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

1. 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.

2. Design Principles

- Single-letter keywords for common programming constructs
- Statically typed
- Support for basic data types and control structures
- Clear and unambiguous grammar

3. Tokens

3.1 Keywords (Single Letters)

- T : Integer type declaration
- B : Boolean type declaration
- S : String type declaration
- P : Print statement
- I : If statement
- E : Else statement
- W : While loop
- F : For loop

3.2 Literals

- Integer literals: [0-9]+
- String literals: "[^"]*"
- Boolean literals: 0|1

3.3 Identifiers

- Pattern: [a-zA-Z][a-zA-Z0-9]*
- Cannot be a single-letter keyword

3.4 Operators

- Arithmetic: +, -, *, /
- Relational: <, >, =
- Logical: & (AND), | (OR), ! (NOT)
- Ternary: ?, :

3.5 Delimiters

- Parentheses: (,)
- Braces: {, }
- Semicolon: ;

4. Grammar (EBNF)

```
program = { statement } ;

statement = declaration_stmt
        | if_stmt
        | while_stmt
        | print_stmt
        | expression_stmt ;

declaration_stmt = type identifier [ "=" expression ] ";" ;

type = "T" | "B" | "S" ;

if_stmt = "I" "(" expression ")" block [ "E" block ] ;

while_stmt = "W" "(" expression ")" block ;

print_stmt = "P" expression ";" ;

block = "{" { statement } "}" ;

expression_stmt = expression ";" ;

expression = logical_expr ;

logical_expr = relational_expr { ("&" | "|") relational_expr } ;

relational_expr = arithmetic_expr { ("<" | ">" | "=") arithmetic_expr } ;

arithmetic_expr = term { ("+" | "-") term } ;

term = factor { ("*" | "/") factor } ;

factor = integer_literal
      | string_literal
      | boolean_literal
      | identifier
      | "(" expression ")"
      | "!" factor ;

integer_literal = digit { digit } ;
```

string_literal = '"' { character } '"' ;

boolean_literal = "0" | "1" ;

identifier = letter { letter | digit } ;

letter = "a" | ... | "z" | "A" | ... | "Z" ;

digit = "0" | "1" | ... | "9" ;

5. Type System

- Integer (T): Whole numbers
- Boolean (B): Truth values (0 or 1)
- String (S): Text enclosed in double quotes

6. Operator Precedence (highest to lowest)

1. Parentheses ()
2. Unary operators !
3. Multiplicative *, /
4. Additive +, -
5. Relational <, >, =
6. Logical AND &
7. Logical OR |