Université Mohamed Khider-Biskra
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique

$2^{ème}$ année LMD — ALGO1
13 Mars 2012 — 8:00-9:30, Amphi 1

Examen de rattrapage

Exercice 1 Tas (10 pts : 1.5 + 1.5 + 2.5 + 2.5 + 2)

Soit les valeurs des clés suivantes :

38, 5, 2, 17, 10, 3, 16, 22, 11, 33

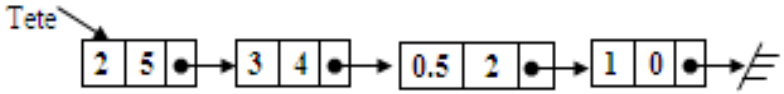
On vous demande de :

1. Donner le tas dynamique obtenu après l'insertion de ces clés dans l'ordre.
2. Donner le tas statique correspondant.
3. Donner la déclaration des structures de données permettant de représenter en mémoire le tas statique précédent.
4. Ecrire la procédure **Enfiler(x)** permettant d'ajouter au tas statique la valeur x.
5. Ecrire la fonction **Défiler** permettant de défiler une clé du tas statique.
6. Donner le tas statique de la question 2, après les appels suivants : Enfiler(4) ; Enfiler(14) ; Défiler ; Enfiler(1) ; Défiler ;

Exercice 2 LLCs (10 pts : 1 + 1.5 + 2.5 + 2.5 + 2.5)

On souhaite représenter les polynômes par des listes linéaires chaînées, où chaque terme d'un polynôme est rangé dans un maillon de la liste contenant le degré du terme et le coefficient correspondant.

Exemple : Le polynôme $2x^5+3x^4+\frac{1}{2}x^2+1$ est représentée par la liste suivante :



On vous demande de :

1. Donner les structures de données nécessaires à la représentation des polynômes.
2. Ecrire la fonction **Valeur(P, x)** permettant de donner la valeur du polynôme P pour x.
   Exemple : Valeur(tete, 1) = 6.5
3. Ecrire la procédure **Dériver(P)** permettant de calculer la dérivée du polynôme P et la retourner dans P.
4. Ecrire la procédure **Intégrer(P)** permettant de calculer l'intégrale du polynôme P et le retourner dans P.
5. Ecrire la procédure **Déviation(P, x)** permettant de retourner *Vrai* si x est un point de déviation du polynôme P et *Faux* sinon.

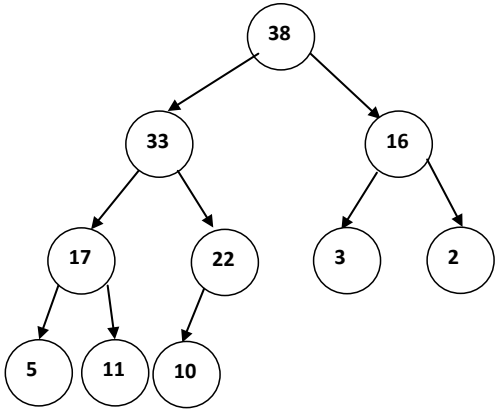
★★★ Bonne chance ★★★

A.Djeffal

Corrigé type

Exercice 1 : Tas

1. Tas dynamique (1.5 pt)



2. Tas statique (1.5 pt)

1	2	3	4	5	6	7	8	9	10
38	33	16	17	22	3	2	5	11	10

3. Structures de données (1 pt)

```
Var  Tas : Tableau[1..NMax] de entier
     N : entier ; // nombre d'éléments dans le tas, initialisé à 0
```

4. Procédure Enfiler(x) (2.5 pts)

```
Procédure Enfiler( x : entier ;);
Var  P,Pere : entier ;
Début
    Si (N=NMax) Alors
        Ecrire('le tas est plein')
    Sinon
        N ← N+1 ;
        Tas[N] ← x ;
        P ← N ;
        Pere ← N div 2 ;
        Tant que ((P > 1) et Tas[Pere] < Tas[P])) faire
            x ← Tas[Pere] ;
            Tas[Pere] ← Tas[P] ;
            Tas[P] ← x ;
            P ← Pere ;
            Pere ← P div 2 ;
        Fin TQ;
    Fin Si;
Fin;
```

