Language definition

Alphabet:

Capital and small letters: A, B, C, …, Z, a, b, c, …, z
Decimal digits: 0, 1, …, 9

Lexic

Special symbols:

Operators: + - * / % =
Separators: [] () {} ; : ?
Reserved words: if else while break let of return
and or
const int char
write read

Identifiers:

identifier ::= lowLetter | {lowLetter} {upperLetter} {digit}
upperLetter ::= “A” | “B” | … | “Z”
lowLetter ::= “a” | “b” | … | “z”
digit ::= “0” | “1” | … | “9”
zero ::= “0”
nonZero ::= “1” | “2” | … | “9”

Constants:

constant ::= integer | “-”nonZero{digit}
int ::= nonZero{digit} | zero
char ::= lowLetter | upperLetter | digit
string ::= char{string}

Syntactical rules:

program ::= "VAR" decllist ";" cmpdstmt "."
decllist ::= declaration | declaration ";" decllist
declaration ::= IDENTIFIER ":" type
type1 ::= "BOOLEAN" | "CHAR" | "INTEGER" | "REAL"
arraydecl ::= "ARRAY" "[" nr "]" "OF" type1
type ::= type1|arraydecl
cmpdstmt ::= "BEGIN" stmtlist "END"
stmtlist ::= stmt | stmt ";" stmtlist
stmt ::= simplstmt | structstmt
simplstmt ::= assignstmt | iostmt
assignstmt ::= IDENTIFIER ":=" expression
expression ::= expression "+" term | term
term ::= term "*" factor | factor
factor ::= "(" expression ")" | IDENTIFIER
iostmt ::= "READ" | "WRITE" "(" IDENTIFIER ")"
structstmt ::= cmpdstmt | ifstmt | whilestmt
ifstmt ::= "IF" condition "THEN" stmt ["ELSE" stmt]
whilestmt ::= "WHILE" condition "DO" stmt

condition ::= expression RELATION expression
RELATION ::= "<" | "<=" | "=" | "<>" | ">=" | ">"

Lexical rules:

identifier ::= lowLetter | {lowLetter} {upperLetter} {digit}
upperLetter ::= “A” | “B” | … | “Z”
lowLetter ::= “a” | “b” | … | “z”
digit ::= “0” | “1” | … | “9”
RELATION ::= "<" | "<=" | "=" | "<>" | ">=" | ">"

b)

```
program PrimeFinder
VAR
   k : int;
   n : int;
   i : int;
   j : int;
   prime : bool;
BEGIN
   READ(k);
   n := 0;
   i := 2;

   WHILE n < k DO
   BEGIN
      prime := true;
      j := 2;

      WHILE j * j <= i DO
      BEGIN
         IF i % j = 0 THEN
         BEGIN
            prime := false;
            BREAK;
         END;
         j := j + 1;
      END;

      IF prime THEN
      BEGIN
         WRITE(i);
```

```
            n := n + 1;
        END;

        i := i + 1;
    END;
END.
```