#### **1. Language Definition**

##### **1.1 Alphabet:**

* **Uppercase (A-Z) and lowercase (a-z) letters**
* **Digits (0-9)**
* **Special symbols**: +, -, \*, /, %,^,:=, <, <=, =, >=, (, ), ;, {, }, [ , ], and whitespace

##### **1.2 Lexical Elements:**

* **Operators**: +, -, \*, /, :=, <, <=, =, >=
* **Separators**: (, ), {, }, [, ], ;, (space)
* **Reserved Words**:
  + int, bool, struct, if, else, while, do, read, write, begin, end,then, true, false
* **Identifiers**: A sequence of letters and digits, beginning with a letter
  + **Rule**: identifier ::= letter { letter | digit }
  + **Letter**: A single uppercase or lowercase letter
  + **Digit**: A single digit from 0 to 9
* **Constants**:
  + **Integer constants**: integer ::= digit { digit }
  + **Bolean constants**: true | false

#### **2. Syntax Definition (BNF Notation)**

Using BNF notation, we define the syntactic rules:

<program> ::= "begin" <statement-list> "end"

<declaration> ::= "int" <identifier> | "bool" <identifier> | "struct" <identifier> "{" <declaration-list> "}"

///<declaration> ::= ("int" | "bool") <identifier> { "," <identifier> }

<declaration-list> ::= <declaration> ";" { <declaration> ";" }

<statement-list> ::= <statement> ";" { <statement> ";" }

<statement> ::= <assignment> | <io-statement> | <conditional> | <loop>

//

[

<statement> ::= <assignment> | <io-statement> | <conditional> | <loop> | <compound-statement>

<compound-statement> ::= "begin" <statement-list> "end"

]

<assignment> ::= <identifier> ":=" <expression>

<io-statement> ::= "read" "(" <identifier> ")" | "write" "(" <identifier> ")"

<conditional> ::= "if" <condition> "then" <statement> ["else" <statement>]

<loop> ::= "while" <condition> "do" <statement>

<expression> ::= <term> { ("+" | "-") <term> }

<term> ::= <factor> { ("\*" | "/") <factor> }

<factor> ::= <identifier> | <constant> | "(" <expression> ")" | "true" | "false"

<condition> ::= <expression> <relation> <expression>

<relation> ::= "<" | "<=" | "=" | ">" | ">="

Program all prime < k

begin

int i, j, k;

bool is\_prime;

read(k);

i := 2;

while i < k do

begin

is\_prime := true;

j := 2;

while j \* j <= i do

begin

if i % j = 0 then

is\_prime := false;

j := j + 1;

end;

if is\_prime = true then

write(i);

i := i + 1;

end;

end