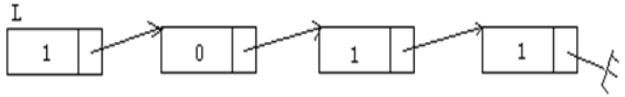
2LMD **07 Fev 2007**

Examen d'algorithmique 1

10h-11h30 **S10, S14**

Exercice 1 (5.5 Pts)

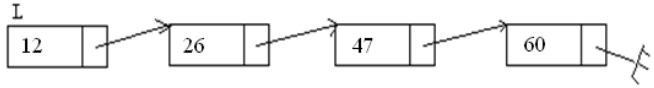
On convient de représenter un nombre binaire b1b2..bn, où chaque bi est un 0 ou un 1, par une liste linéaire chaînée où chaque élément contient un bi.

Exemple : 1011 → 

1. Donner la structure des maillions utilisées (déclaration)
2. Ecrire la procédure Mult2(L) qui multiplie la valeur de la liste par 2 :
   Exemple : Mult2(1011=11) → 10110=22
3. Ecrire la procédure Div2(L) qui divise la valeur de la liste par 2 :
   Exemple : Div2(1011=11) → 101=5
4. Ecrire la procédure Div2k(L, k) qui divise la valeur de la liste L par 2^k en utilisant la procédure Div2.
   Exemple : Div2k(1011,2) → 10 = 2

Exercice 2 (5.5 Pts)

On souhaite utiliser les listes linéaires chaînées unidirectionnelles **triées** pour représenter des ensembles d'entiers. L'exemple de la liste du schéma ci-dessous représente l'ensemble {12, 26, 47, 60}



On rappelle qu'un élément n'apparaît au plus qu'une seule fois dans un ensemble.

1. Donner la structure des maillions utilisées (déclaration)
2. Ecrire la fonction **Cardinal(L)** qui retourne le nombre d'éléments d'un ensemble donnée (sous forme de liste)
3. Ecrire la procédure **Intersection(L1, L2, L)** qui retourne dans L les éléments qui apparaissent à la fois dans L1 et L2.
4. Ecrire la fonction **Inclus(L1, L2)** qui retourne vrai si tous les éléments de L1 existent dans L2 et faux sinon.

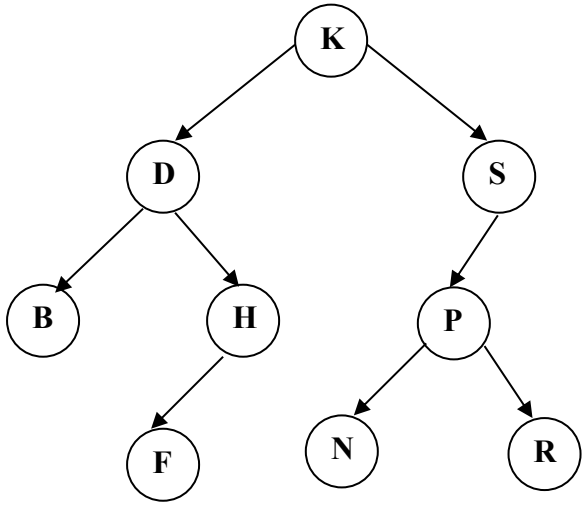
Exercice 3 (5 Pts)

1. Ecrire la procédure **Inverser File(F)** qui permet d'inverser une file d'attente d'entiers donnée en utilisant une pile.
   Exemple File=(5, 2, 3, 0) → (0, 3, 2, 5) ;
2. **Calculer** la complexité de cette procédure (IverserFile).
3. Ecrire la procédure **Inverser Pile(P)** qui permet d'inverser une pile d'entiers donnée en utilisant une file d'attente.
   Exemple Pile=(5, 2, 3, 0) → (0, 3, 2, 5) ;

Tournez la page ../..

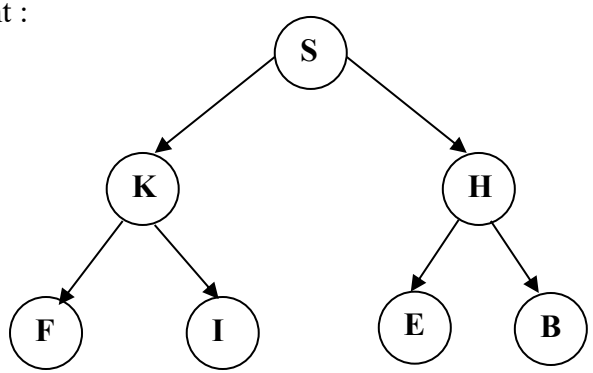
Exercice 4 (4 Pts)

Soit l'arbre binaire de recherche suivant :



1. Donner les résultats des parcours Préfixé, Poste Fixé de cet arbre
2. Donner l'arbre après la suppression du nœud « D ».

