SER 502
Project Milestone 2
Pradyumn Mohta, Arushi Shah, Pavan Kalyan

Brief Overview of the Language:
This language is a minimalistic, single-letter syntax language designed for efficient and compact code expression. Each command or keyword is represented by a single uppercase letter, reducing verbosity, increasing speed and making the language intuitive for those familiar with foundational programming constructs.

Lexical Grammar:

1. **Identifiers**: Represent variable names.
- Regex: [a-zA-Z_][a-zA-Z0-9_]*
- Description: Identifiers must start with a letter or underscore, followed by alphanumeric characters.

2. **Data Types**: Use single-letter uppercase for types.
- Tokens: T for integer, S for string, B for boolean.

3. **Keywords**:
- Print: P (for output).
- Control: I (if), E (else), W (while), F (for).

4. **Operators**:
- Arithmetic: +, -, *, /.
- Comparison: ==, <, >.
- Assignment: =.
- Ternary: ? :.

5. **Literals**:
- Integer literals: [0-9]+
- String literals: Enclosed in double quotes ("...").

Syntactic Grammar (EBNF Syntax):

```
Program       ::= StatementList

StatementList ::= { Statement ";" }

Statement     ::= VariableDecl | PrintStmt | IfStmt | WhileStmt | ForStmt

VariableDecl  ::= ("T" | "B" | "S") Identifier "=" Expression
PrintStmt     ::= "P" Expression
IfStmt        ::= "I" "(" Condition ")" "{" StatementList "}" ["E" "{" StatementList "}"]
WhileStmt     ::= "W" "(" Condition ")" "{" StatementList "}"
ForStmt       ::= "F" "(" AssignStmt ";" Condition ";" Expression ")" "{" StatementList "}"

Condition     ::= Expression ("==" | "<" | ">") Expression
Expression    ::= Term { ("+" | "-") Term }
Term          ::= Factor { ("*" | "/") Factor }
Factor        ::= Identifier | Number | String | "(" Expression ")"
Identifier    ::= [a-zA-Z_][a-zA-Z0-9_]*

AssignStmt    ::= Identifier "=" Expression
Number        ::= [0-9]+
String        ::= "\"" { Character } "\""
Character     ::= any printable character except "\""
```