# *Mi-session 2013*

**Question 1** [2 Points] Quel est le pire cas d'exécution pour l'algorithme suivant (en notation Grand O) en tenant compte du nombre d'additions effectuées?

**Algorithme** MyCode1($A$).
Soit $A$ un tableau 3D de dimension $n \times n \times n$, avec $n \geq 300$.

```
for i ← 0 to n do
    for j ← 0 to 200 do
        for k ← 0 to j do
            A[i][j][k] ← i + j + k;
```

**a) O(logn)** **b)O(n)** **c) O(n2)** **d)O(n3)**

**Algorithme** MyCode2($A$).
Soit $A$ un tableau 3D de dimension $n \times n \times n$, avec $n \geq 300$.

```
for i ← 0 to n do
    for j ← 0 to 200 do
        for k ← 0 to i do
            A[i][j][k] ← i + j + k;
```

**a) O(logn)** **b)O(n)** **c) O($n^2$)** **d)O($n^3$)**

**Question 2** [3 Points] Compléter les énoncés suivants :

- $7\sqrt{n} + 3n^3 + 100n^2$ est O( )

- $(\sum_{i=0}^{100}(i \cdot n)) + \log n$ est O( )

- $n^2 + \log n^n$ est O( )

**Question 3** [5 Points] Soit a[] un tableau contenant n entiers triés et soit k un entier, considère les algorithmes suivants **S1** et **S2 :**

**Algorithm S1**(int $a[]$, int $n$, int $k$)

```
OK ← true
i ← 0
while ((i ≤ n − 1) and OK)
{if (a[i] >= k) then OK ← false
 i ← i + 1
}
if (a[i − 1] = k) then return i − 1
                  else return −1
```

**Algorithm S2**(int $a[]$, int $n$, int $k$)

```
beg ← 0
end ← n − 1
mid ← (beg + end)/2
if (a[mid] = k) then return mid
else if (a[mid] > k) then end ← mid − 1
                     else beg ← mid + 1
OK ← true
i ← beg
while ((i ≤ end) and OK)
{if (a[i] >= k) then OK ← false
 i ← i + 1
}
if (a[i − 1] = k) then return i − 1
                  else return −1
```

Que font ces deux algorithmes?

Donner le nombre exact de comparaisons effectués par ces algorithmes et leur caractérisation Grand O en fonction de n pour le pire cas?

**Question 4** [5 Points] un TAD spécial est composé de deux Piles partageant le même espace mémoire. Ce TAD est implémente en utilisant un seul tableau A[0…N-1]. La première pile remplit le tableau par la gauche en débutant à l'index 0 alors que la seconde pile insère les éléments par la droite en débutant N-1. On utilise deux variables : p1 contient l'index de l'élément au-dessus de la première pile et p2 contient l'index de l'élément au-dessus de la seconde pile. Donner l'algorithme pour les méthodes suivantes :

- Size1 : retourne la taille de la première pile.
- Size2 : retourne la taille de la seconde pile.
- estPlein : retourne vrai si le TAD est plein. C.a.d qu'on ne peut pas ajouter d'élément dans aucune des deux piles.
- Depile1 : retire et retourne l'élément au-dessus de la première pile.
- Depile2 : retire et retourne l'élément au-dessus de la second pile

**Question 5** [4 Points] Donner un algorithme O(n) non récursif permettant de renverser une liste simplement chainée de n élément. L'algorithme ne doit pas utiliser qu'un nombre fixe de variables temporaires. Le début de la liste est donné par la variable *tête* et chaque nœud dans la liste aux attributs suivants : *élément* (contenant l'information) et *suivant* pointant au nœud suivant dans la liste.

**Question 6** [3 Points] les énoncés suivants sont incorrects. Identifier et réécrire ces énoncés correctement

1. Dans un arbre binaire implémente a l'aide d'une structure chainée, chaque nœud contient une référence a son élément, son enfant de droite, son enfant de gauche, au nœud suivant de même niveau.

2. Un arbre binaire plein contient des nœuds pouvant avoir un maximum de deux enfants

3. Au niveau i, un arbre binaire peut avoir un maximum de $\log_2(i)$ nœuds.

**Question 7** [6 Points] Considérer le type de données *Deque* avec les méthodes

| Méthodes principales | |
|---|---|
| `insertFirst(e)` | Insère e au début de la structure. |
| `insertLast(e)` | Insère e à la fin de la structure |
| `removeFirst()` | Retire et retourne le premier élément |
| `removeLast()` | Retire et retourne le dernier élément |
| **Méthodes secondaires** | |
| `first()` | Retourne l'élément au début de la structure |
| `last()` | Retourne l'élément à la fin de la structure |
| `size()` | Retourne la dimension de la structure |
| `isEmpty()` | Retourne vrai si la structure est vide |

a) Adapter le TAD *Deque* afin d'implémenter un TAD *Pile* en complétant la table ci-dessous

| **Méthode Pile** | **Méthode Deque** |
|---|---|
| Size () | Size() |
| isEmpty() | isEmpty() |
| top () | |
| push (e) | |
| pop() | |

