1 Ex1 : CO1 (4pt)

1.1 (2pt)

(1pt) Appliquer le Master Theorem sur les cas suivants :
- **(0.5pt)** $4T(n/4) + n$
- **(0.5pt)** $4T(n/4) + n^2$

(1pt) Pour chacune des fonctions suivantes, déterminer sa complexité asymptotique dans la notation Grand-O. Exemple : $3n \in O(n)$. Justifier votre réponse.
- **(0.5pt)** $6n^2 + 10n + 5$
- **(0.5pt)** $4log_2 n + 2n$

1.2 (2pt)

Quelle est la complexité de l'algorithme suivant. Utiliser la notation ($\mathcal{O}$) en fonction de la taille n de tableau T. Justifier votre réponse.

```
1  a ← 1
2  b ← n
3  while b ≥ a do
4  |  j ← (a+b)/2
5  |  if x = T[j] then
6  |  |  Retourner oui
7  |  else
8  |  |  if T[j] < x then
9  |  |  |  a ← j + 1
10 |  |  else
11 |  |  |  b ← j − 1
12 |  |  end
13 |  end
14 |  Retourner non
15 end
```

Algorithme 1 : Algo(T,x)

2 Ex2 : CO1 (6pt)

2.1 (4pt)

Soit T un tableau de nombres entiers positifs tous différents les uns des autres et ordonnés dans l'ordre croissant. Soit également n la taille de tableau et e un nombre entier positif. Ecrire un algorithme $Rechercher(T, n, e)$, de complexité $\mathcal{O}(n)$, qui prenne en entrée le tableau T, sa taille n ainsi que le nombre e, et dont la sortie soit le nombre de paires $\{i, j\} \subset \{1, ..., n\}$, avec $i \leq j$, telles que $(T[i] + T[j]) = e$. Par exemple, si $T = (2, 4, 6, 8, 9, 10)$, $n = 6$ et $e = 14$, alors la sortie de $Rechercher(T, n, e)$ doit être 2 ($i = 2, j = 6$ et $i = 3, j = 4$).

2.2 (2pt)

Supposons maintenant que les nombres de tableau T ne soient pas forcément tous différents les uns des autres, ni triés dans l'ordre croissant. Proposez une amélioration de l'algorithme $Rechercher(T, n, e)$ dont la sortie soit, dans ce cas plus général, la même que celle demandée au point **2.1)**. Quelle est la complexité de ce nouveau algorithme ? (utiliser la notation $\mathcal{O}$)

3 Ex3 : CO2 (10pt)

3.1 (5pt)

On se donne le Tas max de la figure 1 :
- **a) (1pt)** Insérer un élément de clé 21 dans ce tas. Donner le nouveau tas.
- **b) (1pt)** En repartant du tas initial, retirer l'élément de clé maximale. Donner le nouveau tas.
- **c) (3pt)** Donner le résultat (étape par étape) de l'algorithme du tri sur le tas initial.

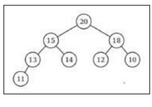

FIGURE 1 – Exemple d'un Tas max

3.2 (5pt)

Décrire le fonctionnement de l'algorithme permettant de trouver les composantes fortement connexes (CFC) sur le graphe de la figure 2. On suppose aussi que la boucle des lignes 5-7 de PP (Parcours en Profondeur) considère les sommets par ordre alphabétique et que les listes d'adjacences sont triées par ordre alphabétique. Lister les composantes trouvées.

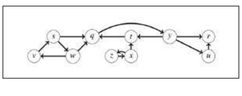

FIGURE 2 – Exemple d'un graphe : CFC