

Level: 1st year “Computer Science”
Module: Algorithmic and Data Structures 1

Date: 16/01/2025
Duration: 1h30m

Exam n°1

“Typical correction”

General questions

(5 points)

(0.5 point for each response). Complete the following expressions

1. The objective of algorithm: **solving the given problem in a systematic way.**
2. The objective of data structure: **stores and organizes different data in computer memory. It ensures the reusability of data efficiently later.**
3. In algorithmic, the basic instructions are: **Read (input), Write (output) and assignment (←).**
4. The number of executions for the Repeat loop is **>=1** and for the While loop is **>=0**.
5. In RAM, the elements of an array are organized **contiguously.**
6. In algorithmic, a matrix is a **two-dimensional array.**
7. A set of data of different types associated with the same object represents **a record.**
8. In C, the character string is represented by **an array of characters (char).**
9. In loops, the counter **is a variable whose value is incremented by 1 at each step of the loop.**
10. To make a choice among several possible cases depending on the value of a variable, we use the instruction **switch-case / multiple choice.**

Exercise n°1

(5 points)

Write an algorithm that asks the user to choose the class of his seat for a flight from Constantine to Algiers. Enter the letter (character) of the desired class

P → First Class; reservation fees 12000 AD.

E → Economy Class; reservation fees 6000 AD.

Then, the algorithm asks the user to weigh the luggage and displays the total amount to be paid by adding the reservation fees to the amount to be paid for the luggage, knowing that the price of 1 kg is 100 AD. Transform the algorithm into C program.

Algorithm flight_Constantine_Algiers

Variables class: character; weigh, fees: integer;	}	(0.5 point)
Begin		
Write(“Enter the desired class”);	}	(0.5 point)
Read (class);	}	
Write (“Enter the luggage weigh”);	}	(0.5 point)
Read (weigh);	}	
If (class='P') then		
fees= 12000+(100* weigh);	}	(2 points)
else if (class='E') then	}	

```

    fees= 6000+(100* weigh);
endif
endif
Write ("The total to be paid is", fees);
END

```

- C program **(1.5 point)**

Exercise n°2 **(3 points)**

Propose an algorithm that requests a positive integer and displays the value of the greatest power of 2 that is less than or equal to that number (using **pow** function). Examples:

User enters 19 → Result: **4**, because ($2^4=16$). User enters 73 → Result: **6**, because ($2^6 = 64$).

Algorithm power

Variables n, i : integer; **(0.25 point)**

Begin

Write ("Enter an integer :"); **(0.25 point)**

Read (n);

i ← 0 ; **(1.5 point)**

While (pow (2,i)<= n) do

i ← i+1;

EndWhile

Write ("The greatest power is : ", i-1); **(1 point)**

END

Exercise n°3 **(7 points)**

Consider an array T1 of size Nmax = 50. Write an algorithm where you declare this array, ask for an integer *n* that represents the real size of the array (number of cells to fill) then perform the following operations:

1. Enter the *n* numbers in the array then display them;
2. Find if there is a prime number in the array or not;
3. Find if there is a perfect number in the array or not;

Consider a second array T2 of size Mmax=100, contains integers.

4. Enter the elements of T2, and then display the elements of T1 that do not exist in T2.
-

Algorithm arrays_manipulation;

Variables T1: array[1..50] integer; T2: array[1..100] integer;

n, m, i, j, nbdiv, div: integer;

perfect_exist, prime_exist, exist: boolean;

Begin

Write ("Enter the real dimension of T1");

Repeat




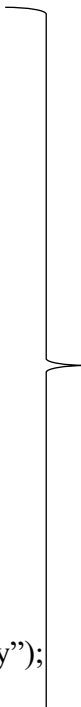
Read (n);

Until (n>=1 and n<=50)

For i←1 to n do

(0.75 point)

(0.5 point)

<pre> Write ("Enter the element n°", i); Read (T1[i]); Endfor For i ← 1 to n do Write ("The element n°", i,"is"); Write (T1[i]); Endfor </pre>		(0.5 point)
<pre> For i ← 1 to n do Write ("The element n°", i,"is"); Write (T1[i]); Endfor </pre>		(0.5 point)
<pre> prime_exist ← false; For i ← 1 to n do nbdiv ← 0; //number of divisors For j ← 2 to (T1[i]/2) do If (T1[i] mod j=0) then nbdiv ← nbdiv+1; Endif Endfor If (nbdiv =0) then prime_exist ← true; Endif Endfor If (prime_exist=true) then Write ("There is a prime number in the array"); Else Write ("A prime number does not exist in the array"); Endif </pre>		(1.25 point)
<pre> perferct_exist ← false; For i ← 1 to n do div ← 0; //sum of divisors For j ← 1 to (T1[i]/2) do If (T1[i] mod j=0) then div ← div+j; Endif Endfor If (div =T1[i]) then perfect_exist ← true; Endif Endfor If (perfect_exist=true) then Write ("There is a perfect number in the array"); Else Write ("A perfect number does not exist in the array"); Endif </pre>		(1.25 point)

Write ("Enter the real dimension of T2");

Repeat

 Read (m);

Until (m>=1 and m<=100)

For i←1 to m **do**

 Write ("Enter the element n°", i);

 Read (T2[i]);

Endfor

(1 point)

For i←1 to n **do**

exist ← false;

For j←1 to m **do**

If (T1[i]=T2[j]) **then**

 exist ← true;

Endif

Endfor

If (exist=false) **then**

Write ("Elements of T1 that do not exist in T2",T1[i]);

Endif

Endfor

(1.25 point)

END