Feuille de travaux pratiques
Listes

Exercice 1 (Cartes suite)

Cet exercice reprend l'exercice du TP3. On suppose définis (au moins) :

- le type `carte`,
- la fonction `compare`,
- la fonction `score_carte`.

Le but de cet exercice est de programmer un jeu de bataille ouverte, entre deux joueurs, ayant chacun reçu la moitié d'un jeu de 52 cartes mélangé. A chaque tour de jeu, les joueurs choisissent chacun une carte, et celui qui a la carte la plus forte remporte le pli, les cartes revenant dans son jeu. Chaque joueur peut décider d'une stratégie qui consiste à choisir astucieusement dans quel ordre il place ses cartes.

Dans un premier temps, on mélange aléatoirement le jeu de 52 cartes. On supposera que toutes les cartes existant dans le jeu sont rangées dans une liste. La méthode utilisée est la suivante : on mélange la liste en échangeant aléatoirement des couples de cartes (swap). Attention, toutes les fonctions ci-dessous doivent être écrites de façon à traiter des listes polymorphes.

On pourra utiliser les fonctions prédéfinies sur les listes, consultables ici : `https://caml.inria.fr/pub/docs/manual-ocaml/libref/List.html`.

1. Ecrire une expression (la plus compacte possible !) `deck` dont la valeur est la liste des cartes possibles.
2. Ecrire une fonction `echange l i` qui, étant donné une liste `l`, renvoie une liste identique à `l` à l'exception de la `i`-ième valeur qui sera placée en tête de liste. Par exemple, `echange [1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13] 7` vaut `[8; 1; 2; 3; 4; 5; 6; 7; 9; 10; 11; 12; 13]`.
3. Ecrire une fonction `melange` qui mélange une liste en appliquant un certain nombre de `echange` sur des couples choisis aléatoirement.
4. Ecrire une fonction `distribue l` qui mélange la liste `l`, puis la distribue en 2 listes (les jeux de chaque joueur) de la façon suivante : la première liste reçoit la première carte, la deuxième la deuxième carte, la première liste reçoit la troisième carte, etc.

On implémente ensuite les tours de jeu à proprement parler. On considère pour l'instant que chaque joueur joue la première carte de la liste qui représente son jeu, sans stratégie particulière.

5. Proposer une méthode pour casser les cas d'égalité - par exemple, que se passe-t-il si un joueur joue le 3 de carreau, et l'autre le 3 de pique ?

6. Ecrire une fonction `un_tour` qui prend en entrée deux listes de cartes (ou un couple), détermine la carte gagnante (en appliquant la méthode proposée ci-dessus), et renvoie les deux nouveaux jeu de cartes, celui du perdant privé de la carte jouée, celui du gagnant augmenté de la carte qu'il vient de gagner.
7. Ecrire une fonction `bataille_ouverte` qui implémente l'ensemble du jeu : distribution des cartes, tours de jeu jusqu'à ce qu'un des joueurs n'aient plus de cartes.

Enfin, on souhaite ajouter des stratégies au jeu de bataille ouverte. Pour cela, il faut que chaque joueur ait une fonction lui permettant de choisir la carte qu'il joue, en fonction de son jeu. On implémentera la stratégie par un tri du jeu de cartes pour un ordre particulier, qui correspond à la stratégie : ainsi, si un joueur veut commencer par jouer ses cartes les plus fortes, on triera le jeu par ordre décroissant. S'il veut jouer aléatoirement, il triera son jeu aléatoirement, par exemple en le gardant dans son état initial. Attention, les stratégies ne dépendent que des cartes dans la main du joueur et pas des tours de jeu précédents.

8. Proposer plusieurs variante d'une fonction `strategie : carte list -> carte` permettant à un joueur de choisir la carte qu'il doit jouer.
9. Reprendre la fonction `bataille_ouverte` de façon à pouvoir lui passer les stratégies des deux joueurs en paramètre.