Introduction à la programmation fonctionnelle

TP 3 (Noté)

Ce TP couvre les séances du **27 mars** et du **3 et 10 avril**.
Il est à rendre par mail à votre chargé de TP (`sylvain.conchon@lri.fr` ou `alice.jacquot@lri.fr`) le **vendredi 10 avril à 15h30**.

Ne changez pas de groupe de TP entre ces deux séances.

Important: Vous devez tester vos fonctions et votre programme au fur et à mesure. Indiquez les tests effectués et les résultats observés, par exemple en commentaire. Cela comptera pour une partie significative de votre note. Veillez à fournir un exécutable qui affiche les résultats pertinents sur la sortie standard. Vous pouvez pour cela utiliser une fonction `main : unit -> unit`, et l'appeler en fin de programme pour qu'elle soit toujours exécutée. Vous pouvez définir plusieurs fonctions main pour garder trace de vos tests précédents (c'est la dernière définition qui sera appelée).

Échauffement : listes

- Quel est le type de la fonction suivante ?

```
let rec f x y =
  match x with
    | [] -> true
    | (0, _ ) :: _ -> false
    | (a, _) :: _ when a=y ->  false
    | _ :: t -> f t y
;;
```

- Écrire une fonction qui prend une liste d'entiers et renvoie la somme de ses valeurs.

- Écrire une version récursive terminale de la fonction précédente.

  Indication : Utiliser une fonction auxiliaire admettant un argument supplémentaire correspondant à la somme partielle déjà calculée.

Un jeu simplifié de bataille

On souhaite réaliser un programme qui joue contre lui-même à une version simplifiées du jeu de carte "la bataille". Les deux joueurs commencent avec chacun la moitié d'un paquet de cartes et les retournent une à une en comparant les valeurs. Celui qui a la carte la plus forte marque un point. En cas d'égalité, personne ne marque. Les deux joueurs passent ainsi les deux paquets. Le joueur à la fin le score le plus élevé gagne.

Question 1

Construire les typres `valeur`, `couleur`, `carte` et `paquet` qui vont servir pour construire le jeu de cartes (un jeu classique avec As, rois, dames, etc. et les quatre couleurs)

Question 2

Écrire une fonction `compare : carte -> carte -> int` qui étant données deux cartes renvoie :

- l'entier `-1` si la première est *strictement inférieure* à la deuxième
- l'entier `1` si la première est *strictement supérieure* à la deuxième
- l'entier `0` en cas d'égalité.

Question 3

Écrire une fonction `string_of_carte : carte -> string` qui transforme une carte en une chaîne de caractères décrivant sa valeur et sa couleur.

Question 4

Définir une liste `p` de type `paquet` contenant les 52 cartes d'un jeu classique.

Question 5

Écrire une fonction `mélange : paquet -> paquet` qui mélange aléatoirement les cartes d'un paquet.

Question 6

Écrire une fonction `partage : paquet -> paquet * paquet` qui sépare un paquet de cartes en deux paquets de même taille (on supposera que la taille du paquet passé en argument est paire).

Question 7

Écrire une fonction `face_a_face : int * int -> carte -> carte -> int * int` qui prend un couple de scores (des valeurs de type `int`) ainsi que deux cartes et augmente les scores de 1 point selon la valeur des cartes (le premier score est celui associé à la première carte).

Question 8

Utiliser la fonction précédente pour écrire une fonction `face_a_face_texte : int * int -> carte -> carte -> int * int` qui se comporte de la même façon que `face_a_face` mais affiche en plus un message sur la sortie standard contenant le score avant la confrontation ainsi que les deux cartes.

Question 9

Écrire une fonction `bataille : unit -> int * int` qui exécute une partie de bataille et renvoie le score final sous la forme d'un couple d'entiers. La fonction mélange un paquet de cartes, le divise en deux paquets puis exécute les face à face jusqu'à épuisement des paquets en gardant le score en mémoire.

Question 10

Écrire une fonction `stat : int -> int` qui prend en argument un nombre de parties à effectuer et les exécute. La fonction affiche au final les statistiques suivantes (pour chaque joueur):

- le nombre de victoires
- le score cumulé
- le score moyen
- le taux de victoires