , ... , name means that this production may have one or more names.
- An arrow (→) indicates a production rule, which means can be expressed as.
1. Bali Syntax

1.1 Program

program → mainfn

1.2 main Function

mainfn → mainheader
   namedcl
   block
   returnexpr

mainfn → mainheader
   block
   returnexpr

mainfn → mainheader
   returnexpr

mainheader → int main ( )

namedcl → int name , ... , name ;

returnexpr → return expression ;

block → { statement
   ... statement }

block → statement

1.3 Statements

statement → name = expression ;

statement → if ( expression ) block else block

statement → if ( expression ) block

statement → while ( expression ) block

statement → do block while ( expression ) ;

statement → expression ;

statement → print expression ;

statement → read name ;

statement → ;

1.4 Expressions

expression → name

expression → integer

expression → ( expression )

expression → ( expression + expression )

expression → ( expression - expression )

expression → ( expression * expression )

expression → ( expression / expression )

expression → ( expression % expression )

expression → ( expression == expression )

expression → ( expression < expression )

expression → ( expression > expression )

expression → ( expression || expression )

expression → ( expression && expression )

expression → ( expression ^ expression )

expression → ( ! expression )
2. Bali Semantics

This section provides more details on Bali’s semantics. As we enhance Bali’s grammar in Parts 3 and 4 of the project, we will update this document to reflect the changes. Note that many of the following rules elaborates the syntax described in Section 1.

2.1 main Function

We will concentrate on just main for Part 2. The first and only function must be called main( ), which takes no arguments.

- main must have a return statement as the last statement.
- main must return exactly one integer.

Bali may not return from anywhere inside main’s body other than the last statement.

2.2 Variables

Bali has only one type of variable, integer, as described below:

- Variables must be declared as the first statement in main. They may not be declared anywhere else.
- Variable names must contain only English letters and are case-sensitive.
- Integers are declared with keyword int on the same line.
- Variables do not receive default values.
- Boolean variables use integers (see below).

You may assign variables according to the following rules:

- You may not declare and assign a variable in the same statement. A variable must be declared before being used.
- Integers use Java’s range for int.
- Boolean values use 0 for false and any non-zero number (usually 1) for true.

2.3 Arithmetic

Bali has the following rules, which are more restrictive than in other languages:

- Bali has addition (+), subtraction (–), multiplication (*), and division (/).
- Bali’s modulus operator (%) has the same syntax as Java’s %.
- All operations produce integers. Division “rounds down,” as Java does.
- Bali does not employ operator precedence for arithmetic and logical operations, so you must surround every operation with parentheses ( ( ) ).

2.4 Control Structures

Control structures, such as selection and repetition statements, follow similar rules to those you have seen in other languages:

- Statements may be collected in blocks as part of a control structure.
- The condition clause of a control structure must be a legal expression, which must result in an integer value.
2.5 I/O
We will use rudimentary I/O statements:

- The `print` statement sends the evaluated result of an expression to the SaM console panel.
- The `read` statement opens a window for the user to input a legal integer value. The value is pushed on the stack after it is read.

2.6 Comments/White Space
As with any language, you should always comment your code:

- The double slash (`//`) indicates a comment line, which is ignored by the compiler.
- Bali has no multiline comments.
- You may place comments on a new line.
- You may place comments on the same line of code: Bali ignores everything to the right of the `//` on the same line.
- You may place comments above and below a function.
- You may place as many blank lines in your code as you wish.

2.7 Punctuation

- Parentheses (`( )`) punctuate expressions. Note that `main`’s header uses parentheses to indicate that `main` is a function with no parameters.
- New statements may be written on the same line. Be sure to properly punctuate them!