Mini-Language Specification (Simplified)

1. Language Definition

1.1 Alphabet:

1. Uppercase (A-Z) and lowercase letters (a-z) of the English alphabet.
2. Underscore ('_') character.
3. Decimal digits (0-9) for numbers.

1.2 Lexical Elements:

a. Special Symbols:

- Operators: +, -, *, /, :=, <=, =, >=, &&, ||, !
- Separators: [ ] { } : ; space
- Reserved Words:

int bool char if else while input output

b. Identifiers:

Identifiers must begin with a letter, followed by letters or digits, and undersore

- Syntax:

identifier ::= letter [letter | digit | '_']

- letter is defined as:

letter ::= "A" | "B" | ... | "Z" | "a" | "b" | ... | "z"

- digit is defined as:

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

c. Constants:

1. Integer Constants:
   - Positive integers (no negatives).

   integer ::= digit {digit}

2. Boolean Constants:
   boolean ::= "true" | "false"

---

2. Syntax
The words - predefined tokens are specified between " and ":
a. Program Structure:
program ::= stmtlist

b. Statements:

stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= assignstmt | iostmt | ifstmt | whilestmt

c. Assignment Statements:

assignstmt ::= IDENTIFIER ":=" expression

d. Input/Output Statements:

iostmt ::= "input" "(" IDENTIFIER ")" | "output" "(" expression ")"

e. Conditional Statements:

ifstmt ::= "if" condition "then" stmt ["else" stmt]

f. While Loop:

whilestmt ::= "while" condition "do" stmt

g. Expressions and Conditions:

expression ::= term | expression "+" term | expression "-" term
term ::= factor | term "*" factor | term "/" factor
factor ::= IDENTIFIER | constant | "(" expression ")"
constant ::= integer | boolean
condition ::= expression RELOP expression | "!" condition | condition "&&" condition | condition
"||" condition
RELOP ::= "<" | "<=" | "=" | ">=" | ">"

Example Program: Find the First K Prime Numbers

```
input(k);
count := 0;
n := 2;

while count < k do {
   isPrime := true;
   i := 2;

   while i * i <= n do {
      if n % i = 0 then {
         isPrime := false;
      }
      i := i + 1;
   }

   if isPrime = true then {
      output(n);
      count := count + 1;
   }

   n := n + 1;
}
```

-