TP 8 SDA

implémentation des arbres

La structure d'un arbre est naturellement récursive. De ce fait, le recours à la récursivité est habituelle dans la plupart des algorithmes sur ce type de données.

Ainsi en est-il de la taille ou de la hauteur d'un arbre qui s'écrivent en pseudo-code de la façon suivante :

```
taille(arbre a)
      si estVide(a) alors ret = 0
      sinon ret = taille(filsGauche(a)) + taille(filsDroit(a)) + 1
      retourne ret

hauteur(arbre a)
      si estVide(a) alors ret = 0
      sinon hg = hauteur(filsGauche(a))
       hd = hauteur(filsDroit(a))
       ret = max(hg, hd) +1
      retourne ret
```

1. Implémenter les arbres binaires étiquetés (par T ici int) *dynamique*

```
Arbre* nouvArbre(); // rend l'arbre vide, ici (arbre)NULL
Arbre* enrac(T racine, Arbre* fg, Arbre* fd);
Arbre* creerFeuille(T val);
T racine(Arbre* a);
Arbre* fg(Arbre* a); // rend le fils gauche
Arbre* fd(Arbre* a); // rend le fils droit
int taille(Arbre* a);
int hauteur(Arbre* a);
Bool estVide(Arbre* a);
Bool estFeuille(Arbre* a);
```

void print_dot(Arbre* a); // voir ci-après (ou mieux void sauve_dot(Arbre* a, char* filename);)

```
void libere_memoire(Arbre* a);
```

Affichage des graphes:
Exemple de fichier .dot:
digraph G
{
1 -> 2
1 -> 3
2 -> 4
2 -> 5
3 -> 6
}

installer le paquet graphviz
Commande: *dot* -Tpdf mon_arbre.dot -o mon_arbre.pdf

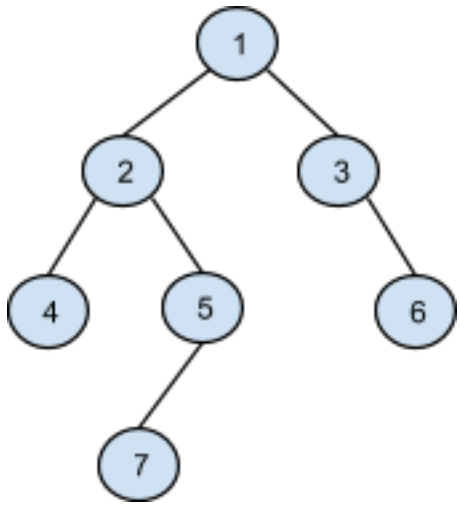
2. Parcours
Afficher les labels de l'arbre en suivant les trois types de parcours possible.

Le parcours **préfixe** :

```
parcoursPrefixe(arbre a) :
  si a non vide alors
     affiche(racine(a))
     parcoursPrefixe(fg(a))
    parcoursPrefixe(fd(a))



Résultat: 1 2 4 5 7 3 6
```

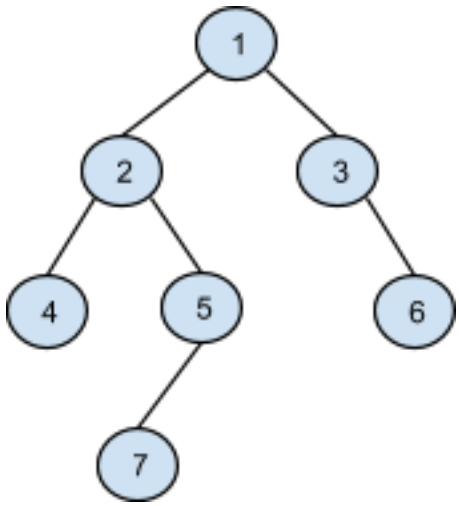


Le parcours **infixe**

```
parcoursInfixe(arbre a) :
  si a non vide alors
     parcoursInfixe(fg(a))
     affiche(racine(a))
     parcoursInfixe(fd(a))




Résultat    : 4 2 7 5 1 3 6
```



Le parcours **postfixe**

```
parcoursPostfixe(arbre a) :

 si a non vide alors
    parcoursPostfixe(fg(a))
     parcoursPostfixe(fd(a))
     affiche(racine(a))

Résultat    : 4 7 5 2 6 3 1
```