Soit le tas suivant :



3. Donner le tas après un retrait.
4. Donner le tas après l'ajout de « P ».

Bon courage

Correction

Exercice 1

1. Type TBit = Structure
   Val :Entier ;
   Suivant : Pointeur (TBit) ;
   Fin ; **(0.5 Pt)**

2. La multiplication d'un nombre binaire par 2 consiste à ajouter un '0' à sa droite s'il n'est pas nul.
   Exemple : 111 (7) * 2 = 111**0** (14)

```
Procedure Mult2 (L : Pointeur(TBit)) ;
  Var   P , PP, Q : Pointeur(TBit)
  Debut
        P ← L ; PP ← Nil ;
        TQ P <> Nil Faire
            PP ← P ;
            P ← Suivant(P) ;          /* aller à la fin de la liste */
        FTQ ;
        Si L <> Nil alors
                Allouer (Q) ;
                Aff_Val (Q, 0) ;      /* Ajouter un '0' */
                Aff_Adr (Q, Nil);
                Aff_Adr (PP,Q);
        Fsi
  Fin;
```
(2 Pts)

3. La division d'un nombre binaire par 2 consiste à supprimer sont dernier bit à droite (le décaler à droite)
   Exemple 1111 (15) / 2 = 111 (7)

```
Procedure Div2(L: Pointeur(TBit)) ;
      Var   P , PP, PPP: Pointeur(TBit) ;
      Debut
        Si L <> Nil alors
            P ← L ; PP ← Nil ; PPP ← Nil ;
            TQ P <> Nil Faire
                PPP ← PP ; PP ← P ;
                P ← Suivant(P) ;          /* aller à la fin de la liste */
            FTQ ;
            Si PPP = Nil alors            /* Un seul élément */
                    L ← Nil ;
            Sinon
                    Aff_Adr(PPP, Nil);
            Fsi
            Libérer (PP) ;
        FSi
      Fin;
```
(2 Pts)

4.
```
Procedure Div2K(L : Pointeur(TBit) ; k : entier) ;
    Var  i :entier ;
    Debut
      Pour i = 1 à k faire
            Div2(L) ;
      FPour
    Fin ;
```
(1 Pt)

Exercice 2

1. Type TMaillon = Structure
   Val : entier
   Suivant : Pointeur (TMaillon) ;
   Fin ;

2. Fonction Cardinal (L : Pointeur (TMaillon) : Entier ;

```
Var P : Pointeur (TMaillon) ;
     N : entier ;
Debut
    P ← L ; N ← 0 ;
    TQ P <> Nil Faire
      N ← N + 1 ;
      P ← Suivant (P) ;
    FTQ ;
  N ← N+1 ;
Fin ;
```

(1 Pt)

3. Procedure Intersection (L1, L2, L : Pointeur (TMaillon) ;

```
Var P1, P2, P, Q : Pointeur (TMaillon)
Debut
    P1 ← L1 ; P2 ← L2 ;
    L ← Nil ;
    TQ P1 <> Nil et P2 <> Nil faire
        Si Valeur(P1) = Valeur (P2) alors
            Allouer(P) ;
            Aff_Val(P, Valeur(P1) ;
            Aff_Adr(P, Nil) ;
            Si L = Nil alors L ← P
                       Sinon Aff_Adr (Q, P) ;
            FSi ;
            Q ← P ;
            P1 ← Suivant (P1) ; P2 ← Suivant (P2) ;
        Sinon
            Si Valeur(P1) < Valeur(P2)   alors  P1 ← Suivant (P1)
                                                Sinon P2 ← Suivant (P2)
            FSi;
        FSi
    FTQ
Fin ;
```

(2.5 Pts)

4. Fonction Inclus (L1, L2 : Pointeur (TMaillon)) : Booleen ;

```
Var P1, P2 : Pointeur (TMaillon) ;
    Inc : Booleen ;
Debut
    P1 ← L1 ; P2 ← L2 ;
    Inc ← Vrai ;
    TQ ( P1 <> Nil) et (P2 <> Ni)l et Inc faire
        Si Valeur (P1) = Valeur (P2) alors
                P1 ← Suivant (P1) ;
                P2 ← Suivant (P2) ;
        Sinon
            Si Valeur (P1) > Valeur (P2) alors P2 ← Suivant (P2)
                                         Sinon  Inc ← Faux
            FSi
        FSi
    FTQ
    Inclus ← Inc et (P1 = Nil) ;
Fin ;
```

