Université A. MIRA de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

Mai 2022
Durée : 1h30
Niveau : L1 RN

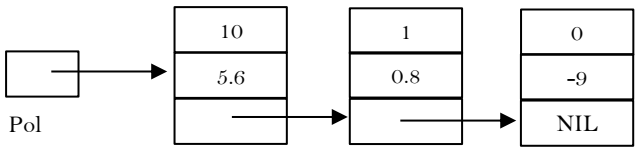
EMD 2 ALGORITHMIQUE

Exercice 1 (7,5 points) (interrogation 2)

1. Proposer une fonction **récursive** qui calcule la somme de deux entiers positifs **a** et **b**, en supposant que les seules opérations de base possibles sont :
   - L'ajout de 1 à un entier a : a + 1.
   - Le retrait de 1 à un entier a : a − 1.
   - Les comparaisons à 0 d'un entier a : a = 0, a > 0, et a < 0.
2. Étendre la fonction aux entiers de signe quelconque.

Exercice 2 (12,5 points)

On s'intéresse à la manipulation de polynômes creux à l'aide de listes linéaires chainées. Un polynôme creux est un polynôme contenant très peu de monômes non nuls. Par exemple : $\mathbf{P(x) = 5.6\ x^{10} + 0.8\ x - 9}$ contient **11** termes, dont **3** seulement sont non nuls. La liste **Pol** correspondant au polynôme **P** est illustrée dans la figure ci-dessous :



Chaque monôme de **Pol** est décrit par un enregistrement de type **Tmonome** comportant les 3 champs suivants : **Deg** (entier) représentant le degré du monôme, **Coef** (réel) représentant le coefficient du monôme, et **Suiv** (pointeur) pointant sur le monôme suivant. La liste **Pol** est triée par **ordre de degré décroissant**. On suppose qu'une liste vide (pointant sur NIL) correspond au polynôme zéro (*i.e.* elle ne contient aucun monôme).

1. Définir la structure **Tmonome**.
2. Écrire une procédure **Additionner** qui additionne les valeurs des coefficients de deux monômes **S** et **E**. Affecte la somme résultante au coefficient de **E** et libère **S**.
3. Écrire une procédure **Rechercher** qui recherche deux monômes **Q** et **P** dans la liste **Pol** d'un polynôme. Explicitement, **Q** est le premier monôme de **Pol** ayant le degré inférieure ou égale à un degré **D** donné, et **P** est le précédent de **Q** dans **Pol**. Si tous les monômes de **Pol** ont des degrés strictement supérieurs à **D**, alors **Q** pointera sur **NIL** et **P** pointera sur le monôme ayant le plus petit degré dans **Pol**.
4. Si on suppose que **Pol** contient **n** monômes, calculer la **complexité** de la procédure **Rechercher** et donner son **ordre de grandeur**.
5. Écrire une procédure **Ajouter** qui ajoute à la liste **Pol** d'un polynôme la valeur d'un monôme **M**. Attention aux différentes positions d'ajout possibles (début, milieu, et fin).
6. Écrire un **algorithme** qui construit une liste **Pol3** d'un polynôme **P3** obtenu par l'addition de deux polynômes **P1** et **P2**. L'algorithme doit utiliser les listes **Pol1** et **Pol2** (de **P1** et **P2** respectivement) qui sont supposées déclarées et remplies.

Soignez votre travail, afud igerzan !

Corrigé type

Exercice 1 :

1- Une fonction récursive qui calcule la somme de deux entiers positifs a et b : **(3 pts)**

```
Fonction Somme (a,b : entier) : entier ;
  Début
    Si (a=0) alors
        Somme ← b ;
    Sinon
        Somme ← Somme (a-1, b+1) ;
    Fin Si ;
Fin ;
```

2- Généralisation de la fonction aux entiers de signe quelconque : **(3,5 pts)**

```
Fonction Somme (a,b : entier) : entier ;
  Début
    Si (a=0) alors
       Somme ← b ;
    Sinon
       Si (a>0) alors
          Somme ← Somme (a-1, b+1) ;
       Sinon
          Somme ← Somme (a+1, b-1) ;
       Fin Si ;
    Fin Si ;
Fin ;
```

