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 du polynôme est le degré maximum des monômes dont il est la somme. On va 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.
On aura donc la structure suivante :

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

Afin de faciliter l'écriture de certaines fonctions, on fait de plus le choix suivant : on aura toujours en tête de liste un maillon "'fictif" représentant la tête de liste : son coefficient sera 0 et son degré sera -1.
Par exemple la Figure **??** représente le polynôme $3X^5 + 2X^2 + 1$:

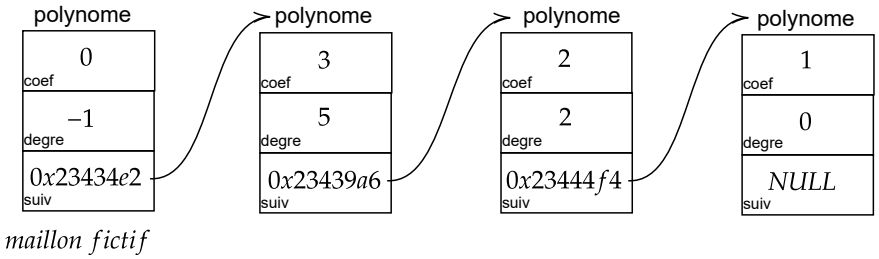

FIGURE 1 – Le polynôme $3X^5 + 2X^2 + 1$

Exercice 1 : Fonctions à écrire $(*)$

Nous vous demandons de coder les fonctions suivantes. Pensez à les tester dès que possible. Attention : pour ces fonctions, on rappelle qu'il faudra 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. `polynome* creer_polynome_vide()` crée le premier maillon factice et retourne son adresse.

2. `polynome* creer_monome(int c, int d)` crée un maillon de coefficient c et degré d. **Attention**, cette fonction renvoie bien un **maillon** et non un polynôme conventionnel puisqu'il lui manque le maillon fictif en tête de liste.

3. `void polynome_destroy(polynome* p)` libère tout l'espace occupé par le polynôme p.

4. `double evaluer_polynome(polynome* p,double x)` retourne la valeur du polynôme p évalué en x.

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

6. `void ajouter_monome (polynome* p , int c, int d)` ajoute un monôme cX^d au polynôme p. Attention notamment au cas où un monôme de même degré existe déjà dans p.

7. `polynome* copie(polynome* p)` retourne une copie du polynôme p.

8. `polynome* somme(polynome* p1, polynome* p2)` retourne un nouveau polynôme correspondant à la somme $p1 + p2$.

9. 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

1. `polynome* produit_monome(polynome* p, int c, int d)` retourne un nouveau polynôme correspondant au produit $p \times cX^d$.

2. `polynome* produit(polynome* p1, polynome* p2)` retourne un nouveau polynôme correspondant au produit $p1 \times p2$.

3. Le programme affichera également le produit des deux polynômes renseignés avant de les détruire et de finir.