University of Echahid Hamma Lakhdar
Faculty of Exact Sciences
Department of Computer Science

Module: Algo & Data Structure I
University Year: 2025/2026
24 January 2026

# Final Exam

## Course questions (4 pts)

- What is a subprogram ?
- What are the types of subprograms ?

ما المقصود بالبرامج الفرعية ؟
ما هي أنواع البرامج الفرعية ؟

## Exercise N°1 (6 pts)

- What does this Algorithm do?
- What will be the result for X=2, Y=8 ?
- Transform this Algorithm into a subprogram

- ما العمل الذي تقوم به هذه الخوارزمية ؟
- ما هي النتيجة النهائية إذا كانت X=2, Y=8
- حول هذه الخوارزمية إلى برنامج فرعي.

```
Algorithm What_Doing
Variables      X, Y, V : Integer     Z : Real
Begin
        Repeat
          Write(Give the values of  X and Y : ')
          Read(X,Y)
        Until ((X>0) and (Y>0) and (Y mod 2 = 0))
           Z ← X
        For V = 1 to Y do
          If (V mod 2 = 1) then
            Z ← Z * Y
          Else
            Z ← Z / X
          End
        Endfor
        Write ('Resultat = ', Z)
End
```

## Exercise N°2 (10 pts)

An Armstrong number (specifically for 3 digits) is defined as a number that equals the sum of its own digits each raised to the power of three. For example, 153 is an Armstrong number because: $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$

نقول عن عدد يتكون من ثلاثة أرقام أنه من أعداد أرمسترنج إذا كان مساويا لمجموع مكعبات كل ارقامه.

Write an algorithm that :

- Find all number of armestrong in a positive range [A .. B] given by the user and put them in ascending order in a table T.
- Look within the table T and show those that are prime if exist.

أكتب خوارزمية تقوم بما يلي :

- تبحث عن أعداد أرمسترنج في مجال [A .. B] موجب يحدده المستخدم ويقوم بوضعها في جدول T بترتيب تنازلي.
- تبحث ضمن الجدول وتظهر على الشاشة الأعداد الأولية من بينها ان وجدت.

# Correction of the Final Exam

## Course questions (4 pts)

- A **subprogram** is a self-contained block of code designed to perform a specific task within a larger program. (2 pts)
- There is two type of subprograms that are: (2 pts)
  - **Functions:** Execute a task (generally, a calculation) and **return a value** (e.g., `square(x)`).
  - **Procedures:** Execute a task (general actions (like printing),…etc) without returning a value.

## Exercise N°1 (6 pts)

- The Algorithm read two positive values X and Y given by the user and Y is odd. (2 pts)
  Then, calculate :

$$Z = X.(Y/X)^{Y/2} = Y^{Y/2}/X^{(Y-2)/2} = Y.(Y/X)^{(Y-2)/2}$$ (1 pt)

- The result $Z = 54$ (1 pt)
- The corresponding subprogram is:

**Function Compute** ( X, Y : Integer) : Real (2 pts)
**Variables** i : Integer Z : Real
**Begin**
    Z ← X
    **For** i = 1 to Y **do**
        If (i mod 2 = 1) then
            Z ← Z * Y
        Else
            Z ← Z / X
        End
    **Endfor**
    **Return** Z
**End**

## Exercise N°2 (10 pts)

1 pt

Algorithm ArmstrongPrimesTable
Variables T[100] : table of integer A, B, i, count, temp, sum, digit : integer

1 pt

- النقطة الاولى لبنية الخوارزمية
- النقطة الثانية للتصريح الصحيح بالمتغيرات
- النقطتين الموالتين لدالة التحقق من أولية العدد (ليس بالضرورة ان تكون في شكل برنامج فرعي)
- النقطة الموالية لقراءة حدود المجال
- النقطتين الموالتين للبحث عن أعداد أرمسترنج
- النقطة الموالية لإدراج اعداد أرمسترنج في الجدول
- النقطة الموالية لترتيب عناصر الجدول (يمكن ان يبحث الطالب في المجال بشكل عكسي وبالتالي يحصل مباشرة على جدول مرتب تنازليا)
- النقطة الموالية لعملية البحث عن الأعداد الأولية في الجدول

```
Function IsPrime(n: integer): boolean
var i: integer     result : boolean
begin
        if (n < 2) then
            result ← false
        else
            result ← true
              i ←  2
            while ((result = true) and (i * i ≤ n)) do
                  if n mod i = 0 then
                        result ← false
                  endif
                  i ← i + 1
            endwhile
        endif
            return result
end

begin
        Repeat
            write('Enter A, B: ')      read(A,B)     { قراءة حدود المجال للبحث عن أعداد أرمسترونج}
        Until ((A ≥ 100) and (B ≥ A))
            count ← 0              {  لحساب عدد أعداد أرمسترونج}
        for i = A to B do              { البحث عن أعداد أرمسترونج وإدراجها في الجدول }
            sum ← 0      temp ← i
                  while (temp > 0) do
                      digit ← temp mod 10     sum ← sum + (digit * digit * digit)   temp ← temp div 10
                  endwhile
                  if (sum = i) then
                      count ← count + 1        T[count] ← i
                  endif
        endfor
                              { الترتيب التنازلي للجدول إذا كان يحتوي على عناصر }
        if (count > 0) then
                  for i = 1 to count div 2 do
                       temp ← T[i]
                       T[i] ← T[count + 1 - i]
                       T[count + 1 - i] ← temp
                  endfor
                              { البحث عن الأعداد الأولية داخل الجدول وعرضها }
                  for i = 1 to count do
                            if (IsPrime(T[i] = true) then
                                write(T[i])
                            endif
                  endfor
        else
                 writeln('No Armstrong numbers found')
        endif

end
```

2 pts

2 pts

1 pt

1 pt

1 pts