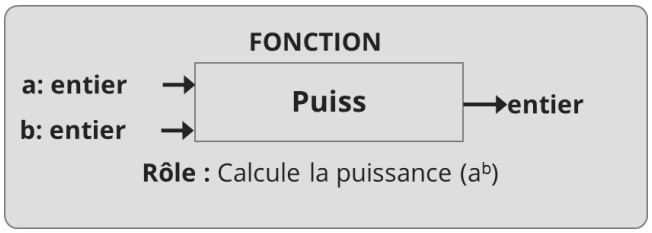
SOLUTION to TUTORIAL N°1 : MODULARITY
Functions and procedures

Exercise 4 : Write a function ***power*** that computes a^b = a x a x a x ...x a (b times); a and b being positive integers.

Solution:



Function Puiss :

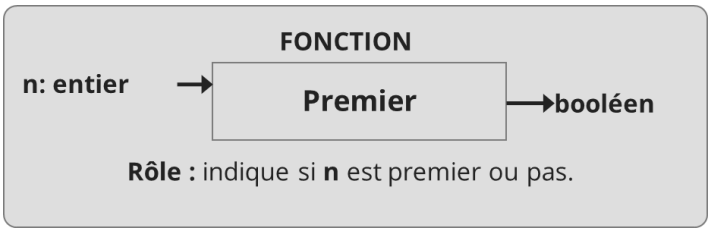
```
Function Puiss (a,b: integer): integer;
Var    P, i: integer;
Begin
   P ← 1;
   For i ← 1 to b Do  P ← P * a;

   Puiss ← P;
End;
```

Exercise 5: Knowing that a prime number is a number that has no divisor other than 1 and itself.

1. Write a function « ***prime*** » that indicates whether an integer N is prime or not.
2. Build the algorithm that gives us the first N prime numbers.

Solution:



Function Premier :

```
Function Premier (a,b: integer): integer;
Var    P, i: integer;
       continue: boolean;
Begin
   continue ← true;
   i ← 2;
   While (continue == true) and (i < n DIV 2) Do
      Dwhile
         If n MOD i = 0 Then  continue ← false;
         Else  i ← i+1;
      Endwhile
   If continue = true Then Premier ← true;
   Else                    Premier ← false;
End;
```

Main Algorithm: that displays the first N prime numbers

```
Algorithm exo_2;
Var i, cpt: integer;
Functions  Premier;

Begin
    cpt ← 0;
    i ← 1;
    While (cpt < 10) Do
       Dwhile
            If Premier(i) = true Then
                  cpt ← cpt +1;s
                  Write(i, ' is prime');
            Else
                  i ← i+1;
            Endif
       Endwhile
End.
```

Exercise 6: Starting from an integer N, we want to obtain two other numbers N1 and N2. The first (N1) will consist of the even digits of N and the second (N2) of the odd digits.

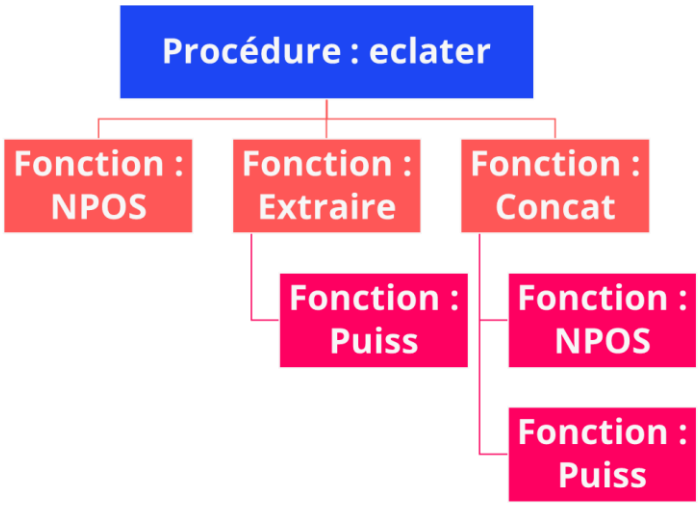
Examples:	N = 25461327	N1 = 2462	N2 = 5137
	N = 42613786	N1= 42686	N2 = 137
	N = 240682	N1 = 240682	N2 = 0

1. Write an algorithm that returns numbers N1 and N2 from a number N.

N.B.: The solution must include at least one function and one procedure.

