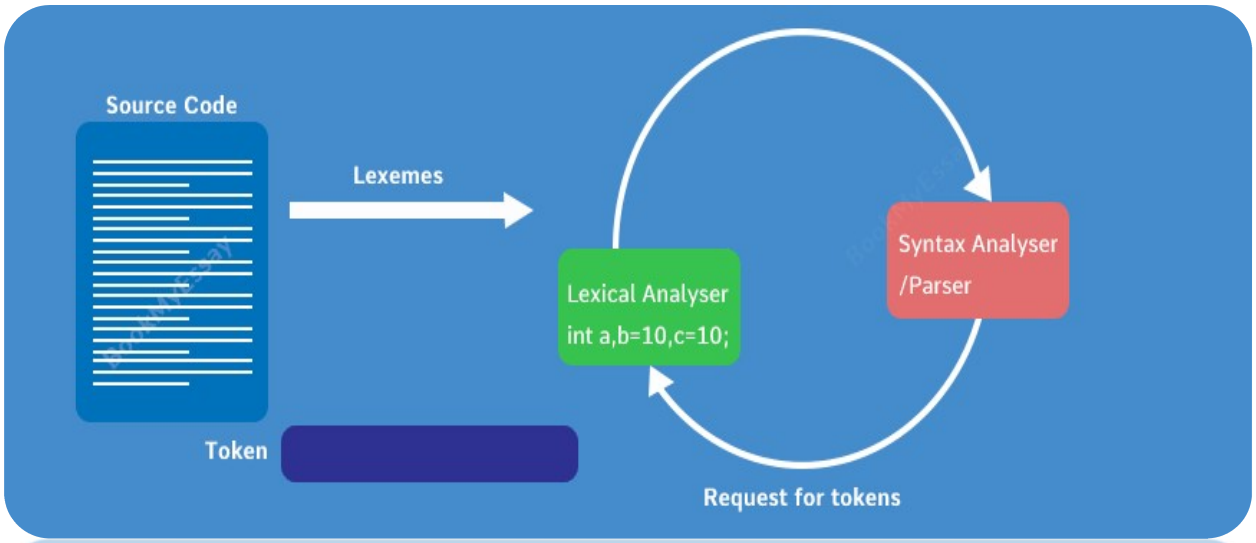

Compiler Project Phase 3

CodeGen

Due :7 /22 /2020

CodeGen :

Welcome To The Final Round !!!!

After your great accomplishment in implementation of a Scanner and a Parser for ***Roulang***, it's time to finish it's development by implementing the CodeGen part.

In this final (and also the most challenging) part of the project, we are going to generate low-level codes for our language. To be more precise, we are going to generate ***JAVA Bytecodes*** which can be executed on **JVM (Java Virtual Machine)**.

Structures:

A complete explanation about ***Roulang*** Structures has been given in Parser phase so your program must be able to generate ***Bytecodes*** for all of those structures.

As it has been explained in Parser Phase, ***Roulang*** communicates with ***Command Prompt*** through two functions, ***Println()*** and ***Input(<type>).*** Please notice that your project will be evaluated by some sample input codes, which attaches great importance to the implementation of those two functions. conspicuously your failure in their implementation results in a significant loss of points.

Some Notes:

- All programs must contain ***start()*** function which is equivalent to C++ ***main()*** function. Absence of this function must lead to a Compile Error.
- Implicit casts includes ***int*** to ***long*** , ***float*** to ***double*** , ***char*** to ***int*** and ***char*** to ***long*** conversions.
- Explicit casts includes ***float*** to ***int***, ***double*** to ***int***, ***float*** to ***long***, ***double*** to ***long***, conversions.
- A variable is global if and only if it's been declared outside of any function.
- Function calls are treated as ***call by value*** which means that any change to a function parameter won't affect it's value outside of that function ***unless*** it's type is ***array*** or ***record*** in which case it's value will be affected.
- Parameter overloading is supported which means that two functions with same names and return types but different parameters can be declared.

- Bitwise operators are used only with integer and long operators.
- Prefix/Postfix increment or decrement is used only with Integer an long operators.
- Recursive functions are supported.

Additional Features:

Following implementations will provide you with extra point :

- Len() function which returns size of a given ***array*** or ***string*** parameter.
- Multidimensional arrays(based on Java Syntax).
- Foreach loop (for complete explanation study Parser phase).
- Concatenation of two Strings with '+' operator.
- Giving Warning if there exists any unreachable code.

- Cascade assignment

    Example : opy1 = copy2 = copy3 = some_value;
- Proper error message for Syntax errors and declaring number of the line in which error raised.
- Any other innovation

Grading Policy:

We will provide some ***Roulang*** codes which will try to cover all aspects of the language. Initially there would be some test cases that are comiled with no errors and some that aren't, and your code must handle them, only showing the right ouput "compiled" or "not compiled" . After that your compiler must pass the tests which focus on ***ByteCodes***. Relative point of each part is specified in following table :

structure	point
Variables and Arithmetic Calculations	8
1D Array & String operations	8
Type Casting	5
If then else	7
Switch case	5
loops	12
Records	8
Functions	17
Len() function	2
Multidimensional arrays	5
Foreach loop	7
String '+' operator	3
Warning for unreachable code	8
Cascade assignment	3
Proper Error declaration	2

- ***Notes***:
  - If you haven't implemented Scanner in previous phases we will provide it for you however, you will lose 15% of the project score.
  - You will need to add semantic rules to your PGen diagrams which may require some modifications to be made in those diagrams.
  - This part of Project can be done in groups of two.

GOOD LUCK