(1.5 Pt)

Exercice 3

1. Procedure Inverser_File (F)

```
 Var X : entier ;
      P : Pile ;
Debut
    Init_Pile (P) ;                              O(1)

    TQ Non File_Vide (F) Faire  n fois     O(n)        O(n)
       Défiler (F, X) ;          O(1)
       Empiler (P,X) ;           O(1)
    FTQ ;

    TQ Non Pile_Vide (P) Faire  n fois     O(n)
       Dépiler (P, X) ;          O(1)
       Enfiler (F,X) ;           O(1)
    FTQ ;
Fin ;
```

(2 Pts)

2. La complexité de cette Procedure est O(n) **(1 Pt)**

3. Procedure Inverser_Pile (P)

```
 Var X : entier ;
       F : File ;
Debut
    Init_File (F) ;

    TQ Non Pile_Vide (P) Faire  n fois
       Dépiler (P, X) ;
       Enfiler (F,X) ;
    FTQ ;

    TQ Non File_Vide (F) Faire  n fois
       Défiler (F, X) ;
       Empiler (P,X) ;
    FTQ ;
```

(2 Pts)

Exercice 4

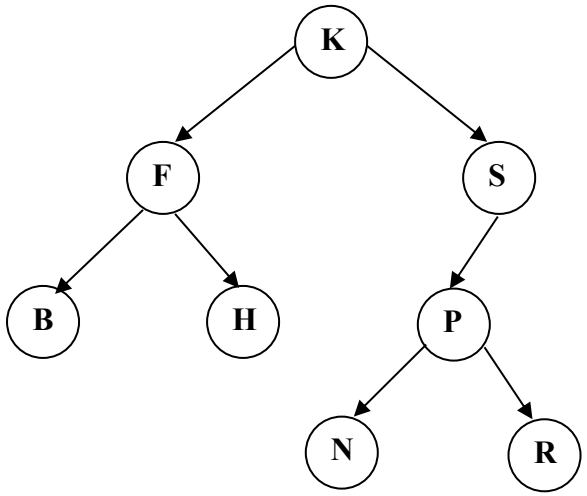
1. Parcours Préfixé P, FG, FD

   K D B H F S P N R **(0.5 Pt)**

   Parcours PostFixé FG, FD, P

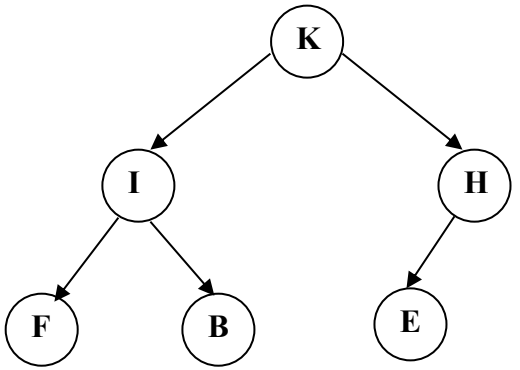
   B F H D N R P S K **(0.5 Pt)**

2.



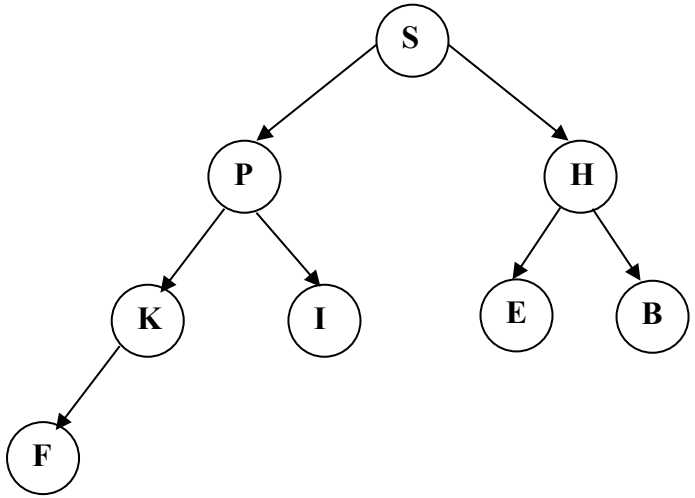
(1 Pt)

3.



(1 Pt)

4.



(1 Pt)