SER 502 Project

Language name: Habesome

Milestone 2

Group 18

Group Member

Sowmya Madabhushi

Harshita Kajal

Melissa Day

Behnaz Sabbaghi

Operators	Description
‘+’	Add
‘-‘	Subtract
‘*’	Multiply
‘/’	Divide
‘.’	End of statement
‘[’	Beginning of block
‘]’	End of block
‘is’	Assignment
‘=:’	Comparator
‘<<’	Less than
‘>>’	Greater than
‘!!’	NOT
‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’	Digits
‘a’, ‘b’, ‘c’, …, ‘x’, ‘y’, ‘z’, ‘A’, ‘B’, ‘C’, …, ‘X’, ‘Y’, ‘Z’	Letters

Keywords	Description
Bool	Boolean
Int	Integer
if-else	Conditional statement
while	Iterative execution
start	Indicates the beginning of the program
end	Indicates the end of the program
True	Values to the boolean data type
False	Values to the boolean data type

Design And The Grammar of the Language:

P is Program
K is Block
D is Declaration
SL is Statement List
DT is data types
S is Statement
A is Assignment Statement
IF is If Statement
W is While Statement
B is Boolean Expression
E is Arithmetic Expression
EX is Multiplication or Division Arithmetic Expression
I is Identifier
N is Number
INT is integer data type
BOOL is boolean type
DG is Digit
L is Letter

P ::= 'start' K 'end'

K ::= '[' D '.' SL ']' |
'[' SL ']'

D ::= DT SL '.' D |
DT SL '.'

DT ::= INT | BOOL

SL ::= S '.' SL |
S '.' |
K

S ::= A | IF | W

A ::= I 'is' E

IF ::= 'if' '(' B ')' K 'else' K

W ::= 'while' '(' B ')' K

```
B ::= T |
        F |
        E '=:' E |
        '!!' E |
        E '<<' E |
        E '>>' E

E ::= EX '+' E |
        EX '-' E |    // left associative???
        EX

EX ::= I '*' EX |
        N '*' EX |
        I '/' EX |
        N '/' EX |
        I |
        N

I ::= L I |
        L

N ::= DG N |
        DG

INT ::= [0-9]+

BOOL ::= 'true'|'false'

DG ::= [0-9]+

L ::= ('a'..'z' | 'A'..'Z')*
```

Parsing Technique :

We have chosen to work with the top-down Parsing technique since we are more familiar with this approach and also it is more popular. Moreover, efficient parsers can be constructed more easily by hand with these methods which is how we plan to implement our program.

Interpreter Information:

We plan to use the “eval” predicate since it helps us to change from one environment to the other environment. Although our initial plan is to write in Java, we ultimately hope to shift to Prolog as we further develop our language.

Data Structures:

Currently, we plan to use lists or stacks as our basic data structure.