

IAP1 - feuille de TP n0 2

Représentation creuse des polynômes -

1 Polynômes à une indéterminée

L'objet de cet exercice est la manipulation de polynômes à une variable, codés par des listes. On parle alors de polynômes creux. Un monôme est représenté par un couple dont le premier composant est le coefficient et le second composant le degré du monôme. Dans la suite le type `polynome` désigne le type `int*int list`.

On impose deux conditions : les monômes d'un polynôme devront toujours être triés par degré décroissant ; aucun coefficient ne doit être nul. Le polynôme nul est représenté par la liste vide []. Les monômes ont tous des degrés différents.

Ces contraintes seront à prendre en considération dans les fonctions qui manipuleront les polynômes, il est, en effet, impossible de les *faire entrer* dans les déclarations de type. Ces contraintes doivent être respectées à chaque manipulation de polynôme. On appelle ces contraintes un *invariant de représentation*.

Par exemple le polynôme $X^5 - 2X^4 + 1$ sera représenté par la liste [(1,5);((-2),4); (1,0)]

1. Écrivez la fonction `dg_max : polynome → int`. Elle retourne le degré du polynôme, degré maximal des monômes. Par convention, on posera que le degré du polynôme nul est -1.
2. Écrivez une fonction `opposé: polynome → polynome` qui renvoie le polynôme dont les coefficients sont les opposés des coefficients du polynôme paramètre. On vérifiera (informellement) que l'invariant de représentation est bien conservé.
3. Pour rendre plus lisible ces polynômes, écrire la fonction `string_of_polynome` qui les transforme en chaînes de caractères. Elle prend en paramètre le polynôme lui-même et le nom de l'indéterminée. Elle suit les règles usuelles de présentation des polynômes. Néanmoins, le monôme aX^n sera mis sous la forme aX^n (le coefficient ne sera pas écrit s'il vaut 1). On commence par écrire la fonction `string_of_monome` qui traduit un monôme en une chaîne de caractères. On utilise, dans le texte de cette fonction, la fonction prédéfinie `string_of_int`, qui transforme un entier en la chaîne de caractères correspondante. Par exemple, `string_of_int 4` vaut "4".
4. Écrivez une fonction `somme: polynome * polynome → polynome` qui renvoie le polynôme somme des deux polynômes paramètres. On vérifiera (informellement) que l'invariant de représentation est bien conservé.
On peut assez facilement atteindre une complexité linéaire en s'inspirant de la fusion de deux listes triées (en effet, les polynômes sont des listes de monômes ordonnés selon les degrés décroissants).
5. Écrivez une fonction `deriver: polynome → polynome` qui renvoie le polynôme dérivé du polynôme passé en argument. On vérifiera (informellement) que l'invariant de représentation est bien conservé.
6. Écrivez une fonction `produit: polynome * polynome → polynome` qui renvoie le produit des polynômes passés en arguments. On vérifiera (informellement) que l'invariant de représentation est bien conservé.
7. Écrivez une fonction `evaluer: polynome * int → int` qui calcule la valeur d'un polynôme en un point. Vous chercherez une solution efficace en vous inspirant de la méthode de Horner.

2 Polynômes à deux indéterminées

On considère maintenant un polynôme à deux indéterminées X et Y comme un polynôme à une indéterminée X dont les coefficients sont des polynômes à une indéterminée Y .

On adopte les mêmes conventions que ci-dessus.

1. Quel est le type de la représentation d'un tel polynôme ?

2. Comment représentez-vous le polynôme $X^4 + 2X^2Y + 5X^2 + 5Y^2$? Il est considéré comme le polynôme suivant : $X^4 + X^2(2Y + 5) + 5Y^2$.
3. Écrire une fonction `somme2` qui réalise la somme de deux polynômes en X et Y (supposés bien formés). Le résultat sera encore un polynôme bien formé.
4. Ecrire une fonction qui permet de transformer un polynôme à deux indéterminées en une chaîne de caractères.