Larbi Ben M'hidi-Oum El Bouaghi- University
Faculty of Exact Sciences and Natural and Life Sciences
Mathematics and Computer Science Department

| | | |
|---|---|---|
| **Level:** 1st year "Computer science" | **Module:** Algorithmic and Data Structures 2 | **Date :** 15/05/2024<br>**Duration**: 1h30m |

# Exam n°2

## Typical correction

### Exercise n°1

1. **Procedure Admitted_Students** (T: [1..100] array of integer, N: integer) (1 pt)

```
Var i: integer;
    Begin
      For  i←1 to N do
            If T[i] >= 10  then
             Write ("Student",  i,  " is admitted");
            End If
      End for
    End ;
```

2. **Procedure Non_Admitted_Students** (T: [1..100] array of integer, N: integer) (1 pt)

```
    Var i: integer;
    Begin
      For  i←1 to N do
            If T[i] <10  then
             Write ("Student",  i,  " is not admitted");
            End If
      End for
    End ;
```

3. **Function Sum_Grades** (T: [1..100] array of integer, N: integer): integer (1 pt)

```
    Begin
        If N = 0 then  Sum_Grades ← 0;
        Else Sum_Grades ← T[N] +  Sum_Grades (T, N-1);
        End if;
    End;
```

**Procedure Class_Average** (T: [1..100] array of integer, N: integer) (1 pt)

Var S, M: integer;

Begin

S ← Sum_Grades (T, N);

M ← S/N;

Write (" Class_Average is: ", M);

End;

4. **Procedure Highest_Grade** (T: [1..100] array of integer, N: integer) (1 pt)

```
Var: i, max: integer;
Begin
max ← T[1];
    For  i←2 to N do
        If T[i] >= max then
            Max ← T[i];
        End If
    End for
Write ("The Highest Grade is: ", max);
End;
```

5. **Procedure** Lowest_Grade (T: [1..100] array of integer, N: integer) (1 pt)

```
Var: i, min: integer;
Begin
min ← T[1];
    For  i←2 to N do
        If T[i] <= min then
            min ← T[i];
        End If
    End for
Write ("The Lowest Grade is: ", max);
End;
```

6. **Algorithm Exercise_1;**

**Var: T**: array [1..100] of integer; N, i integer; **(0.5 pt)**

**Copy the previous procedures into this section**

```
Begin
    Repeat
        Write ("Enter the real dimension of the vectors");
        Read(N);
    Until (N>=1 and N<=100)                                   ( 0.5 pt)
  Write ("Enter the elements of the vector T");
   For  1←i to N do
      Read (T[i]);
   Endfor;

   Admitted_Students(T, N);
   Non_Admitted_Students(T, N);
   Class_Average(T, N);                                       (1  pt)
   Highest_Grade(T, N);
   Lowest_Grade(T, N);
   END.
```

7. **No, we can't;** **(0.5 pt)**

   **Justification**: The variables "**max** and **min**", declared to identify the Highest Grade and the Lowest Grade, **are local variables** . Indeed, they cannot be used in the main algorithm to compare between **max** and **min** because the space of these local variables is freed when the execution of the procedure is finished.

   **Solution**: there are three possible solutions **(1 pt)**

   1. Use **max** and **min** as parameters and pass them by variable (var max, min).
   2. Declare **max** and **min** as global variables.
   3. Convert the two procedures: Highest_Grade and the Lowest_Grade into **functions**

8. This case requires the use of **linked lists**. **(0,5 pt)**

**Exercise n°2**

```
Algorithm Exercise_2 ;
  Var  T : array [1..100] integer ; P, P1:*integer ;  (0.5 pt)
Begin
For P=T to (T+100) do  (1.5 pt)
    Read (*P);
Endfor
```

Allocate (P1); **(1 pt)**

*P1←1;

For P=T to (T+100) do

  If ((*P) mod 3 = 0) then

    Write("The index  is",P-T);

    *P1←*P1 * *P;

  End If

Endfor

**End**

**(2 pts)**

**Exercise n°3**

**Algorithm** Exercise_3 ;

**Type** Computer=Record **(1 pt)**

serial_number: integer;

brand: string(20);

model: string(20);

price: real;

**EndRecord**

**Var:** **(0.5 pt)**

C : array [1..20] of Computer;

i : integer;

**Begin**

Write ("Enter the information about the Computers");

**For** 1←i to 20 do **( 1.5 pt)**

Read (C[i]. serial_number);

Read (C[i]. brand);

Read (C[i]. model);

Read (C[i]. price);

**Endfor**

**For** i←1 to 20 do **( 2 pts)**

If (C[i].price > 50000) then

Write (C[i].serial_number, C[i].brand, C[i].model,  C[i].price);

endif

**Endfor**

**END**