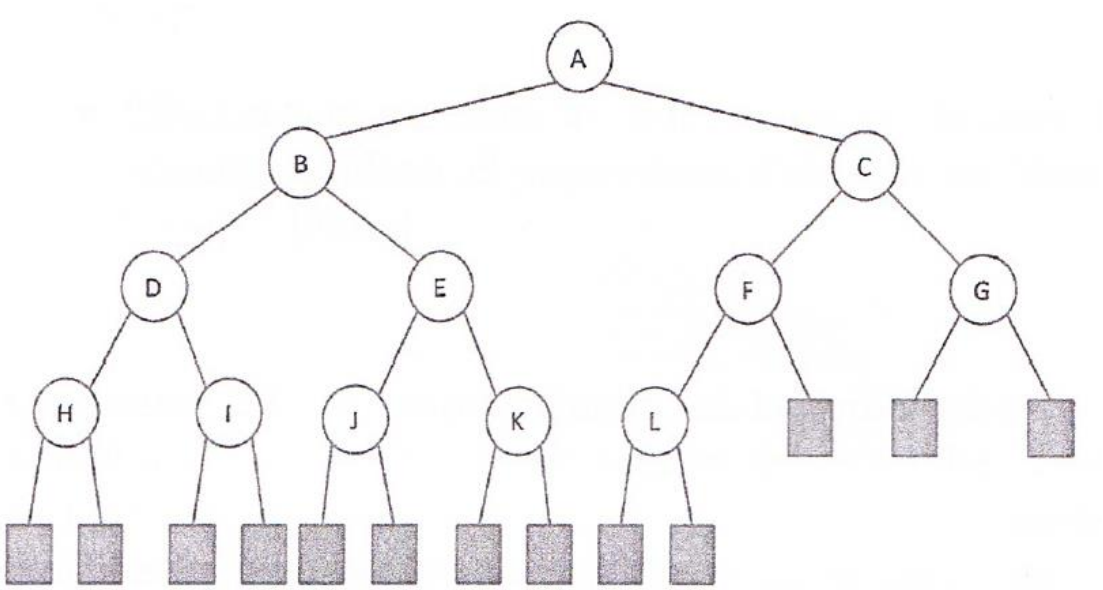
b) Adapter le TAD *Deque* afin d'implémenter un TAD *File* en complétant la table ci-dessous

| **Méthode Pile** | **Méthode Deque** |
|---|---|
| Size() | Size() |
| isEmpty() | isEmpty() |
| Front () | |
| Enqueue (e) | |
| Dequeue() | |

**Question 8** [2 Points] un arbre binaire complet comprend 29 nœuds (incluant les feuilles), combien de nœuds externes a-t-il?

**Question 9** [3 Points] Lister les nœuds de l'arbre ci-dessous dans l'ordre qu'ils seront visités en utilisant :

- Un parcours pré-ordre :
- Un parcours post-ordre :
- Un parcours in-ordre :




**Question 10** [9 Points] Construction du Monceau

Afin de trier sur place et en ordre décroissant la séquence S dans le tableau A en utilisant un tri en monceau, il faut procéder en deux phases

| **Index** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Value** | 6 | 2 | 14 | 13 | 1 | 9 |

- Phase 1) : Construction du Monceau

  En utilisant la stratégie de construction ascendante d'un monceau-min, montrer quel sera l'état du tableau une fois ce monceau construit

| **Index** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Value** | | | | | | |

- Phase 2) : Tri

Montrer l'état du tableau A après que les deux premiers éléments aient été retirés du monceau et après que le dernier élément ait été retiré

| **Index** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| **Après le premier retrait** | | | | | | |
| **Après le second retrait** | | | | | | |
| **Après le dernier retrait** | | | | | | |

**Question 11** [3 Points] Soit un arbre binaire de recherche implémenté dans un tableau A. Vous désirez créer un monceau-min à partir de A. Indiquer si les énonces suivants sont vrais ou faux.

- Effectuer un parcours pré-ordre de l'arbre binaire de recherche A et placer le résultat dans un nouveau tableau B permettra d'obtenir un Monceau-min

[Vrai] [Faux]

- Trier le tableau A en ordre croissant produira un monceau-min dans A

[Vrai] [Faux]

- Effectuer un parcours in ordre de l'arbre binaire de recherche A et placer le résultat dans un nouveau tableau B permettra d'obtenir un Monceau-min

[Vrai] [Faux]

**Question 12** [5 Points] quelle est la complexité du meilleur et du pire cas dans le retrait du nœud à la racine d'un arbre binaire de recherche. Donner ces complexités en notation Grand O en fonction du nombre de nœuds n dans l'arbre. Justifier votre réponse en dessinant des exemples d'arbre correspondant au pire cas et au meilleur cas.

**Question 13** [6 Points]

Quel sera l'arbre binaire résultat si la séquence de clés suivante est insérée 6,8,2,4,3,7

1. Si l'arbre est un Monceau-min?
2. Si l'arbre est un arbre binaire de recherche?

**Question 14** [4 Points]

1. Ajouter l'élément ayant la clé 27 à l'arbre binaire montre ci-dessous. Dessiner l'arbre résultant.
2. Retirer l'élément ayant la clé à l'arbre binaire montre ci-dessous. Dessiner l'arbre résultant