Team Members: Abhishek Dutta, Akil Kumar Thota, Apoorv Khairnar, Dhananjay Pandit

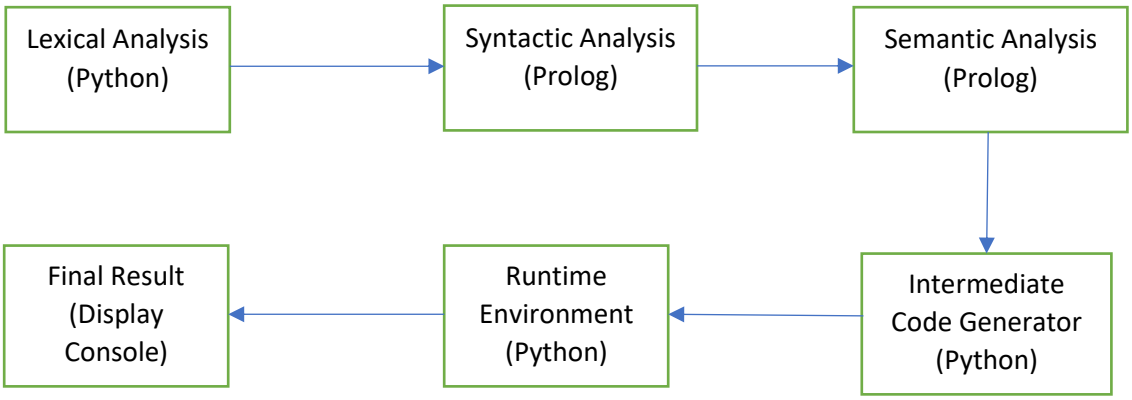
Team Number: 8

Name of Language: ***LESL (Lazy Evaluated Simple Language)***

Design of Language:

Our team is building an Imperative Structured Language. Please find below our architecture for whole project and the techniques which we are using.

Architecture: The words enclosed in brackets denote the technology used to implement the module



We have decided to use Python for Lexical Analysis, Intermediate code generation and Runtime Environment. The motivation behind using Python is that it is simple and fast to prototype and all 4 of us have good knowledge of the language. We will be using Prolog for Syntactic and Semantic Analysis. Professor Bansal had given us a rough idea of how these stages can be accomplished in Prolog and we feel it would be easier to build on that idea, given that a solid foundation was already laid in class.

The input for the lexical analysis will be the code written in the source file. This code will be in our language. Lexical analysis using Python will give us “tokens” which will be nothing but a list of lists of all the tokens present in the source code. These tokens will be checked for syntax and semantics using Python. If there is no syntax error in the source code, the semantic analyzer will generate a parse tree which will be passed to another Python program as input. This program will generate the intermediate code from the parse tree which will emulate opcodes. These opcodes will undergo “lazy evaluation” in the runtime environment to finally display the result on console.

The key features of our language are inferred data type, reuse of a variable as different data type, lazy evaluation of expressions and the inclusion of the ternary operator as a conditional statement. We feel that these features will make our language more interesting to build. The lazy evaluation feature will allow our language to be evaluated if and only if the value of variable is asked for. This will reduce our evaluation time because we won't make redundant evaluation where the value is never used again in the program.

Language Constructs

1) Statements: Assignments and expressions should end with "." operator.

2) Variables: It should always start with a capital letter.

3) No data type declaration.

4) Assignment is done using "=" operator.

5) Conditional Statements: We are supporting "if", "elseif" and "else" clauses. We are also supporting ternary operators (?,:)

6) Loops: We are supporting "while" loop.

7) Operators: We support "+", "-","*","/","%",">","<",">=","<=",”! =","==","(",")", "and”, "or" , "not"

8) Code Blocks: Code blocks under the conditional Statements and Loops should start with "begin" and end with "end"

9) Show: We are trying to output the standard output using "#show".

10)Keywords: begin, end, if, elseif, else, while, #show, and, or, not.

11)Comments: We are allowing single line comments starting with "@"

Sample Code

```
A = 2.
B = 3.
if(A > B)
        begin
                #show A.
        End
else
        begin
                #show B.
        end


while(A<5)
        begin
                A = A + 1.
                #show A.
        end
C = false.
#show C.
```

PS: Grammar is in another document.