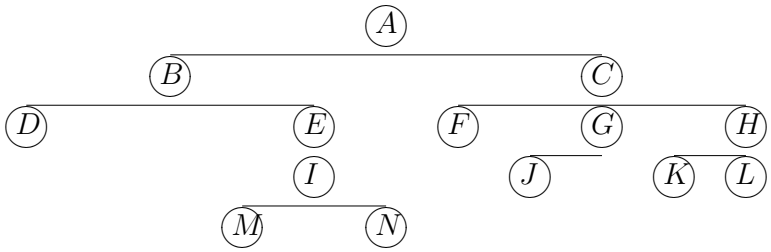
3.5 Exercices

Exercice 1 (- pts)
Lister les nœuds de l'arbre ci-dessous suivant les parcours préfixé, postfixé et infixé.



Exercice 2 (- pts)
Soit A un arbre binaire représenté par pointeurs (utiliser les types NŒUD et ARBRE définis en cours).

a. Écrire des programmes récursifs permettant les parcours préfixé, postfixé et infixé.
b. Écrire la version itérative du parcours préfixé.

Exercice 3 (- pts)

1. Écrire une fonction qui teste si deux arbres binaires sont égaux ou non.

```
procedure  equal_tree(A, B : TREE)
debut
si (A=NIL) alors  si (B=NIL) alors  retourner (VRAI)
                             sinon  retourner (FAUX)
                   finsi
           sinon  si (B=NIL) alors  retourner (FAUX)
                             sinon si (A↑val≠B↑val)alors
                                                   retourner  (FAUX)
                                                   sinon
                                                   retourner (equal_tree(A↑fg,B↑fg)ET
                                                             equal_tree(A↑fd,B↑fd))
                                         finsi
                   finsi
finsi
fin
```

2. Écrire une procédure qui transforme un arbre binaire en son symétrique par rapport à la racine.

```
procedure  rotatel_tree(A : TREE)
debut
si (A≠NIL) alors  swap(A↑fg, A↑fd)
                  // la fct swap permute les valeurs des deux arguments
                  rotate_tree(A↑fg)
                  rotate_tree(A↑fd)
finsi
fin
```

3. Écrire une procédure qui construit une copie d'un arbre binaire.

```
procedure  copy_tree(A : TREE, var  copy : TREE)
debut
si (A = NIL) alors  copy ←NIL
             sinon  Allouer (copy)
                    copy↑val ←A↑val
                    copy_tree(A↑fg, copy↑fg)
                    copy_tree(A↑fd, copy↑fd)
             finsi
fin
```

4. Soit un arbre binaire d'entiers. Écrire une fonction qui calcule la somme des valeurs dans un tel arbre.

```
fonction  sum_tree(A : TREE):entier
debut
si (A = NIL) alors  retourner  0
             sinon  retourner  (A↑val+ sum_tree(A↑fg)+ sum_tree(A↑fd))
finsi
fin
```

Exercice 4 (- pts)
Soient A un arbre quelconque non vide et n un nœud de A. On définie le degré de n dans A comme étant le nombre de ses fils. Notons par Nn_2 le nombre de nœuds de degré 2 et par Nf le nombre de feuilles dans A. En supposant que A est un arbre binaire, on vous demande de :

1. Déterminer le nombre maximum de nœuds de l'arbre A s'il est de hauteur h.

Le nombre de nœuds d'un arbre binaire est maximale quand il est complet (tous les nœuds ont 2 fils) et toutes les feuils sont de profondeur h.

$$N_{tot} = 1 + 2 + 4 + 8 + \cdots + 2^{h-1} = \sum_{i=0}^{h-1} 2^i = \frac{2^h - 1}{2 - 1} = 2^h - 1$$

2. Montrer, en vous inspirant de la question précédente, que $Nf = Nn_2 + 1$.

$N_{tot} = 2^h - 1$
Nf= le dernier terme de la suite = 2^{h-1}
Nn_2 sont les nœuds qui admettent 2 fils c-à-d les nœuds internes
$Nn_2 = N_{tot} - Nf = 2^h - 1 - 2^{h-1} = 2^{h-1} - 1 = Nf - 1$

3. Écrire une fonction pour calculer la hauteur h de l'arbre A.

```
fonction  h(A : TREE):entier
debut
si (A = NIL) alors  retourner  0
             sinon  retourner  (1+ max(h(A↑fg),h(A↑fd)))
              //La fct max retourne la valeur la plus grande des 2 arguments
finsi
fin
```