

# Examen d'algorithmique 1

14h-15h30

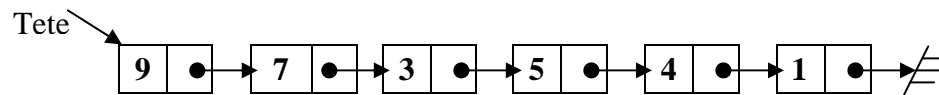
A2, A3

## Exercice 1 (10 pts: 4 + 3 + 3)

On souhaite représenter les nombres naturels par des listes linéaires chaînées où chaque chiffre d'un nombre est rangé dans un maillon de la liste :

Exemple :

La nombre "145379" est représentée par la liste suivante :



On suppose que les fonctions suivantes sont disponibles et peuvent être utilisées :

- Fonction **Nbre**(c :car) :entier retourne la valeur entière du caractère numérique c : Nbre('1')=1
- Fonction **Car**(N :entier) : car : retourne le caractère du naturel  $N < 10$  : Car(1) = '1'
- Fonction **Comparer**(N1, N2 : pointeur(TMaillon)) : entier  
Compare les deux entiers représentés par les deux listes N1 et N2 et retourne : -1 si  $N1 < N2$ , 0 si  $N1 = N2$  et 1 si  $N1 > N2$ .
- Fonction **Somme**(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon)  
qui retourne une liste contenant la somme des deux naturels N1 et N2.

Il est demandé d'écrire les fonctions suivantes :

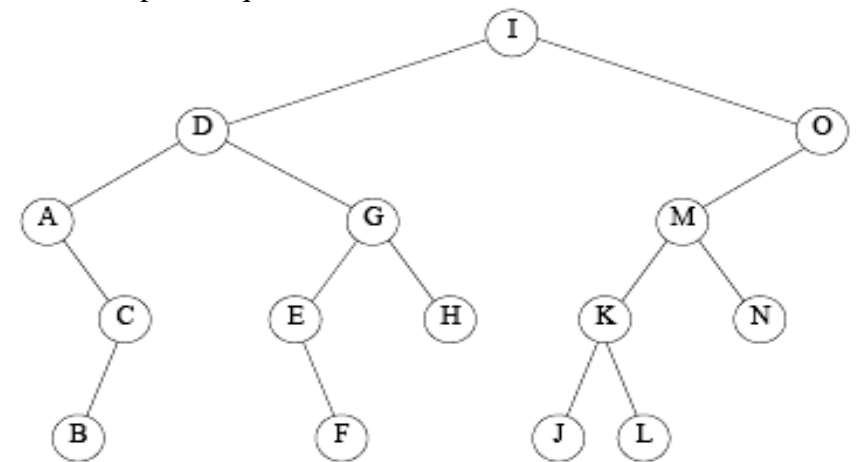
1. Fonction **Moins**(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon) ;  
Retourne NIL si  $N1 \leq N2$  et la liste représentant  $N1 - N2$  sinon.
2. Fonction **Mult**(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon) ;  
Retourne une liste représentant  $N1 * N2$ .
3. Fonction **Div**(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon) ;  
Retourne NIL si  $N1 < N2$  et le résultat de la division sinon.

## Exercice 2 (2.5 pts)

Ecrire la procédure **TrierTas**(T :Tableau[1..N]), permettant de trier le tableau T en utilisant un tas (utiliser sans implémentation les procédures InsérerTas et RetirerTas)

## Exercice 3 (7.5 pts : 1.5 + 1 + 2.5 + 2.5)

Soit l'arbre binaire de recherche suivant composé de caractères et basé sur l'ordre alphabétique :