5. Fonction Défiler (2.5 pts)

```
Fonction Defiler() : entier;
Var  P,Pere : entier ;
Début
    Si (N=0) Alors
        Ecrire('le tas est vide')
    Sinon
        Defiler ← Tas[1] ;
        Tas[1] ← Tas[N] ;
        N ← N-1 ;
        P ← 1 ;
        Fils1 ← 2 ;
        Fils2 ← 3 ;
        Tant que (((Fils1 ≤ N) et Tas[P] < Tas[Fils1]))
        ou ((Fils2 ≤ N) et Tas[P] < Tas[Fils2])) ) faire
            Si (Fils2 > N) Alors
                PMax ← Fils1
            Sinon
                Si (Tas[Fils1] > Tas[Fils2]) Alors
                    PMax ← Fils1
                Sinon
                    PMax ← Fils2
                Fin Si;
            Fin Si;

            x ← Tas[P] ;
            Tas[P] ← Tas[PMax] ;
            Tas[PMax] ← x ;
            P ← PMax ;
            Fils1 ← P * 2 ;
            Fils2 ← P * 2 + 1 ;
        Fin TQ;
    Fin Si;
Fin;
```

6. Tas statique (1 pt)

1	2	3	4	5	6	7	8	9	10	11
22	17	16	11	10	14	2	5	1	3	4

Exercice 2 : LLCs

1. Structures de données (1 pt)

```
Type  TTerme = Structure
        Coef : réel ;
        Degre : entier ;
        Suivant : Pointeur(TTerme) ;
Fin ;
```

2. Fonction Valeur (1.5 pt)

```
Fonction Valeur( P : Pointeur(TTerme), x : réel) : réel;
Var  P1 : Pointeur(TTerme) ;
       S,C : réel ;
Début
    S ← 0 ;
    P1 ← P ;
    Tant que (P1 ≠ Nil) faire
        C ← 1 ;
        Pour i de 1 à Degre(P1) faire
            C ← C*x ;
        Fin Pour;
        S ← S + C * Coef(P1) ;
        P1 ← Suivant(P1) ;

    Fin TQ;
    Valeur ← S ;

Fin;
```

3. Procédure Dériver (2.5 pt)

```
Procédure Dériver( P : Pointeur(TTerme));
Var  P1,PP1 : Pointeur(TTerme) ;
Début
    PP1 ← Nil ;
    P1 ← P ;
    Tant que (P1 ≠ Nil) faire
        Si (Degré(P1) ≠ 0) Alors
            Aff_ Coef(P1, Coef(P1)*Degré(P1)) ;
            Aff_ Degré(P1,Degré(P1)-1) ;
            PP1 ← P1 ;
            P1 ← Suivant(P1) ;
        Sinon
            Si (PP1 = Nil) Alors
                P ← Nil ;
            Sinon
                Aff_ Adr(PP1,Nil)
            Fin Si;
            Libérer(P1) ;
            P1 ← Nil ;

        Fin Si;
    Fin TQ;
Fin;
```

4. Procédure Intégrer (2.5 pt)

```
Procédure Intégrer( P : Pointeur(TTerme));
Var  P1,PP1 : Pointeur(TTerme) ;
Début
    PP1 ← Nil ;
    P1 ← P ;
    Tant que (P1 ≠ Nil) faire
        Aff_Coef(P1, Coef(P1)/(Degré(P1)+1)) ;
        Aff_Degré(P1,Degré(P1)+1) ;
        PP1 ← P1 ;
        P1 ← Suivant(P1) ;
    Fin TQ;
    Allouer(P1) ;
    Aff_Coef(P1,1) ;
    Aff_Degré(P1,0) ;
    Aff_Adr(P1,Nil) ;
    Si (PP1 = Nil) Alors
        P ← P1 ;
    Sinon
        Aff_Adr(PP1,P1) ;
    Fin Si;
Fin;
```

5. Fonction Déviation (2.5 pt)

```
Fonction Déviation( P : Pointeur(TTerme),  x : réel) :
booléen;
Début
    Si (P=Nil ou Degré(P)< 2) Alors
        Déviation ← Faux
    Sinon
        Dériver(P) ;
        Dériver(P) ;
        Déviation ← (Valeur(P,x) = 0) ;
    Fin Si;
Fin;
```