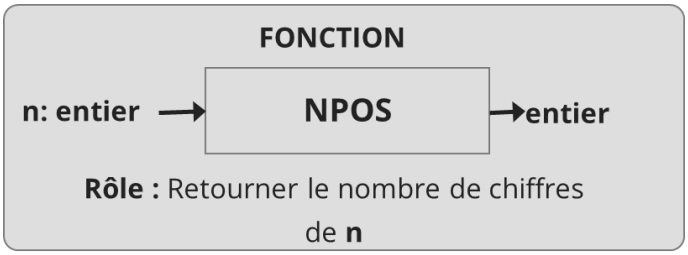
Solution:

Modular Decomposition:



1. **Function NPOS:**

✓ Computes the number of digits of an integer n (number of positions).



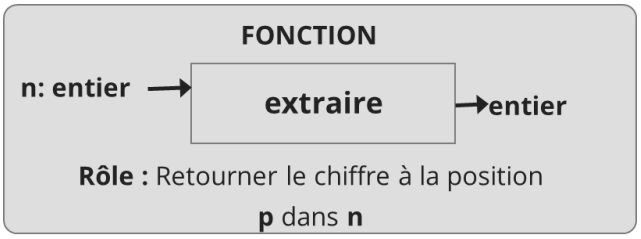
Examples: NPOS (0) = 1
NPOS (2) = 1
NPOS (142) = 3

```
Function NPOS (n: integer): integer;
Var    P, i: integer;

Begin
    P ← 0;
    repeat
        P ← P + 1;
        n ← n DIV 10;
    until n = 0;
    NPOS ← P;
End;
```

2. Function Extract:

✓ Extracts a digit (integer) at a given position from an integer n.



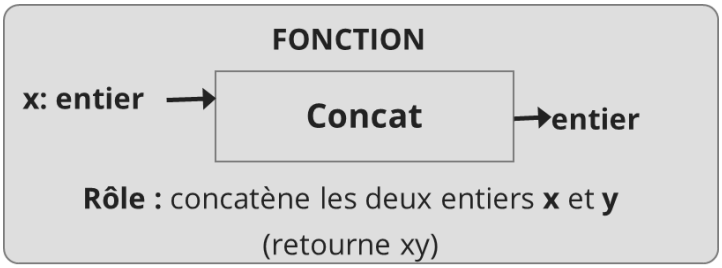
Examples:
```
Extraire (1983,3)   = 9
Extraire (1983,1)   = 3
```

```
Function Extraire (n,p: integer): integer;
Functions  Puiss;

Begin
    Extraire ← (n DIV Puiss(10, p-1)) MOD 10;
End;
```

3. Function Concats:

✓ Concatenates (or merges) two integers x and y (and returns xy).

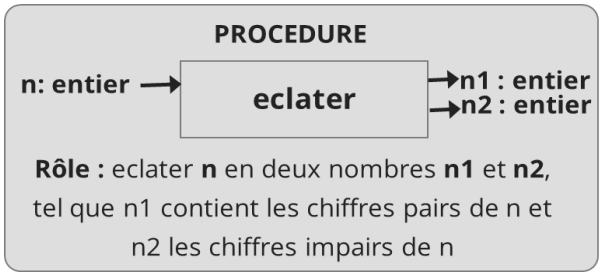


Examples:
```
Concat (19,83)   = 1983
Concat (0,12)    = 12
Concat (12,0)    = 120
```

```
Function Concat (x,y: integer): integer;
Functions  Puiss, NPOS;

Begin
    Concat ← x * Puiss(10, NPOS(y)) + y;
End;
```

4. Procedure eclater:



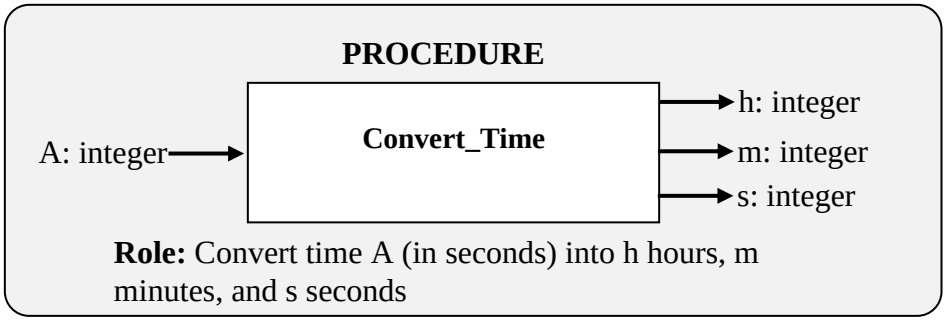
```
Procedure  eclater(n: integer, VAR n1,n2: integer);
Var  i, c: integer;
Function  NPOS, Extraire, Concat;
Begin
    n1 ← 0; n2 ← 0;
    For i ← 1 to NPOS(n) Do
        Dfor
            c ← Extraire(n,i);
            If c MOD 2 = 0 Then
                n1 ← Concat(c, n1);
            Else
                n2 ← Concat(c, n2);
            Endif
        Endfor
    n1 ← n1 DIV 10;
    n2 ← n2 DIV 10;
End;
```

Exercise 7: Let A be a time in seconds.

