Correction TD4 - Les arbres

Exercice 1

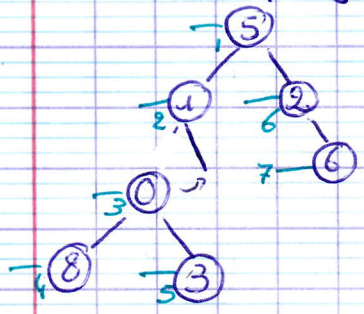
```
def creee_arbre (racine, etiquette):
    return [racine, {racine: []}, {racine: etiquette}]

def ajoutee_fils (A, pere, fils, etiquette_fils):
    adj = A[1]
    etiq = A[2]
    if not pere in adj:
        raise ValueError ("Le père n'existe pas")
    if fils in adj:
        raise ValueError ("Le fils existe déjà dans l'arbre")
    adj[fils] = []
    adj[pere].append(fils)
    etiq[fils] = etiquette_fils

def fils (A, p):
    return A[1][p]

def pere (A, p):
    adj = A[1]
    for pere in adj:
        for s in adj[pere]:
            if s == p:
                return pere
    return None

def racine (A):
    return A[0]
```

```
def etiquette (A,s):
    return A[2][s]

def taille_arbre (A):
    nb_s = taille_rec (A, racine(A))
    Return nb_s

def taille_rec (A, p):
    nb_s = 1
    for f in fils (A, p):
        nb_s += taille_rec (A, f)
    Return nb_s

def taille_sous_arbre (A, p):
    return taille_rec (A, p)

def parcours_arbre (A, p):        p=[]
    parcours_recursif (A, p, racine (A))
    return p
def parcours_recursif (A, p, s)
    p.append (s)
    for f in fils (A, s)
        parcours_recursif (A, p, f)

def parcours_niveau (A, h, p):
    parcours_niveau_recursif ( A, h, A[0], p)
    return p

def parcours_niveau_recursif (A, h, s, p):
    if h==0:
        p.append (s)
    else:
        for f in fils (A, s):
            parcours_niveau_recursif (A, h-1, f, p)
```

t = A[2]
t[f] = étiquette
return A

Arbres binaires

Parcours préfixe

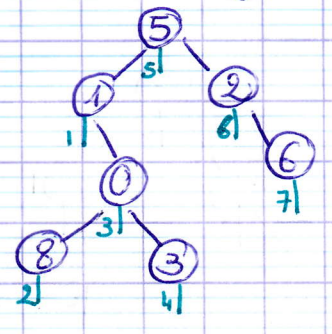


Résultat : [5, 1, 0, 8, 3, 2, 6]

On fait le tour de l'arbre et dès que l'on rencontre un sommet on le met dans la file.

```
def prefixe_rec(A, s, l)
    l.append(s)
    fg = fils_gauche(A, s)
    if not fg is None:
        prefixe_rec(A, fg, l)
    fd = fils_droit(A, s)
    if not fd is None:
        prefixe_rec(A, fd, l)
```

Parcours infixe

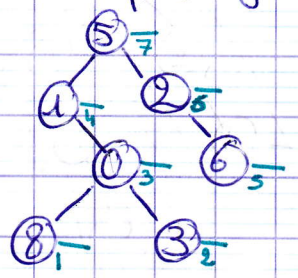


[1, 8, 0, 3, 5, 2, 6]

On fait le tour de l'arbre et quand on rencontre un sommet qui n'a pas encore été inséré dans la liste on l'insère uniquement si son fils gauche (s'il existe) a déjà été inséré.

```
def infixe_rec(A, s, l)
    fg = fils_gauche(A, s)
    if not fg is None
        infixe_rec(A, fg, l)
    l.append(s)
    fd = fils_droit(A, s)
    if not fd is None:
        infixe_rec(A, fd, l)
```

Parcours post-fixe



On fait le tour de l'arbre et quand on rencontre un sommet on l'insère dans la liste uniquement si tous les fils ont déjà été ajoutés.

[8, 3, 0, 1, 6, 2, 5]

```
def post_fixe_rec(A,s,l)
    fg = fils_gauche(A,s)
    if not fg is None:
        post_fixe_rec(A,fg,l)
    fd = fils_droit(A,s)
    if not fd is None:
        post_fixe_rec(A,fd,l)
    l.append(s)
```