TP n^{o}1 : Programmation de fonctions non récursives

Préambule

1) Pour tester vos fonctions, 3 méthodes
- lancez l'interpréteur `ghci` et tapez vos commandes en ligne
- écrivez vos commandes dans un fichier `.hs`, par exemple `toto.hs`
  - △ si `toto.hs` ne comporte pas de `main` :
    chargez le dans `ghci` par `:load toto.hs`
    et utilisez alors vos commandes.
  - △ si `toto.hs` est un programme (avec un `main`) alors
    soit chargez le par `:load toto.hs` et exécuter le par `:run main`
    soit compilez le par `ghc --make toto.hs` puis exécutez le (`./toto`)

2) Pour commencer à afficher des messages à l'écran vous pouvez utiliser les fonctions suivantes :

```
print $ "variable n = " ++ show(n)
```

où :
- `print` est une fonction d'affichage à l'écran. La syntaxe avec "$" permet de s'affranchir des parenthèses
- `show` est une fonction de transformation d'un type de base en chaine de caractères
- "++" est la concaténation de chaînes de caractères

3) L'instruction `do` permet d'enchaîner plusieurs actions. Par exemple

```
... do
let nb = 5 in
print $ show(nb) ++ " pesetas"  ;
print $ "pour payer l'addition de " ;
tourneegenerale( nb ) ...
```

Questions

Q1 Fonctions à 4 paramètres :

(a) Définir une fonction qui, pour 4 nombres, renvoie `True` si les 4 nombres sont égaux et `False` sinon.

(b) Définir une fonction qui, pour 4 nombres, retourne le plus grand des 4

(c) Définir une fonction qui, pour 4 nombres, retourne le plus éloigné. On suppose que la distance entre deux nombres est la valeur absolue de leur différence

Q2 Dominos : Un domino est une pièce divisée en deux moitiés réversibles comportant chacune une valeur comme dans la FIGURE 1 . Dans la

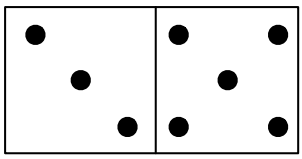

FIGURE 1 – Domino (3-5) ou Domino (5-3)

variante classique du jeu, on forme des chaînes linéaires de dominos en réunissant des moitiés ayant la même valeur.

(a) Ecrire un programme qui détermine s'il est possible de former une chaîne linéaire avec deux dominos donnés. Par exemple
- avec `(2-3) et (1-3)` → possible
- avec `(2-3) et (1-4)` → impossible

(b) Ecrire un programme qui détermine s'il est possible de former une chaîne linéaire avec trois dominos donnés. Par exemple
- avec `(2-3) (2-4) et (1-3)` → possible
- avec `(2-3) (2-4) et (1-2)` → impossible

Q3 Orthographe des nombres
Définir une fonction qui traduit un nombre entre 1 et 100 en sa chaîne littérale en français. Par exemple
- pour le nombre 16 → "seize"
- pour 23 → "vingt-trois"
- pour 72 → "soixante-douze"

On peut utiliser les règles du français suivantes :

- pas de règle particulière pour l'écriture des nombres jusqu'à 16
- la dizaine 80 s'écrit avec un "s" mais pas les chiffres de cette dizaine ("quatre-vingts", "quatre-vingt-un", ...)
- les dizaines s'écrivent en un seul mot à part 70, 80 et 90
- si l'unité est 1 alors toutes les dizaines <= 7 s'écrivent avec "et" sinon avec un "-". (""vingt et un", "trente et un", ..., "soixante et onze" , "quatre-vingt-un", "quatre-vingt-onze")
- si d'une part, la dizaine est 7 ou 9, et que d'autre part, l'unité < 6 alors on applique les règles d'écriture
  — de la dizaine-1
  — et de l'unité+10

Q4 Monnayeur

(a) Faire une fonction qui vérifie si on peut payer une somme s avec un stock de pièces donné.

- une somme s
- un stock de pièces :
  — a de 2 €
  — b de 1 €
  — c de 50 centimes
  — d de 10 centimes

renvoie

- `vrai` si on peut payer la somme s en utilisant le stock de pièces
- `faux` sinon

On peut procéder ainsi pour vérifier si l'on peut payer avec des pièces de 10 cents

i. on regarde si le reste de la somme payée avec des pièces de 10 cents est égale à 0

ii. si oui alors le nombre de pièces de 10 cents nécessaires est
    le reste de la somme payée avec des pièces de 5 cents
    divisée par 10

iii. si le nombre de pièces de 10 cents nécessaires est inférieur ou égal au nombre de pièces de 10 cents en stock alors

iv. on fait des paquets de 5 pièces de 10 cents avec les pièces de 10 cents restantes (pour faire la même valeur qu'une pièce de 50 cents)

v. puis on réitère le même procédé pour voir si on peut payer avec des pièces de 50 cents avec
   - la somme diminuée de la valeur du nombre de pièces de 10 cents nécessaires
   - le nombre de pièces de 50 cents augmentée du nombre de paquets de pièces de 10 cents

vi. Puis on réitère avec les pièces de 1 € et enfin celles de 2 €

(b) Faire une fonction qui, si c'est possible, affiche le nombre de pièces qu'il faut donner pour payer une somme s