1 Ex1 : CO1 (5pt)

1.1 (2pt)

(1pt) Appliquer le Master Theorem sur les cas suivants :

- **(0.5pt)** $T(n) = 2T(n/2) + n$
- **(0.5pt)** $T(n) = T(2n/3) + 1$

(1pt) Montrez que :

- **(0.5pt)** $n^2 logn = O(n^3)$
- **(0.5pt)** $5n^3 - 10n^2 + 2n = O(n^3)$

1.2 (1pt)

Quelle est la complexité de ce programme.

for *(int i=0 ;i<n ;i++)* **do**
    for *(j=0 ;j<i ;j++)* **do**
        printf("hello") ;
    end
end

Algorithme 1 : *DFor*

1.3 (1pt)

Quelle est la complexité de ce programme.

for *(i=0 ;i<n ;i++)* **do**
    for *(j=0 ;j<n ;j++)* **do**
        printf("hello")
    end
    for *(k=0 ;k<n ;k++)* **do**
        printf("hello")
    end
end

Algorithme 2 : *TFor*

1.4 (1pt)

Quelle est la complexité de ce programme.

$i = 0$
while $(i < n)$ **do**
    $i + +$
end

Algorithme 3 : *Wedo*

2 Ex2 : CO1 (5pt)

2.1 (2pt)

Quelle est la sortie de l'algorithme *Algo(T,x)* et quelle est sa complexité (utiliser la notation $\mathcal{O}$) en fonction de la taille n de tableau T.

$a \leftarrow 1$
$b \leftarrow n$
while $b \geq a$ **do**
    $j \leftarrow \frac{a+b}{2}$
    if $x = T[j]$ **then**
        Retourner oui
    else
        if $T[j] < x$ **then**
            $a \leftarrow j + 1$
        else
            $b \leftarrow j - 1$
        end
    end
    Retourner non
end

Algorithme 4 : Algo(T,x)

2.2 (2pt)

Soit la fonction suivante, *Atrouver(A,s,f,e)*.

if $s = f$ **then**
    if $A[s] = e$ **then**
        retourner 1
    else
        retourner 0
    end
end
m=(s+f)/2
gauche=Atrouver(A,s,m,e)
droite=Atrouver(A,m+1,f,e)
retourner gauche + droite

Algorithme 5 : entier : Atrouver(A,s,f,e)

Quelle est la valeur retournée par cette fonction.

2.3 (1pt)

L'algorithme suivant ayant comme entrée un tableau T de nombres entiers, de taille n, et comme sortie une variable binaire r (oui/non).

for *i allant de 2 à n* **do**
    if $T[i-1] > T[i]$ **then**
        $r \leftarrow non$
    end
end
Retourner r

Algorithme 6 : *Afind(T,n)*

Que fait cet algorithme.

3 Ex3 : CO2 (10pt)

Soit T un tableau de n nombres entiers, de dimensions $2 \times n$. Par exemple, si $n = 7$, un tel tableau pourrait être par exemple de la forme suivante :

$$\begin{pmatrix} 5 & 4 & 12 & 9 & 13 & 15 & 8 \\ 6 & 3 & 13 & 14 & 16 & 6 & 4 \end{pmatrix}$$

L'objectif est de trier ce tableau comme suit. Dans chaque ligne et dans chaque colonne, les nombres sont rangés dans l'ordre croissant (de gauche à droite et de haut en bas, respectivement).

$$\begin{pmatrix} 3 & 4 & 5 & 6 & 9 & 12 & 13 \\ 4 & 6 & 8 & 13 & 14 & 15 & 16 \end{pmatrix}$$

(8pt) Donner un algorithme permettant de produire une version triée du tableau. L'algorithme a comme entrée le tableau T et sa dimension horizontale n. Vous pouvez utiliser :

- le sous-algorithme tri par *insertion(T, n)* d'un tableau T de taille n.
- la notation $T[1]$ et $T[2]$ pour désigner respectivement les première et seconde lignes du tableau T.

(2pt) Donner la complexité en $\mathcal{O}$ de cet algorithme.