```
## maximum of 3 numbers
CREATE_PROGRAM;

SPAWN nr a;
SPAWN nr b;
SPAWN nr c;
SPAWN nr max;

max <- 0;
a <- IN("Type first number: ");
b <- IN("Type second number: ");
c <- IN("Type third number: ");

CHECK a GREATER_THAN max =>
@ max <-a; @

CHECK b GREATER_THAN max =>
@ max <- b; @

CHECK c GREATER_THAN max =>
@ max <- c; @

OUT("Maxium number is: ", max);

DESTROY_PROGRAM;
```

```
## check if a number is prime
CREATE_PROGRAM;

SPAWN nr number;
SPAWN nr divisor;
SPAWN nr halfNumber;
SPAWN flag isPrime;

number <- IN("Type number: ");
divisor <- 2;
halfNumber <- numer / 2;
isPrime <- true;

LOOP divisor LESS_OR_EQUAL_THAN halfNumber AND isPrime IS true =>
@
 CHECK number % divisor EQUAL 0 =>
 @ isPrime <- false; @
@

CHECK isPrime IS true =>
@ OUT("The number is prime"); @

CHECK isPrime IS false =>
@ OUT("The number is false"); @

DESTROY_PROGRAM;
```

```
## compute sum of n numbers
CREATE_PROGRAM;

SPAWN nr numbers;
SPAWN nr var;
SPAWN nr index;
SPAWN nr sum;

numbers <- IN("Input maximum number of variables: ");
index <- 0;
sum <- 0;

LOOP index LESS_THAN numbers =>
@
 var <- IN("Type number: ");
 index <- index + 1;
 sum <- sum + var;
@

OUT("Sum is: ", sum);

DESTROY_PROGRAM;
```

```
## check if a number is prime, contains lexical errors
CREATE_PROGRAM;

SPAWN nr number;
SPAWN nr divisor;
SPAWN nr halfNumber;
SPAWN flag isPrime;

number <- IN("Type number: ");
divisor <- 2;
halfNumber <- numer / 2;
isPrime <- true;

LOOP divisor LESS_OR_EQUAL_THAN halfNumber AND isPrime = true =>
@
 CHECK number % divisor = 0 =>
 @ isPrime <- false; @
@

CHECK isPrime IS true =>
@ OUT("The number is prime"); @

CHECK isPrime IS false =>
@ OUT("The number is false"); @

DESTROY_PROGRAM;
```