Travaux Dirigés n° 3

Exercice 1 *(Types somme)*

On souhaite définir les opérations de manipulation d'entiers signés. On définit le type `entier_signe` par :

```
type entier_signe = Pos of int | Neg of int ;;
```

1 Écrire une fonction `creer_signe`, qui prend un argument `n` de type `int` et rend la valeur `Pos(n)` si (`n >= 0`) et `Neg(-n)` dans le cas contraire.

2 On souhaite que les constructeurs `Pos` et `Neg` ne soient appliqués qu'à des entiers positifs ou nuls. Écrire une fonction `correct` qui vérifie cette condition. Une valeur `v`, telle que `correct(v) = true`, est dite correcte.

3 Écrire les fonctions `plus_signe`, `moins_signe` et `mult_signe` qui prennent un couple d'arguments de type `entier_signe` et, si ces arguments sont corrects, effectuent les opérations indiquées dans leur nom. (Indication : la construction (`let rec ... and ...`) permet de définir des fonctions mutuellement récursives)

Exercice 2 *(Évaluation des tableaux)*

Évaluez les expressions ci-dessous. On supposera qu'à chaque question correspond une nouvelle session **CAML** interactive.

1
```
let t1 = init_vect 4 (function i -> i);;
t1.(1) <- 5;;
let t2 = make_vect 2 0;;
t1 = t2;;

let m = make_matrix 3 4 0;;
m.(2).(1) <- 5;;
m;;
m.(1) <- [| 1;1;1 |];;
let n = init_vect 3 (function i -> init_vect 4 (function j -> i + j));;
```

2
```
let t3 = [|1;2;3|];;
t3.(2) <- 2;;
t3;;
let t4 = [|3;4|];;
let t5 = t4;;
t4.(1) <- 2;;
t5;;
```

Exercice 3 *(Vecteurs)*

On se propose de définir des fonctions de calcul sur les polynômes à une variable et à coefficients entiers. Pour cela, on représentera un polynôme

$$P(X) = \Sigma_{i=0}^{n} a_i.X^i$$

par un vecteur `p` de taille $n+1$ tel que `p.(i)`$= a_i$.

1 Écrire une fonction d'évaluation d'un polynôme en un point. Pour cela, vous remarquerez que :

$$\Sigma_{i=0}^{n} a_i.X^i = a_0 + X.(\Sigma_{i=1}^{n} a_i.X^{i-1})$$

2 Écrire une fonction de multiplication d'un polynôme par une constante.

3 Écrire une fonction calculant la somme de deux polynômes.

4 Écrire une fonction calculant la difference de deux polynômes.

5 Écrire une fonction permettant d'afficher un polynôme.

Exercice 4 *(Tableaux)*

Dans un match de football, chaque équipe joue un match à domicile, un match chez l'adversaire. Chaque match gagné vaut deux points, chaque match nul un point et chaque match perdu zero point. Le gagnant est celui qui totalise le plus grand nombre de points. On ne traitera pas ici le cas où deux équipes sont ex-aequo.

Le tableau suivant récapitule des résultats du championnat. Les lignes horizontales marquent les résultats à domicile, les lignes verticales les résultats à l'extérieur. Un match gagné sera marqué `g`, un match perdu `p` et un match nul `n`.

	Auxerre	Bastia	Bordeaux	Lens	Marseille
Auxerre			`p`	`p`	`p`
Bastia	`g`				
Bordeaux	`p`			`g`	
Lens			`g`		
Marseille		`n`			

1 Proposez une structure de données permettant de représenter les équipes ainsi que les résultats du championnat.

2 Écrire une fonction qui met à jour les résultats d'une série de matchs. Vous écrirez d'abord une fonction `ajout_match` qui permet d'ajouter les résultats d'un match puis une fonction `ajout_list_de_match` permettant d'ajouter les résultats d'une liste de matchs.

3 Écrire une fonction qui effectue une correction des résultats (en cas de contestation).

4 Écrire une fonction qui imprime les résultats acquis depuis le championnat.

5 Écrire une fonction qui calcule le nombre de matchs gagnés par une équipe.

6 Écrire une fonction qui donne le classement des équipes et l'affiche. Vous regrouperez les équipes ex-aequo.

Exercice 5 *(Listes)*

On souhaite calculer la liste des entiers premiers plus petits que `n` par la méthode du *crible d'Eratosthene*. Pour cela, vous définirez et donnerez le type des fonctions suivantes :

1 la fonction `enlever_multiples` enlève les éléments d'une liste d'entiers `l` multiples d'un entier `n`. Vous proposerez deux versions différentes, dont une utilisant la fonction `filter` de la librairie standard.

2 la fonction `intervalle` calcule la liste des entiers naturels compris entre `n` et `m`. Vous définirez alors la fonction `nat2` rendant la liste des entiers de `2` à `m`.

3 la fonction `crible` calcule les premiers nombres premiers plus petits que `n`.