p1: compute the max of 3 numbers

```
start
{
define int: a;
define int: b;
define int: c;
define int: max;
read: a;
read: b;
read: c;
max:=a

if max < b then {max:=b}
if max < c then {max:=c}

display("The maximum number is ", max);
}
end
```

p1err: compute the max of 3 numbers with syntax errors

```
start
{
define int: a;
define int: b;
define int: c;
define int: max;

read: a;
read: b;
read: c;
max:=a;
if max < b then {max:=1b}  //identifier starts with a digit
if max < c then {max:=c}


display("The maximum number is ", "max"); //it will print the word max
not the value
}
end
```

p2: check if a number is prime

```
start
{
define int: x,i;
read: x;
i:=3;
if x == 2 then {display(x, "is a prime number");}

while(i * i <x) do
{ if n%i == 0 then {display(x, "is not a prime number"); stop;}
  i++;
}

display(x, "is a prime number");

}
end
```

```
p3: compute the sum of n numbers from an array
start
{
define int: n;
define int: sum;
define int: i;
define array: int[10];
sum:= 0;
i:= 0;

read: n;
while(i<n) do
{
read: array[i];
sum:= sum + array[i];
i++;
}
display("The sum of the numbers is ", sum);
}
end
```

Alphabet:

a. Upper (A-Z) and lower case letters (a-z) of the English alphabet
b. Underline character '_';
c. Decimal digits (0-9);

Lexic:

a. Special symbols, representing:

- operators: + - * / := < > <= >= == !=
- separators: [ ] { } ; : space ( )
- reserved words: start end define int char string array display read if then else while stop

b. Identifiers = a sequence of letters and digits s.t. the first character is a letter; the rule is:

identifier = letter | letter {digit | letter}
letter = "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"
digit = "0" | "1" | ... | "9"

c. Constants

-integer - rule:
int_no = [("+" | "-")] non_zero_digit {digits} | "0"
non_zero_digit = "1" | ... | "9"

- character:
character_const = 'letter'|'digit'|'symbol'
symbol = ":" | ";" | "?" | "!" | "."

- string:
string = "character{character}"
character = letter | digit | symbol

Sintactical rules:

<program>::= "start" <compound_statement> "end"

<compound_statement>::= (<declaration> | <array_declaration_statement> | <statement>) ";" [<compound_statement>]

<declaration>::= "define" <type> ":" IDENTIFIER ";"

<type>::= "int" | "char" | "string"

constant_value = int_no | character | string

<array_declaration_statement>::= "define" <type> ":" IDENTIFIER

<statement>::= <assignment_statement> | <io_statement> | <if_statement> | <while_statement> | "stop" ";"

<io_statement>::= <read_statement> | <write_statement>

<assignment_statement>::= IDENTIFIER ":=" <expression> ";"

<read_statement>::= "read" ":" IDENTIFIER ";"

<write_statement>::= "display" "(" <expression> ")" ";"

<expression>::= <term> [("+" | "-") <expression>]

<term>::= <factor> [("*" | "/") <term>]

<factor>::= IDENTIFIER | constant_value "(" <expression> ")"

<if_statement>::= "if" <condition> "then" "{" <compound_statement> "}" ["else" <compound_statement>]

<while_statement>::= "while" "(" <condition> ")" "do" "{" <compound_statement> "}"

<condition>::= <expression> <RELATION> <expression>

<RELATION>::= "<" | ">" | "<=" | ">=" | "==" | "!="

+
-
*
/
<
>
:=
<=
>=
==
!=
[
]
{
}
;
:
(
)
start
end
define
int
char
string
display
read
if
then
else
while
stop