

# Travaux Dirigés de programmation n°1

## Cours de Programmation C

—L2.1—

### ► Exercice 1. Polynômes

Dans toutes les réponses, une attention particulière sera donnée à la gestion de la mémoire.

On décide de représenter des polynômes à coefficients entiers à une variable par des listes simplement chaînées de monômes, chaque monôme étant représenté par une structure constituée d'un entier `coef` pour le coefficient du monôme, d'un entier `degre` pour le degré du monôme, et d'un pointeur sur le monôme suivant. Un polynôme ne contient pas 2 monômes de même degré.

On ordonne les monômes par degré croissant.

1. Définir les types `Monome` et `Polynome`
2. Écrire la liste ordonnée correspondant au polynôme :  $12 + x + 3x^4 - 5x^{13}$
3. Écrire une fonction `float EvaluePolynome(Polynome p, float val)` qui renvoie la valeur de `p` pour `x= val`.
4. Écrire une fonction `int AjouteMonome(Monome *m, Polynome *p)` qui ajoute le monôme `*m` au polynôme `*p`. Un polynôme ne doit pas avoir deux monômes avec le même degré, cependant, on ne traitera pas pour l'instant le cas des monôme à coefficient nul
5. Écrire une fonction `int SommePolynome(Polynome p, Polynome q, Polynome *res)` qui place à l'adresse `res` le `Polynome` égal à  $p + q$ . La fonction renvoie 0 en cas d'échec, 1 sinon. On suppose que les deux polynomes envoyés sont correctement formatés : ils ne contiennent pas de monomes avec un coefficient nul sauf s'ils sont le polynôme nul et dans ce cas, ils ne contiennent qu'un seul monôme de coefficient 0 et de degré 0. Le résultat obtenu doit avoir le même format.
6. Écrire une fonction `int prod(Monome m, Polynome p, Polynome *res)` qui place à l'adresse `res` le `Polynome` égal au produit du monôme et du polynôme. La fonction renvoie 0 en cas d'échec d'allocation, 1 sinon.
7. Écrire une fonction `int ProduitPolynome(Polynome p, Polynome q, Polynome *res)` qui place à l'adresse `res` le polynôme égal à  $p * q$ . La fonction renvoie 0 en cas d'échec, 1 sinon.

### ► Exercice 2. Couleurs

On travaille à la création d'un logiciel simple de dessin. Pour représenter la zone de dessin, on utilise une grille de pixels dont chacun contient une couleur sous la forme de trois valeurs `R V B`.

1. On veut que la grille soit représentée sous la forme d'un tableau à deux dimensions de pixels. Ecrire les définitions des types `Pixel` et `Grille`.
2. Ecrire la fonction d'allocation et de désallocation de la grille.
3. Ecrire la fonction `Remplir(Grille G, int ligne, int col, Pixel couleur)` qui modifie la couleur du pixel situé en position `G[ligne,col]` ainsi que de tous les pixels pouvant être atteints par voisinage de même couleur, pour leur donner la couleur envoyée en argument de la fonction.