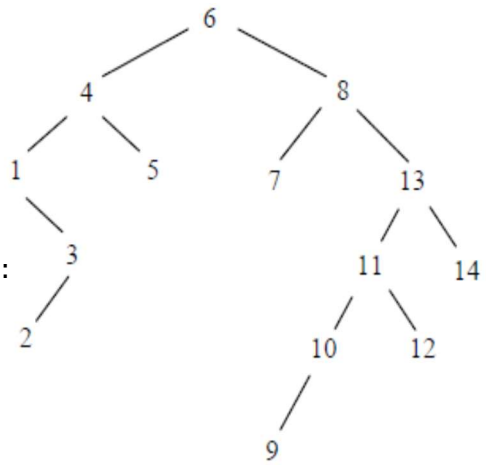
FICHE TD N°3
LES ARBRES



EXERCICE N°1 :

1. Donner les 3 parcours en profondeur de l'arbre binaire suivant :
2. Dessiner un arbre binaire avec dix nœuds étiquetés 0, 1, 2, …, 9 de telle façon que des parcours infixe et postfixe donnent respectivement les listes suivantes : 9, 3, 1, 0, 4, 2, 7, 6, 8, 5 (infixe) et 9, 1, 4, 0, 3, 6, 7, 5, 8, 2 (postfixe). Donner le parcours préfixe de l'arbre ainsi obtenu.

EXERCICE N°2 :

- Ecrire une fonction récursive permettant de retourner la hauteur d'un arbre binaire.
- Ecrire une procédure récursive qui inverse un arbre binaire (Chaque fils gauche est inversé avec son frère)

EXERCICE N°3 :

Etant donné un arbre n-aire représenté de manière chainée où chaque nœud possède un lien vers le fils le plus à gauche et un autre lien vers le nœud représentant le prochain « frère ».

1. Ecrire une routine permettant d'afficher tous les fils d'un nœud d'adresse donnée.
2. Ecrire une routine permettant de réaliser un parcours postfixe de l'arbre.

EXERCICE N°4 :

1. Soit la liste d'entiers $\boldsymbol{s}$ **= 11, 15, 5, 2, 3, 9, 17, 21, 22, 13, 19, 4, 12**. Construire l'arbre binaire de recherche associé à la liste (en prenant les éléments dans l'ordre où ils sont donnés). Que devient l'arbre après la suppression de 15 ?
2. Ecrire en C/C++ une procédure récursive afficheDecr qui affiche les valeurs d'un arbre binaire de recherche dans l'ordre décroissant.
3. Ecrire la fonction C/C++ qui prend en entrée la racine de l'arbre et supprime le plus grand élément.

EXERCICE N°5 :

1. Soit $T = [37,12,30,10,3,9,20,3,7,1,1,7,5]$ un max-tas. Donner son contenu :
   - Après insertion de l'élément 31.
   - Après suppression de l'élément 37.

2. Etant donné un tas, écrire en C/C++ une procédure itérative permettant d'afficher les ancêtres d'un élément d'indice i.