# SER 502 Project
# First Milestone

## Language Name : YASL (yet another simple language)

## Grammar of the language:

Program -> Block.

Block -> Initialization;Block | Declaration;Block | Assignment;Block | Print;Block | If_conditional Block | Ternary_conditional;Block | For_loop Block | While_loop Block | Range_loop Block | Declaration; | Assignment; | Print; | If_conditional | Ternary_conditional; | For_loop | While_loop | Range_loop

Initialization -> Integer | String | Boolean

Declaration -> int Identifier | string Identifier | bool Identifier

Integer -> int Identifier = Digit | int Identifier = Identifier
String -> string Identifier = 'Sentence' | string Identifier = Identifier
Boolean -> bool Identifier = Identifier | bool Identifier = Boolean_value
Boolean_value -> true | false

Assignment -> Identifier = Expression | Identifier = Identifier | Identifier = 'Sentence'
| Identifier = Boolean_value | Identifier = Digit

Print -> print ('Sentence')

Sentence -> Digit Sentence | Identifier Sentence | Digit | Identifier
Digit -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Identifier -> a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z

If_conditional -> if (Condition) then { Block } | if (Condition) then { Block } else { Block }
Condition -> Expression Conditional_operator Expression | Boolean_value
Conditional_operator -> == | < | > | <= | >= | !=
Unary_operator -> ++Identifier | Identifier++ | --Identifier | Identifier--

Ternary_conditional -> Expression = Expression Conditional_operator Expression ? Expression
: Expression

While_loop -> while (Condition) { Block }
For_loop -> for (Integer; Condition; Unary_operator) { Block }
Range_loop -> for (Identifier) in range(Digit,Digit) { Block }

Expression -> Term + Expression | Term - Expression | Term
Term -> Factor * Term | Factor / Term | Factor
Factor -> Identifier | Digit

## Design:

For this project, we are planning to design a compiler for an Imperative programming language which will initially contain the following features:

- Three primitive types:- Integer, Boolean, and String.
- An identifier:- all the lowercase alphabets of the English language.
- An assignment operator, so that we can associate a value with an identifier.
- If then else condition - If the initial condition evaluates to true then the if block would be executed, otherwise the else block would be executed.
- Ternary condition:- It is a shorter notation of the if then else condition. If the condition is true, then the first block is assigned to the identifier, otherwise, the second block is assigned instead.
- While loop - A block that is executed repeatedly until the given condition evaluates to false.
- For loop - A block that is executed repeatedly for a known period of time until the given condition evaluates to false.
- Range loop - A block that is executed repeatedly between a defined range of numbers.
- Print - This is used to display the value of an identifier.

For the final product we also plan to add the following features to our design:

- Iterator for strings.
- Functions and function calls.
- Syntactic sugar for addition and subtraction,
- Implement escape characters like newspace and tabspace
- Step size for range

## Summary:

For this project we were asked to work as a team and design a compiler for our own language. This includes writing the grammar for our language, and making use of open source tools to generate a lexical analyzer and a parser. This document contains the initial version of the grammar for our language. To generate the lexical analyzer and the parser, we are going to make use of a tool called **ANTLR** (ANother Tool for Language Recognition). The parser, which will take in a stream of tokens generated by the lexical analyzer, is going to generate a **Binary Parse Tree** which we are going to give to our Semantic Analyzer. Then, we plan to write our Semantic Analyzer using Java to ensure that our language is semantically and syntactically correct.