1. Write a **procédure** that converts A into hours, minutes, and seconds.

Solution:

1. **Procedure Convert_Time:s**




```
Procedure  Convert_Time (A: integer, VAR h,m,s: integer);
Var  R: integer;

Begin
    h ← A DIV 3600;
    R ← A MOD 3600;
    m ← R DIV 60;
    s ← R MOD 60;
End;
```

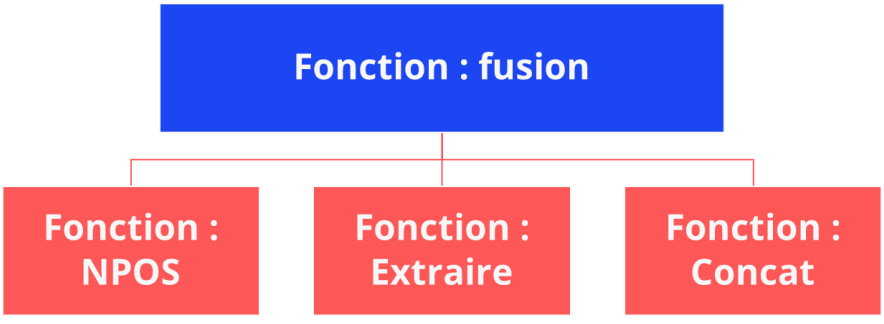
Exercise 8: Given two numbers N1 and N2.

1. Write a **fonction** that merges the digits of two integers N1 and N2 in alternating fashion. Examples:
   N1 = 381 N2 = 946 N= 398416

N.B.: The solution must include at least two functions.

Solution:

Modular Decomposition:



1. **Function merge:**

✓ We assume that the two numbers N1 and N2 have the same number of digits.

```
Function  fusion (N1,N2: integer) : integer;
Var N, i, c1, c2, A : integer;
Function  NPOS, Extraire, Concat;
Begin
   N ← 0;
   For i ← 1 to NPOS(N1) Do
       Dfor
          C1 ← Extraire(N1,i);
          C2 ← Extraire(N2,i);
          A ← Concat(C1, C2);
          N ← Concat(A, N);
       Endfor
   N ← N DIV 10;
   fusion ← N;
End;
```

Exercise 9: (Examen rattrapage Init.Algo S1 2015/2016)

1. Write a procedure **Insert_Chiff (c, p, N)** that inserts a digit **c** in the number **N** at position **p** (the result is N itself).

   Exemple: Insert_Chiff(5, 3, N) with N = 14721 gives the result N = 147521.

Solution:

1. **Procedure Insert_Digit:**

✓ We assume that the two numbers N1 and N2 have the same number of digits.

```
Procedure  Insert_Digit (c,p: integer; VAR N : integer);
Var left, right: integer;
Function  NPOS, Concat;
Begin
   left ← N DIV PUISS(10, p-1);
   right ← N MOD PUISS(10, p-1);
   N ← Concat(c, right);
   N ← Concat(left, N);
End;
```

Exercise 15: Fill and Display a Matrix

Let A be a matrix containing integer values with n rows and m columns.

1. Write a procedure ***Read_Mat*** that fills – or reads – matrix A.
2. Write a procedure ***Write_Mat*** that displays – or writes – the contents of matrix A.

Solution:

```
Type matrix: Array [1..100,1..100] of integer;
```

2. **Procedure Read_Mat:**

```
Procedure Read_Mat (n, m: integer; VAR A: matrix);
Var i, j: integer;

Begin
   //1. Fill matrix A
   For i from 1 to n Do
      For j from 1 to m Do
         Read(A[i,j]);

End;
```

3. Procedure Write_Mat:

```
Procedure Write_Mat (A: matrix; n, m: integer);
Var i, j: integer;

Begin
   //1. Display the contents of matrix A
   For i from 1 to n Do
      For j from 1 to m Do
         Write(A[i,j]);

End;
```

Exercise 16: 90° Rotation of a Matrix (ASD-II Exam 2020-2021)

Let A be a square matrix of dimension n×n containing integer values.

