

Option Informatique en Spé MP et MP*

Programmation Caml : simulation d'un tirage sans remise

Question 1 Rédigez en Caml une fonction :

```
decoupe : int -> 'a list -> ('a * 'a list)
```

spécifiée comme suit : `decoupe k l` rend le couple dont la première composante est le k -ième élément de la liste ℓ , et dont la deuxième composante est la liste déduite de ℓ par suppression de son k -ième élément. Par exemple, `decoupe 3 [9;2;7;10;5;2]` rendra le couple $(10, [9;2;7;5;2])$. Vous lèverez une exception si $k \geq |\ell|$.

► Le générateur aléatoire de Caml `random__int` est une fonction de type `int -> int`, spécifiée comme suit : `random__int n` est une variable aléatoire uniformément répartie dans l'intervalle discret $\llbracket 0, n-1 \rrbracket$. Deux appels consécutifs de `random__int` sont indépendants, au moins en apparence.

Question 2 Rédigez en Caml une fonction :

```
alea_entre : int -> int -> int
```

spécifiée comme suit : `alea_entre d f` est une variable aléatoire uniformément répartie dans l'intervalle discret $\llbracket d, f \rrbracket$. Vous lèverez une exception si $d \geq f$.

Question 3 Expliquez la phrase «deux appels consécutifs de `random__int` sont indépendants, au moins en apparence».

Question 4 Rédigez en Caml une fonction :

```
extrait : 'a list -> ('a * 'a list)
```

spécifiée comme suit : `extrait l` rend un couple (t, q) où t est un élément de ℓ choisi au hasard, et q est la liste déduite de ℓ par suppression de cet élément. Vous lèverez une exception si ℓ est la liste vide.

► Nous nous proposons de rédiger en Caml une fonction :

```
tirage_sans_remise : int -> 'a list -> 'a list
```

spécifiée comme suit : `tirage_sans_remise k l` rend une liste de k éléments choisis au hasard dans la liste ℓ .

► L'étudiant Jean-Benoît MALENCONTREUX a eu l'idée suivante : il choisit au hasard un élément x de ℓ ; si x n'appartient pas à la liste résultat r en cours de construction, il l'y ajoute, sinon il recommence. Le travail est terminé lorsque $|r| = k$.

Question 5 Mettez en œuvre l'idée de Jean-Benoît ; vous pourrez utiliser `mem`, pour tester l'appartenance à une liste.

Question 6 La solution de notre ami présente deux gros défauts. Lesquels ?

Question 7 Rédigez en Caml une solution corecte. Vous lèverez une exception si $k \geq |\ell|$.

Question 8 Rédigez en Caml une fonction :

```
produit_cartesien : 'a list -> 'b list -> ('a * 'b) list
```

spécifiée comme suit : `produit_cartesien x y` rend la liste (dans un ordre quelconque) des couples (x_i, y_j) où x_i (resp. y_j) est un membre de la liste x (resp. y). Par exemple, `produit_cartesien [1;2;1] ["a";"b"]` rendra (à l'ordre près) la liste $[(1, "a"); (2, "a"); (1, "a"); (1, "b"); (2, "b"); (1, "b")]$.

► Rappelons brièvement les règles du jeu *Démineur* : vous devez explorer complètement un champ de mines, divisé en *cases* carrées, toutes de même taille. Lorsque vous sondez une case, deux cas peuvent se produire : si cette case est minée, la mine explose, et le jeu est fini. Sinon, l'ordinateur indique le nombre (compris entre 0 et 8) de cases voisines minées.

► Un champ de mines sera représenté par une matrice d'éléments du type `bool`, une case minée étant signalée par la valeur `true`.

Question 9 Rédigez en Caml une fonction :

```
champ_de_mines : int -> int -> int -> bool vect vect
```

spécifiée comme suit : `champ_de_mines n p m` crée un champ de $n \times p$, contenant m mines. Vous lèverez une exception si le nombre de mines est supérieur au nombre de cases.

FIN