# Algorithmique - Informatique deuxième année

## Tas

**Exercice 1. Tas et représentation par un tableau.**

1. Dessiner les arbres binaires parfaits dont le parcours en largeur correspond à ces suites d'entiers :
   (a) 1, 5, 7, 6, 5, 8, 9, 12, 7, 5, 8, 9
   (b) 4, 5, 8, 6, 6, 9, 8, 7, 5, 11
2. Parmi les deux arbres représentés ci-dessus, quels sont ceux qui sont des arbres tournois ?
3. Soit $i$ un indice dans un tableau représentant un arbre parfait. Quel est l'indice de son nœud parent ?
4. Soit $i$ un indice dans un tableau qui représente un arbre parfait.
   Quels sont les indices de ses fils gauche et fils droit ?
5. Étant donné un tas vide, dessiner le tas et donner le tableau qui le représente après lui avoir ajouté dans l'ordre les valeurs suivantes : 5, 13, 4, 14, 1, 20.
6. Après lui avoir enlevé sa valeur minimale, dessiner le tas et le tableau qui le représente.

Pour représenter un tas, on utilise la structure suivante :

```
typedef struct
{
    int taille;
    int arbre[MAX];
} Tas;
```

où `MAX` désigne une constante.

**Exercice 2. Vérification.**
Écrire une fonction `int estTas(int tab[], int taille)` qui vérifie qu'un tableau `tab` de taille `taille` représente un tas.

**Exercice 3. Fils minimal.**
Écrire une fonction `int Fils(Tas T, int indice)` qui renvoie $-1$ si le nœud représenté par la case `indice` n'a pas de fils, sinon renvoie l'indice du nœud fils qui contient la plus petite étiquette.

**Exercice 4. Change.**
Écrire une fonction `void Change(Tas *T, int indice, int valeur)` qui remplace la valeur de la case `indice` par `valeur` et effectue différents échanges de case afin de respecter la structure de tas. On vérifiera que `indice` est inférieur à la taille du tas `T`.

**Exercice 5. Ajout.**
Écrire une fonction `int Ajout(Tas *T, int valeur)` qui ajoute une valeur au tas, tout en respectant sa structure. La fonction renverra 1 si l'ajout est correct et 0 sinon.

**Exercice 6. Extrait l'élément minimal.**
Écrire une fonction `int ExtraireMin(Tas *T)` qui extrait la valeur minimum d'un tas et la retourne, tout en respectant la structure d'un tas.

**Exercice 7. Complément.**
Reécrire les fonctions précédentes avec la structure suivante

```
typedef struct
{
    int taille;
    int *arbre;
    int Max;
} Tas;
```

En utilisant ces fonctions écrire la fonction `void TriTas(int T [],int taille)` qui effectue un tri en ordre décroissant du tableau `T`