LMI 1 - Semestre 1

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO1 – Algorithmique et Programmation 1

Fiche de TP numéro 6 - Dictionnaires

Le contenu de votre réfrigérateur est mémorisé dans une structure du type `TIngredients = dico(string:int)` où la clé est le nom d'un produit alimentaire, et la valeur la quantité de ce produit. Une recette de cuisine est mémorisée dans une structure de même type (on ne s'intéresse ici qu'aux ingrédients nécessaires à la réalisation d'une recette). Une collection de recettes est une structure du type `TRecettes = dico(string;TIngredients)` où la clé est le titre d'une recette.

Exemples :

```
# TIngredients : dico(str,int)
# TRecettes : dico(str,TIngredients)
# On supposera les variables suivantes définies pour les exemples
>>> gateauChoc = {'oeufs':4,'sucre':150,'farine':80,\
                  'beurre':200,'chocolat':200}
>>> quatreQuarts = {'oeufs':4,'sucre':250,'farine':250,'beurre':250}
>>> lesRecettes = {'omelette' :{ 'oeufs': 4, 'lait (en cl)': 5},\
    'soupe' : {'poireau': 4, 'pommes de terre': 2},\
    'fondant au chocolat' : gateauChoc,\
    'quatre-quarts' : quatreQuarts}
```

Exercice 1 : Spécifier puis écrire la fonction `recettePossible(frigo,uneRecette)` qui prend en paramètre deux éléments de type `TIngredients` et qui retourne vrai si la recette peut être faite à partir du contenu du réfrigérateur.

```
>recettePossible({'oeufs':3, 'tomates': 4},gateauChoc)
False
```

Exercice 2 : Spécifier puis écrire la fonction `ajouteCourses(frigo,courses)` qui ajoute dans le réfrigérateur les courses qui viennent d'être faites. Les courses sont aussi de type `TIngredients`.

```
>ajouteCourse({'oeufs':3, 'tomates':4},{'oeufs':2, 'melon':1})
{'oeufs':5, 'tomates':4, 'melon':1}
```

Exercice 3 : Spécifier puis écrire la fonction `recettesPossibles(frigo,lesRecettes)` qui retourne la liste des noms des recettes qui peuvent être faites à partir du contenu du réfrigérateur. Chaque recette de la liste retournée doit pouvoir être faite, mais on ne peut pas forcément faire toutes les recettes de la liste.

```
>recettesPossibles({'oeufs':4, 'sucre':350, 'farine':300,
'chocolat':200, 'beurre':250}, lesRecettes)
['fondant au chocolat','quatre-quarts']
```

Exercice 4 : Spécifier puis écrire la fonction `cuisineRecette(frigo,uneRecette)` qui retourne le contenu du réfrigérateur après avoir fait `uneRecette`. Attention, si vous prenez complètement un ingrédient, il ne doit plus apparaître dans le résultat (pas de quantité nulle).

```
>cuisineRecette({'oeufs':4, 'sucre':350, 'farine':300,
'chocolat':200, 'beurre':250}, gateauChoc)
{'sucre':200,'farine':220,'beurre':50}
```

Exercice 5 : Spécifier puis écrire la fonction `cuisineLesRecettes(frigo,lesRecettes)` qui retourne le contenu du réfrigérateur après avoir fait toutes `lesRecettes`.

Exercice 6 : Spécifier puis écrire la fonction `coursesPourRecette(frigo,uneRecette)` qui, cette fois-ci, retourne la liste des courses à faire pour pouvoir faire une recette.

Exercice 7 : Spécifier puis écrire la fonction `toutesLesCourses(frigo,lesRecettes)` qui retourne la liste des courses nécessaires pour faire toutes les recettes de `lesRecettes` (de type `TRecettes`).

Un repas est maintenant représenté par une liste de noms de recettes.

```
# TRepas : liste de noms de recettes
# TRepas : list(str)
```

Exercice 8 : Spécifier puis écrire la fonction `lesIngredients` qui, pour un repas et un livre de recettes, retourne tous les ingrédients (et leur quantité) nécessaires pour faire ce repas.

```
>>> lesIngredients(['soupe',omelette'],lesRecettes)
{ 'oeufs': 4, 'lait (en cl)': 5, 'poireau': 4, 'pommes de terre': 2}
```

On organise un peu mieux les informations, et maintenant les recettes sont associées à des catégories (*recettes végétariennes*, *entrées*, *potages*, *recettes à mijoter*,...). Une recette peut être associée à plusieurs catégories. On représente cette classification par un dictionnaire dont la clé est le nom d'une catégorie et la valeur la liste des noms de recettes qui entrent dans cette catégorie, ce qui correspond au type suivant :

```
# TClassification : dico(str,liste(str))
# L'exemple suivant servira à illustrer les questions suivantes :
>>> laClassification = {'recettes végétariennes' : ['omelette', 'soupe'],\
        'entrées' : ['soupe'],\
        'desserts' : ['quatre-quarts', 'fondant au chocolat'] }
```

Exercice 9 : Spécifier puis écrire la fonction `categoriesDUneRecette` qui, pour une recette et la classification des recettes, retourne la liste des catégories de cette recette.

```
>>> categoriesDUneRecette('soupe',laClassification)
['recettes végétariennes','entrées']
```

Exercice 10 : Spécifier puis écrire la fonction `recettesDUneCategorie` qui, pour une catégorie `categ`, la classification des recettes, et un repas, retourne les recettes du repas qui sont associées à `categ`. Rappel : un repas est représenté par une liste de noms de recettes.

```
>>> recettesDUneCategorie('recettes végétariennes',laClassification,
['escargots de Bourgogne','soupe','carbonnade','tarte au sucre'])
['soupe']
```

Exercice 11 : Spécifier puis écrire la fonction `convient` qui, pour un repas `rep`, une liste de catégories `categs` souhaitées, et des informations sur les catégories des recettes, retourne vrai si `rep` contient au moins une recette pour chaque catégorie de `categs`, faux sinon. Il est possible qu'une même recette corresponde à plusieurs catégories souhaitées, ou que le repas contienne des recettes hors classification.

```
>>> convient(['soupe','carbonnade','tarte au sucre'],
['recettes végétariennes','poissons'],laClassification)
False
>>> convient(['escargots de Bourgogne','soupe','fondant au chocolat'],
['recettes végétariennes','entrées'],laClassification)
True
```