Lambda calcul et programmation fonctionnelle

TD 3

Exercice 1

Écrivez une fonction `estPremier` qui permet de savoir si un nombre passé en paramètre est premier. Un nombre est premier s'il n'est divisible que par lui-même ou par un. Par convention, 1 n'est pas un nombre premier.

Un nombre a est divisible par un autre nombre b ssi le reste de la division entière de a par b est nul. Le reste de la division entière est donné, en Haskell, par la fonction mod (modulo) :

```
mod :: Int -> Int -> Int
```

Écrivez une fonction `premiersPremiers` qui prend en paramètres un entier n et renvoie la liste de tous les nombres premiers entre 1 et n. Par exemple, l'appel suivant renverra :

```
premiersPremiers 20
> [3, 5, 7, 11, 13, 17, 19]
```

En Haskell, il est possible de créer une liste contenant tous les entiers de i à j avec la syntaxe `[i..j]`. Par exemple, la liste de tous les entiers de 1 à 10 s'écrit `[1..10]` (ou `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`).

Réécrivez la fonction `premiersPremiers` en utilisant la syntaxe ci-dessus et une fonction d'ordre supérieur. Rappel : on veut savoir, parmi tous les entiers entre 1 et n, lesquels sont premiers ou non.

Exercice 2

On veut écrire une fonction qui détermine le vainqueur d'une partie de pierre-feuille-ciseau.

Le jeu se joue à deux joueurs. À chaque manche, chaque joueur choisit un coup parmi les trois coups possibles : pierre, feuille, ciseaux. La pierre bat les ciseaux, les ciseaux battent la feuille, la feuille bat la pierre. Si les deux joueurs jouent la même chose, c'est une égalité.

Écrivez une fonction `résultatManche` qui prend un tuple de deux coups en paramètres et renvoie le résultat de l'affrontement. Si le joueur 1 gagne, le résultat sera 1. S'il perd, le résultat sera -1. En cas d'égalité, le résultat sera 0. Vous devrez très probablement créer un nouveau type.

Écrivez une fonction `deathMatch` qui prend en paramètres une liste de plusieurs manches et renvoie le score final.