POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S6) **Structures de Données 2009/2010**

Évaluation de TP sur Machine - Structures de Données en C

Ce sujet traite du problème de rendu de monnaie. Nous allons automatiser l'algorithme que vous utilisez si vous avez assez de pièces. Pour cela, nous allons utiliser les **listes chaînées**.
Inspiration : sujet d'informatique MP du concours Centrale/Supélec 2002.

Premières fonctions Quelques fonctions (ou procédures !) utiles, pour commencer :

1. Déclarer les types `Cellule` et `Liste`.
2. Écrire une fonction qui imprime tous les éléments d'une liste, dans l'ordre de la liste.
3. Écrire une fonction pour que l'utilisateur puisse entrer une liste *d'éléments positifs ou nuls* par ajout en tête.
4. Écrire une fonction récursive qui retourne le nombre d'occurences du chiffre 0 dans une liste donnée.

Avant de passer à la suite, votre programme principal devra obligatoirement comporter des tests de vos fonctions précédentes.

Les choses sérieuses Un système monétaire est une suite *strictement décroissante* d'entiers positifs, dont le dernier élément est égal à 1. Par exemple, la suite `[5,2,1]` est un système monétaire, mais les suites `[7,5,2]` et `[5,7,1]` n'en sont pas.

5. Écrire une fonction qui teste si une liste donnée est un système monétaire. Tester.
6. Écrire une fonction qui crée une liste qui représente le système monétaire européen (en centimes d'euros), c'est-à-dire la liste `[500,200,100,50,20,10,5,2,1]` à partir d'un fichier contenant ces valeurs en l'ordre inverse (`1 2 5 10 20 50 100 200 500`). Le nom de ce fichier sera fourni en paramètre du `main` (il est interdit d'utiliser les redirections).

Pour rendre un certain montant x avec un système monétaire $[c_1, c_2, \dots c_n]$ donné (en supposant qu'on dispose de suffisamment de pièces de chaque sorte), on essaie de maximiser le nombre de pièces de la sorte c_1, puis on calcule le reste, et on essaie de maximiser les pièces de la sorte c_2, et ainsi de suite. Par exemple, avec le système $[10, 5, 2, 1]$, pour rendre 27 on va rendre 2 pièces de 10, puis 1 pièce de 5, puis 1 pièce de 2, que l'on peut coder sous forme de liste de même taille que le système monétaire : $[2, 1, 1, 0]$. Finalement :

> Pour rendre x, on rend $q = x$ div c_1 pièces de la sorte c_1, puis on rend $x - qc_1$ en utilisant le système $[c_2, \dots c_n]$

7. Écrire une fonction récursive `rendre_monnaie`, qui étant donné un montant et un système monétaire, imprime le nombre et le détail des pièces à rendre (et non les autres), sur l'exemple cela donne :

```
2 piece(s) de 10
1 piece(s) de 5
1 piece(s) de 2
==> Il faut donc 4 piece(s).
```

Remarque : Il est prouvé que l'algorithme ci-dessus, appelé algorithme glouton, minimise le nombre de pièces rendues pour le système européen. Par contre, ce n'est pas le cas pour d'autres systèmes. Avant 1971, le Royaume Uni utilisait le système `[30, 24, 12, 6, 3, 1]`, pour lequel l'algorithme ne rend pas toujours le nombre minimal de pièces.