Exercice 2 :

1- La structure d'un monôme : **(1 pt)**

```
Type Tmonome=^Monome ;
       Monome=Enregistrement
            Deg : entier ;
            Coef : réel ;
            Suiv : Tmonome ;
       Fin ;
```

2- La procédure Additionner : **(1 pt)**

```
Procédure Additionner (E/S E, S : Tmonome) ;
Début
  Si (E^.Deg = S^.Deg) alors
     E^.Coef ← E^.Coef + S^.Coef;
     Libérer (S);
  Fin Si ;
Fin;
```

3- La procedure **Rechercher : (2,5 pts)**

```
Procédure Rechercher (E Pol : Tmonome; S  Q, P : Tmonome; E D : entier);
Var trouve : Booléen;
```

```
Début
   Q ← Pol ;
   Trouve ← faux ;
   Tant que (Q<>NIL) et (non trouve) faire
     Si (Q^.Deg > D) alors
         P ← Q ;
         Q←Q^.Suiv ;
     Sinon
         Trouve ← vrai ;
     Fin Si;
  Fin Tant que ;
Fin ;
```

4- La **complexité** de la procédure **Rechercher** : **(2,5 pts)**

La complexité algorithmique $C_{rechercher}$ est :

$C_{rechercher} = 1 + 1 + 3*(n+1) + (1+ \max(2,1)) * n = 2 + 3n+3 + 3n = 6n + 5$;

Donc, elle est d'ordre de grandeur **O(n) ;**

5- La procédure **Ajouter** (ajout au début, milieu, et à la fin) : **(3 pts)**

```
Procédure Ajouter (E/S Pol : Tmonome ; E M : Tmonome) ;
Var P, Q : Tmonome ;
Début
  Si (M^.Coef <> 0) alors
    Si (Pol=NIL) alors
       Pol ← M ;
       M^.Suiv ← NIL ;
    Sinon
      Si (M^.Deg>Pol.Deg) alors
          M^.Suiv ← Pol ;
          Pol ← M ;
      Sinon
        Si (Pol^.Deg=M^.Deg) alors
            Additionner (Pol, M) ;
            Si (Pol^.Coef=0) alors
                P ←Pol ;
                Pol← Pol^.Suiv ;
                Liberer(P) ;
            Fin Si ;
        Sinon
          Rechercher (Pol, Q, P, M^.Deg);
          Si (Q=NIL) alors
              P^.Suiv ← M ;
              M^.Suiv ← NIL ;
          Sinon
              Si (Q^.Deg=M^.Deg) alors
                  Additionner (Q, M) ;
                  Si (Q^.Coef=0) alors
                      P^.Suiv ← Q^.Suiv ;
                      Libérer (Q) ;
                  Fin si ;
              Sinon
                 P^.Suiv ← M ;
                 M^.Suiv ← Q ;
              Fin Si ;
```

```
            Fin Si ;
        Fin Si ;
    Fin Si ;
  Fin Si ;
Fin ;
```

6- L'algorithme principal : **(2,5 pts)**

```
Algorithme princ ;
Var
     Pol1, Pol2, Pol3, M, L1, L2 : Tmonome ;
// Déclarer les sous-algorithmes : Additionner, Rechercher, et Ajouter
Début
// Remplir les listes Pol1 et Pol2
Pol3←NIL ;
L1 ← Pol1 ;
 Tant que (L1<>NIL) faire
    Allouer (M) ;
    M^.Deg ← L1^.Deg ;
    M^.Coef ← L1^.Coef ;
    Ajouter (Pol3, M) ;
    M ←NIL ;
    L1 ← L1^.Suiv ;
 Fin Tant que ;
L2 ←Pol2 ;
Tant que (L2 <> NIL) faire
   Allouer (M) ;
   M^.Deg ← L2^.Deg ;
   M^.Coef ← L2^.Coef ;
   Ajouter (Pol3, M) ;
   M ←NIL ;
   L2 ← L2^.Suiv ;
 Fin Tant que ;
Fin ;
```