DEVOIR : FONCTIONS SUR LES LISTES

Représentation d'une fonction unaire à domaine fini par une liste de couples dont les 1er éléments sont tous différents.

On décide de représenter une fonction unaire `f`, définie sur un domaine fini $\mathcal{D}$, par la liste des couples (x , `f`(x)) pour x appartenant au domaine $\mathcal{D}$.

Par exemple, la fonction `f`, définie sur l'intervalle [1,5], par `f`(x) = 2x +1 est représentée par la liste [(1 , 3) ; (2 , 5) ; (3 , 7) ; (4 , 9) ; (5 , 11)]

1. Ecrire une fonction `estFonction` qui, étant donnée une liste de couples, vérifie que la liste des premiers éléments ne contient pas de duplication.

On suppose maintenant qu'on a bien des fonctions.

2. Ecrire la fonction `image` qui, étant donnés un élément et une liste de couples (représentant `f`) retourne la valeur associée à l'élément si elle existe.

3. Ecrire la fonction `imageEns` qui, étant données une liste d'éléments L et une liste de couples (représentant une fonction `f`) retourne la liste des valeurs associées à chaque élément de L.

4. Ecrire `estInjective` qui, appliquée à une fonction `f` représentée par une liste de couples, vérifie que 2 éléments n'ont pas la même image.

5. Ecrire la fonction `surcharge`, prenant 2 listes représentant les fonctions `f1` et `f2`, et retournant une liste représentant la fonction `f` définie sur l'union des domaines de définition de `f1` et `f2`, et dont l'image d'un élément est donné soit par `f1`, soit par `f2`, en donnant priorité à `f2`.

6. Ecrire la fonction `composition`, prenant 2 listes représentant les fonctions `f1` et `f2`, et retournant une liste représentant la fonction `f` dont l'image d'un élément est donné par `f1` appliquée à l'image par `f2` de cet élément.

7. Ecrire la fonction `produit`, prenant en argument 2 listes représentant 2 fonctions `f1` et `f2`, et retournant la liste représentant la fonction qui à un couple (x , y) associe le couple (`f1` x , `f2` y).

On pourra écrire toute fonction auxiliaire jugée nécessaire si elle est spécifiée et commentée.

Vous devez respecter absolument les profils des fonctions suivants :

```
val estFonction : ('a * 'b) list -> bool = <fun>
val image : 'a -> ('a * 'b) list -> 'b = <fun>
val imageEns : ('a * 'b) list -> 'a list -> 'b list = <fun>
val estInjective : ('a * 'b) list -> bool = <fun>
val surcharge : ('a * 'b) list -> ('a * 'b) list -> ('a * 'b) list = <fun>
val composition : ('a * 'b) list -> ('c * 'a) list -> ('c * 'b) list = <fun>
val produit : ('a * 'b) list -> ('c * 'd) list -> (('a * 'c) * ('b * 'd)) list = <fun>
```