Langage C
TP n° 8 : Polynômes

Les exercices indiqués avec une ∗ sont à rendre.

Présentation. Un monôme est un polynôme de la forme cX^d, où c est le coefficient et $d \geqslant 1$ est le degré du monôme. Un polynôme peut donc être vu comme une somme de monômes. Le degré d'un polynôme est le degré maximum de ses monômes.

Nous allons représenter nos polynômes par des listes chainées. Chaque maillon représentera un monome $c_i X^i$, et on ne gardera que ceux pour lesquels c_i est non nul. La condition importante est que dans cette liste, les monômes doivent être triés par ordre décroissant de degré. Cette condition devra toujours être maintenue dans les programmes que nous allons écrire.

Les maillons d'un polynôme seront représentés à l'aide de la structure suivante :

```c
typedef struct maillon{
  int coef;
  int degre;
  struct maillon *suiv;
} maillon;
```

Un polynôme sera représenté par un pointeur de type `maillon *` valant l'adresse de son premier maillon. Par convention, le pointeur `NULL` représente le polynôme nul. Par exemple la Figure 1 représente le polynôme $3X^5 + 2X^2 + 1$:

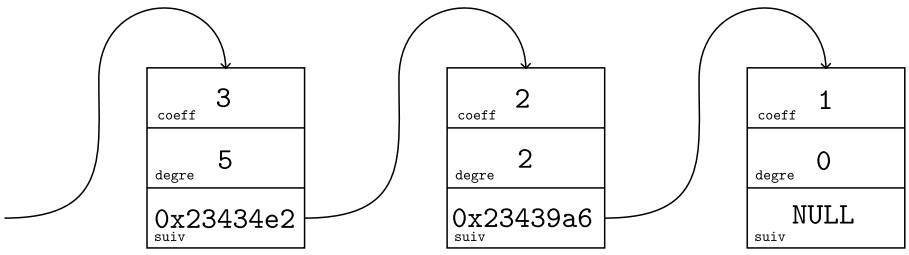

FIGURE 1 – Le polynôme $3X^5 + 2X^2 + 1$

Exercice 1 : Fonctions à écrire (∗)

Nous vous demandons de coder les fonctions suivantes. Servez-vous de la récurrence partout où son usage est naturel. On rappelle qu'il faut garantir le fait que les monômes d'un polynôme restent triés par degré décroissant *et* qu'il n'y a aucun monôme de coefficient 0.

1. `maillon *creer_monome(int c, int d)` crée un maillon de coefficient c et degré alloué par `malloc`, et renvoie l'adresse de ce maillon (*i.e.* renvoie un polynôme réduit à ce monôme).

2. `void liberer(maillon *p)` libère tout l'espace occupé par le polynôme p.

3. `double evaluer_polynome(maillon *p, double x)` retourne la valeur du polynôme `p` évalué en `x`. Rappelons que si `p` est un pointeur nul, il représente le polynôme nul.

4. `void afficher_polynome(maillon *p)` affiche le polynôme `p` sous forme standard. Ex : `3X^4-5X^3+X+1`.

5. `maillon *ajouter_monome(maillon *p , int c, int d)` ajoute un monôme cX^d au polynôme `p`, et renvoie l'adresse du premier maillon du polynôme résultant. Noter que cette adresse peut être différente de celle du premier maillon de `p`, par exemple si un nouveau maillon est ajouté en tête de `p`. Attention au cas où un monôme de même degré existe déjà dans `p`.

6. `maillon *copie(maillon *p)` retourne une copie du polynôme `p`.

7. `maillon *somme(maillon *p1, maillon *p2)` retourne un nouveau polynôme égal à la somme des polynômes `p1` et `p2`.

8. Modifiez votre `int main(int argc, char *argv[])` pour qu'il prenne en arguments deux entiers. Ces deux entiers représenteront le nombre de monômes contenus respectivement dans deux polynômes. A l'aide de `printf` et `scanf`, le programme demandera ensuite successivement à l'utilisateur de rentrer les coefficients et degrés des monômes. Enfin il affichera les deux polynômes et leur somme avant de les détruire et de finir.

Exercice 2 : Produit

Écrire enfin les deux fonctions suivantes, et compléter votre `main` de manière à afficher aussi le produit des deux polynômes entrés avant de les détruire et de finir.

1. `maillon *produit_monome(maillon *p, int c, int d)` retourne un nouveau polynôme représentant le produit $p \times cX^d$.

2. `maillon *produit(maillon *p1, maillon *p2)` retourne un nouveau polynôme représentant le produit $p1 \times p2$.