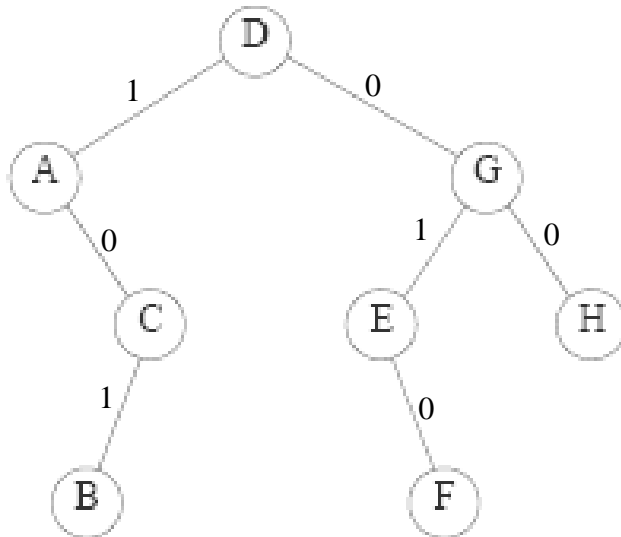


Tournez la page ../.

1. Donner les chaînes de caractères obtenus en parcourant l'arbre en profondeur en l'ordre préfixe, infixe puis postfixe.

2. Donner l'arbre après la suppression de 'D' puis de 'I'.

On associe à chaque caractère un code binaire représentant sa position dans l'arbre binaire comme suit :



A chaque fois qu'on passe à un fils gauche, on considère un 1 et chaque fois qu'on passe à un fils droit, on considère un 0.

Il est demandé d'écrire les deux fonctions suivantes :

3. Fonction **Code**(Racine: Pointeur(TNœud), C: caractère): chaîne ;

qui retourne le code binaire du caractère C dans une chaîne de caractères contenant des '0' et des '1'.

Exemple : Code('C') = '10', Code('F') = '010'

4. Fonction **Caract**(Racine:Pointeur(TNœud), B:chaîne):caractère;

qui retourne le caractère correspondant au code binaire donné dans la chaîne B.

Exemple : Caract('00') = 'H' ; Caract('101') = 'B'

**N.B :** On utilise la relation d'ordre entre les caractères alphabétiques, si C1 et C2 sont deux caractères, C1 > C2 si C2 se trouve après C1 dans l'ordre alphabétique.

***Bon courage***

A. Djefal

## Corrigé type

### Exercice 1

#### 1. 4 Pts

```

Fonction Moins(N1,N2: pointeur(TMaillon)): pointeur(TMaillon);
Var P, tete,Q : pointeur(TMaillon) ;
    V1, V2, Reste : entier ;
Debut
Si Comparer(N1,N2)<1 alors Moins ← NIL ;
Sinon
    Reste ← 0 ;
    Tete ← NIL ;
    TQ N1≠NIL Faire
        Allouer(P) ; Aff_Adr(P,NIL) ;
        Si Tete=NIL alors Tete ←P
            Sinon Aff_Adre(Q,P) ;
        FSi
        V1← Nbre(Valeur(N1)) ;
        Si N2≠NIL alors V2 ← Nbre(Valeur(N2))
            Sinon V2←0 ;
        FSi
        Si V1≥(V2+ Reste) alors
            Aff_Val(P, Car(V1- V2 – Reste))) ;
            Reste ← 0 ;
        Sinon
            Aff_Val(P, Car(V1 + 10 - V2 - Reste))) ;
            Reste ← 1 ;
        Fsi
        Q ← P ;
        N1 ← Suivant(N1) ;
        Si N2≠NIL alors N2 ← Suivant (N2) ;
    FTQ ;
    Mois ← tete ;
FSi
Fin ;

```

#### 2. 3 Pts

```

Fonction Mult(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon) ;
Var Un, Resultat : pointeur(TMaillon) ;
Debut
Si N1 = NIL ou N2 = NIL alors Mult ← NIL ;
Sinon
    Allouer(Un) ; Aff_Adr(Un,Nil) ;
    Aff_Val(Un,'1') ;
    Resultat ← NIL ;
    TQ Comparer(N2,NIL) =1 Faire
        Resultat ← Somme (Resultats, N1) ;
        N2 ← Moins(N2, Un) ;
    FTQ ;
    Mult ← Resultat ;
FSi
Fin ;