3. Write an algorithm that performs a 90° clockwise rotation. This is achieved in 2 steps:
   a. Compute the transpose of matrix A, then
   b. Swap the columns (first with last, second with second-to-last, ...)

Exemple : A

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Compute *transpose* →

Transpose of A

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

swap the columns →

Rotation result

13	9	5	1
14	10	6	2
15	11	7	3
16	12	8	4

Solution:

1. Procedure Transpose:

```
Procedure Transpose (A: matrix; n: integer; VAR B: matrix);
Var i, j: integer;
Begin
   For i from 1 to n Do
      For j from 1 to n Do
         B[i,j] ← A[j,i];

End;
```

2. Procedure Swap_Columns:

```
Procedure Swap_Columns (A: matrix; n: integer; VAR B: matrix);
Var i, j: integer;
Begin
   For i from 1 to n Do
      For j from 1 to n Do
         B[i,j] ← A[i,n-j+1];

End;
```

3. Main Algorithm:

```
Algorithm rotation_90;
Type matrix: Array [1..100,1..100] of integer;
Var A,B,C: matrix;
      i, j, n: integer;
Procedures Read_Mat, Write_Mat, Transpose, Swap_Columns;
Begin
   Read(n);
   //1. Fill matrix A
   Read_Mat(n, n, A) ;
   //2. Compute the transpose of A
   Transpose(A, n, B) ;
   //3. Swap the columns of A
   Swap_Columns(B, n ,C) ;
   //4. Display the matrix after rotation
   Write_Mat(A, n, n) ;
End.
```

Exercise 17: 1st non-repeating element

Let **T** be a one-dimensional array of **n** valeurs entists (avec n <1000)

1. Write a function ***Occurrences (T, n, Val)*** that returns the number of occurrences (repetitions) of the value **Val** in the array **T**.
2. Using the previous function, write an algorithm that returns the 1er non-repeating element in the array T.

Exemple : Le 1er non-repeating element est : **8**

2	2	3	8	6	6	7	3	3	2	9	9

Solution:

1. **Function Occurrences:**

```
Fonction Occurances(T: array[1..1000] of integer; n, Val: integer):integer ;
 Var i, cpt : integer;

Begin
   cpt ← 0;

   //Traverse the array and count occurrences
   For i from 1 to n Do
      If (T[i] = Val) Then
         cpt ← cpt + 1;
      Fsi
   Endfor

   Occurances ← cpt
End.
```

2. **Function Main Algorithm:**

```
Algorithm non_repitititif;
 Var   T: array[1..1000] of integer;
       n, i : integer;
 Fonctions    Occurances ;

Begin
   //Fill the array
   Read(n) ;
   For i from 1 to n Do Read(T[i]) ;

   //Traverse the array and find the element with '1' occurrence
   i ← 0 ;
   Répéter
      i ← i + 1;
   Until (i > n or Occurances(T, n, T[i]) = 1)

   Write('the first non-repeating element is', T[i]);
End.
```

Exercise 18: (PSD S2 Exam 2013/2014)

Let T be a one-dimensional array of **N** valeurs entists (N <= 100)

1. Write a function **Tab_Rech (T, N, val)** that searches for an integer value **val** in the array **T** and returns its index in the array **T** if it exists (returns the index of the first occurrence), otherwise returns a negative value.
2. Write a procedure **Tab_sans_double (T1, N, T2)**, using the previous function, that builds an array **T2** containing only one occurrence of each number in **T1** de **N** elements.

Solution:

```
Type Tab = array [1..100] of integer;
```

3. Function Tab_Rech

```
Function Tab_Rech (T : Tab ; N, val: integer) : integer
    Var i: integer;
        Found: boolean;
Begin
    Found ← false;
    While (i <= N) et (Found = false) Do
       DWhile
           If (T[i] = val) Then Found ← true;
           Else      i ← i + 1 ;
       FWhile

    If (Found = true) alors       Tab_Rech ← i ;
    Else                          Tab_Rech ← -1 ;
End.
```

4. Procedure Tab_sans_double

```
Procedure Tab_sans_double (T1 : Tab ; N: integer, VAR T2 : Tab)
    Var i, j: integer;
Begin
    j← 0 ;
    For i ← 1 to N Do
       DFor
           If (Tab_Rech(T2, j, T1[i]) < 0 ) alors
              DSi
                     j ← j + 1 ;
                     T2[j] ← T1[i] ;
              FSi
       FFor
End.
```