POLYNÔMES LACUNAIRES

On représente généralement les polynômes par des tableaux. Mais quand la plupart des coefficients sont nuls, on enregistre en mémoire des tas de coefficients inutiles : le polynôme $X^{100} + 12X^{49} - 35$ nécessite un tableau de 101 cases, alors qu'il suffit de ne connaître que 6 nombres pour connaître le polynôme (les 3 degrés $100, 49, 0$ et les 3 coefficients $1, 12, -35$).

Puisqu'il faut fixer le type des coefficients, nous considèrerons des polynômes à coefficients flottants.

Un polynôme peut être défini de manière récursive de la façon suivante :
– le polynôme nul est un polynôme
– si p est un polynôme et m un monôme aX^d, alors $m + p$ est un polynôme.
Nous allons donc représenter les polynômes par des assemblages de monômes. Plutôt que les couples classique de CAML, nous utiliserons une des fonctionnalités de CAML qui est de pouvoir créer de nouveaux types en plus des types natifs comme `int`, `float`, etc. Ici nous utiliserons un type "enregistrement" ("record", dans d'autres langages) :

```
(* définition du type monôme : enregistrement à 2 champs entier et flottant *)
type monome = {deg : int; coef : float};;

(* création d'un monôme qui représente -12. X^2 *)
let m = {deg = 2; coef = (-12.) };;

(* un monôme étant connu, on peut récupérer son degré et son coefficient *)
let d = m.deg and c = m.coef;;
```

Puis le type polynôme

```
type polynome = Nul | Somme of (monome * polynome);;
```

Donc pour notre étude, un monôme est un enregistrement du type précédent et un polynôme est soit Nul, soit une somme d'un monôme et d'un polynôme (remarque : cette somme n'est pas forcément ordonnée). **Les coefficients des monômes ne seront jamais nuls. Un polynôme ne contient jamais deux monômes de même degré.**

1 Création et affichage d'un polynôme

Nous créons d'abord une fonction poly liste de type `monome list -> polynome`, qui transforme une liste de monômes en un polynôme.

```
let p =
  let m1 = {deg = 1; coef = (-1.)} and m2 = {deg = 2; coef = (-5.2)} in
  poly [m1; m2];;
(* réponse de CAML : *)
Somme({deg = 1; coef = (-1.)}, Somme({deg = 2; coef = (-5.2)}, Nul))
```

Nous créons ensuite une fonction pour afficher un monôme de façon plus lisible. Écrivez donc une fonction print_monome de type `monome -> string` telle que :
– si m est un monôme de degré 0 et de coefficient c, alors on calcule la chaîne "(C)␣", où dans la chaîne, le symbole C désigne la valeur du flottant c convertie en chaîne de caractères
– si m est un monôme de degré 1 et de coefficient c, alors on calcule la chaîne "(C)␣X␣"
– si m est un monôme de degré au moins 2 et de coefficient c, alors on calcule la chaîne "(C)␣X^D␣", où D désigne la valeur de l'entier d convertie en chaîne de caractères
(le symbole "␣" visualise le caractère espace). Dans l'aide-mémoire, vous trouverez les fonctions nécessaires à la conversion d'entier ou de flottant en chaîne de caractère.

```
print_monome {deg=0; coef=2.};;
(* réponse de CAML : *) (2.)
print_monome {deg=1; coef=(-32.)};;
(* réponse de CAML : *) (-32.) X
print_monome {deg=5; coef=3.4};;
(* réponse de CAML : *) (3.4) X^5
```

Écrivez enfin une fonction récursive print_poly de type `polynome -> string`, qui écrit un polynôme sous forme lisible :

```
print_poly p;;
(* réponse de CAML : *) (-5.2) X^2 + (-1.) X
print_poly Nul;;
(* réponse de CAML : *) (0.)
print_poly (poly [ {deg=0; coef=2.} ]);;
(* réponse de CAML : *) (2.)
```

2 Opérations sur les polynômes

2.1 Produit par un scalaire

Écrivez une fonction récursive prod_scal de type `float -> polynome -> polynome`, qui calcule le produit d'un polynôme par un scalaire.

2.2 Polynôme dérivé

Écrivez une fonction récursive derive de type `polynome -> polynome`, qui calcule le polynôme dérivé d'un polynôme.

2.3 Addition

Écrivez une fonction récursive somme_m_p de type `monome -> polynome -> polynome`, qui calcule la somme d'un monôme et d'un polynôme.

Écrivez une fonction récursive somme qui calcule la somme de deux polynômes et une fonction qui calcule la différence.

2.4 Multiplication

Écrivez une fonction récursive produit qui calcule le produit de deux polynômes.

3 Division euclidienne

Écrivez une fonction dominant, qui donne le monôme dominant d'un polynôme (*i.e.* le monôme de plus haut degré) s'il existe.

Écrivez une fonction euclide de type `polynome -> polynome -> (polynome * polynome)`, qui calcule le quotient et le reste de la division euclidienne de p par q.

4 S'il vous reste du temps...

Écrivez une fonction de définition d'un polynôme, qui permet à un utilisateur de saisir un polynôme sous sa forme « naturelle »(chaîne de caractères). Par exemple, plutôt que de définir le polynôme par

```
let p = poly [{deg = 1; coef = (-2.)}; {deg = 5; coef = 3.}];;
```

on le définit par

```
let p = polynome "(-2.) X + (3.) X^5";;
```

5 Travail à rendre

Donnez vos fonctions de sommes et de produits, démontrez leur correction et précisez leur complexité (en fonction des longueurs des polynômes, c'est-à-dire de leur nombre de monômes).