```

#### 3. 3 Pts

```

Fonction Div(N1, N2 : pointeur(TMaillon)) : pointeur(TMaillon) ;
Var Un, Resultat, Reste : pointeur(TMaillon) ;
Debut
Si Comparer(N1,N2)=-1 ou N1 = NIL ou N2 = NIL alors Div ← Nil ;
Sinon
    Resultat ← Nil ;
    Reste ← N1 ;
    Allouer(Un) ; Aff_Adr(Un,Nil) ;
    Aff_Val(Un,'1') ;
    TQ Comparer(Reste,N2) ≥ 0 Faire
        Resultat ← Somme(Resultat, Un) ;
        Reste ← Moins (Reste, N2) ;
    FTQ
    Div ← Resultat ;
FSi
Fin ;

```

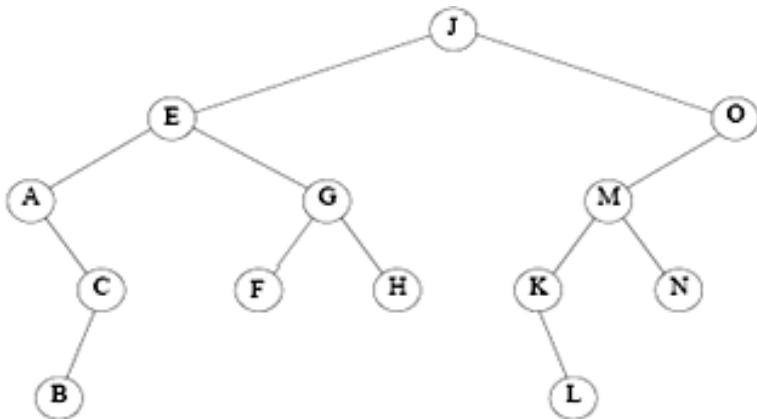
### Exercice 2 (2.5 pts)

```
Procédure TrierTas(T :Tableau[1..N]) ;  
Var i : entier ;  
Debut  
    Init_Tas ;  
  
    Pour i =1 à N faire  
        Insérer_Tas(T[i]) ;  
    FPour ;  
  
    Pour i = 1 à N faire  
        Retirer_Tas(T[i]) ;  
    FPour ;  
  
Fin ;
```

### Exercice 3 (7.5 pts : + 1 + 2.5 + 2.5)

1. **1.5 Pt**
- Parcours en profondeur préfixe : IDACBGEFHOMKJLN
  - Parcours en profondeur infixe : ABCDEFGHIJKLMNO
  - Parcours en profondeur postfixe : BCAFEHGDJLKNMOI

2. **1 Pt**



### 3. **2.5 Pts**

```
Fonction Code(Racine: Pointeur(TNoeud),C: caractère): chaîne ;  
Var N : pointeur(TNoeud) ;  
    S : Chaîne ;  
Debut  
    N ← Racine ; S ← "" ;  
    TQ N≠ NIL et Valeur(N) ≠ C Faire ;  
        Si Valeur(N)>C et FG(N) ≠NIL alors  
            S ← S + '1' ; N ← FG(N) ;  
        FSi  
        Si Valeur(N)<C et FD(N) ≠NIL alors  
            S ← S + '0' ; N ← FD(N) ;  
        FSi  
    FTQ  
    Si N =NIL alors Code ← '#' // Erreur : caractère inexistant  
        Sinon Code ← S ;  
    FSi  
Fin ;
```

### 4. **2.5 Pts**

```
Fonction Caract(Racine:Pointeur(TNoeud), B:chaîne):caractère;  
Var N : pointeur(TNoeud) ;  
    I : entier ;  
Debut  
    N ← Racine ; I ← 1;  
    TQ N≠ NIL et I <= Longueur(B) Faire ;  
        Si B[I]='1' alors N ← FG(N) ;  
        Sinon N ← FD(N) ;  
    FSi  
    I←I+1;  
    FTQ  
    Si N =NIL alors Caract ← '#' // Erreur : code inexistant  
        Sinon Caract ← Valeur(N) ;  
    FSi  
Fin ;
```