(Note for Grader) When testing my program that looks at the lexical and syntactical analysis. Please remember to place spaces between all the lexical tokens in the input file. The professor told me personally I could require this for my program. So for example, a while loop would look like: while ( expression ) { //code blocks ; } rather than: while(expression){//code blocks;}

1. Data Types/Values

In my programming language there are 5 data types that is available for variables.

The data types are: int (integer), float (real numbers), string, void, and Boolean.

* 1. Int (Integer)

Int values are just standard numbers/values with no decimal point whatsoever, so for example: 1, 2, 21, would all be considered Integer values in our language.

* 1. Float (Real Numbers)

Floating-point values are the values with decimal points or are fractions. These values have less limits then Integers and should be used when you want to get exact values that are not integers. Examples of these values would be: 1,75, 9.989, 0.918382 or 1.23e7

* 1. String

String values are used when we have a sequence of characters in between normal quotation marks (“) or single quotation marks (‘)

* 1. Void

Void values in my language are just nothing. They are essentially equivalent to pythons “None” value.

* 1. Boolean

Boolean values are used when you only need to choose 1 of 2 values. Whether that be 0 or 1, or true or false.

1. Rules for literal values

2.1) To be considered a valid integer literal, you have to be a integer/ or sequence of just integers. Our regular expression used to identify integers is: “([1-9][0-9]\*) | 0“

2.2) To be considered a floating point literal, you need to be a sequence of numbers, but the thing I look for to differentiate a float and an integer is either a period(.) or an e ( e or E ) which is the letter usually representing the “times 10 raised to the power of). The regular expression used was:

'([+|-])?(\d+([.]\d\*)?([e]([+|-])?\d+)?|[.]\d+([eE]([+|-])?\d+)?)'

2.3) To be considered a valid string literal, you simply have to be a sequence of letters in between either a single quotation mark or double quotation mark. The regular expression used was:

"(\"|\')([a-zA-Z][a-zA-Z]\*)(\"|\')"

2.4) A Boolean value will either be set to: 0, 1, true or false

1. Variables

In order to use variable sin my language, you must first initialize them and give it them a types. (Refer to 3.1 for the initialization process). When you initialize a variable, they will be stored in a symbol table, and in the symbol table it will keep track of a variables type, and will track the exact value of the variable also.

When that variable is used in mathematical operations, we will first make sure all the variables in the math expression are of the same type, then we will pull the values and do actual math. If the variable is being assigned a new value, that old value will be replaced by the new value from the expression.

3.1) Syntax for the initialization of a variable

TYPE IDENTIFER ;

An real example would be: int x ;

3.2) Syntax for assignment to a variable

IDENTIFIER = <expression> ;

Or

IDENTIFIER = STRING LITERAL ;

Examples: x = 3 + 2 ;

Or

S = “Majesty”

Semantic Rules for assignment: If we try to assign a value to a variable and they are not of the same type we will receive a mismatch error. Remember to make sure they are the same types. So for example a string literal should be assigned to a string variable, and a string literal shouldn’t be assigned to an integer variable.

1. Rules for control Structures

4.1)

* Switch

The syntax I followed for my switch statement, was similar to C++ switch statement.

Switch ( <expression> ) {  
  case x:  
    *// code block*  
  case y:  
    *// code block*  
  default:  
    *// code block*  
}

* Foreach

The syntax I used for my foreach statement was similar to Java’s for each statement

foreach ( identifier/int identifier : identifier ) {

//code blocks

}

* For

The syntax I used for my for loop was similar to C++’s for loop statement

for ( <expression> ; <expression> ; <expression> ) {  
   *code blocks*  
}

* While

The syntax I used for my while loop was similar C++’s for loop statement

while ( <expression> ) {  
 *code block*  
}

* Do-while

The syntax I used for my do-while loop was similar c++’s do while statement

do {  
 *code block*  
} while ( <expression> )

* If

The syntax I used for my if statement was C++’s if statement

if ( <expression> ) {  
   code blocks   
}

* Assignment

The syntax I used for assignment is similar to Java.

Identifier = <expression> ;

* Return

The syntax I used for the return statement is similar to Pythons return statement, so in ours we can just have keyword return with the semicolon, or you can have keyword return with an identifier, float, or integer followed by a semicolon

return ;

Or

return identifier ;

Or

return float ;

Or

return integer ;

4.2) Tradeoffs

The only real trade off that I had to make was with the assignment variable. My program looks through the code one lexem at a time, and it

1. Rules for Methods and Classes

* Class

The syntax I followed was similar to java

Public class IDENTIFIER {

//code block

}

Or

private class IDENTIFIER {

//code block

}

* Method

Method TYPE IDENTIFER ( ) {

//code blocks

}

* Expression

<expr> -> <term> {(+|-) <term>} | string\_literal

* Term

<term> -> <factor> {(\* | /) <factor>}

* Factor

<factor> -> id | int\_literal | floating\_point\_literal | ( <expr> )

1. Keywords/Special symbols

6.1) keyWords = {

'foreach' This is used to begin a signal for a foreach loop

'for' This is used to begin a signal for a for loop

'if' This is used to begin a signal for a if statement

'while' This is used to begin a signal for a while loop

'do': This is used to begin a signal for a do-while loop

'int' This is used to declare a type of int

'float' This is used to declare a type of float

'switch': This is used to begin a signal for a switch statement

'return': This is used to begin a signal for a return statement

'else' : This is used to signal the other option in an if statement

'case': This is specifically used in the switch statements to signal for each case

'default' : This is specifically used in the switch statements to signal for the default case

'in' : This is used like pythons ‘in’

'VOID' This is used at the beginning of our program

'MAIN' This is used at the beginning of our program

'type'

'public' This is used when we are determining the security for our class

'private' This is used when we are determining the security for our class

'class' : This

'method' This is used to signal the beginning of a method

'void' : This is used to declare a type of void

}

special\_sym = {

'+': ['Plus\_sign', 11],

'-': ['Minus\_sign', 12],

'(': ['Left\_parenthesis', 41],

')': ['Right\_parenthesis', 42],

'{': ['Left\_bracket', 43],

'}': ['Right\_bracket', 44],

'\*': ['Multiplication\_symbol', 13],

'/': ['Division\_symbol', 14],

'$': ['Dollar\_sign', 19],

'%': ['Percent\_symbol', 15],

'=': ['Equal\_sign', 17],

'<': ['Less\_than', 21],

'>': ['Greater\_than', 22],

'<=': ['Less\_Than\_Equal', 23],

'>=': ['Greater\_Than\_Equal', 24],

';': ['semicolon', 77],

':' :['colon', 78]

}

1. Instructions for creating valid code.

We must first begin with: “ VOID MAIN ( ) { < // CodeBlock > }

In the code block area, you are able to use all of the statements and control structures that have already been discussed and remember to put a space between all of your lexemes or you’ll likely receive a syntax error. When we are in these code blocks, my code looks at the next Lexem to determine where it will go next. So, for example, if the next token is an if statement, it will run to the if statement and it will expect the syntax for the if statement to be carried out immediately. In these code blocks it will continuously look at the first lexem to determine what it will be do, until it sees a right bracket, in which case it will return from where it came from