TP 9 : Des types et un peu d'algorithmique

Antoine Frénoy
frenoy@gmx.com
http://perso.crans.org/frenoy/caml2012.html

Mardi 10 avril 2012

Exercice 1

Un commerçant vend des patates, des carottes et des navets. Les patates sont à 1 euro le kilo, les carottes à 1.30 euro le kilo, et les navets à 1.70 euro le kilo. Les clients se servent eux-même dans des sachets plastiques. Lors de l'encaissement, le prix de chaque sachet est déterminé par son poids (un nombre réel) et le type de légume qu'il contient (un seul type de légume par sachet). Le panier d'un client contient plusieurs sachets (éventuellement plusieurs sachets d'un même type de légume).

1. Définissez un type `legume`.
2. Définissez un type `sachet`.
3. Définissez un type `panier`.
4. Définissez une fonction `prixsachet` prenant en argument un élément de type `sachet` et retournant son prix en euros.
5. Définissez une fonction `prixpanier` prenant en argument un élément de type `panier` et retournant son prix en euros.

Exercice 2

Le même commerçant doit rendre la monnaie à ses clients. Il utilise pour cela des pièces de 1, 2, 5, 10, 20, et 50 centimes, et de 1 et 2 euros.

1. Définissez un type `piece`.
2. Définissez une fonction `rendumonnaie` qui prend en argument un flottant (l'argent à rendre) et qui renvoie une liste de pièces optimale (utilisation du moins de pièces possible).

Exercice 3

On s'intéresse à des expressions logiques booléennes, qui sont composées des constantes "VRAI" et "FAUX", de l'opérateur unaire "NON", des opérateurs binaires "ET", "OU", "OU EXCLUSIF" et "IMPLIQUE".

1. Définissez un type `expression`.
2. Utilisez ce type pour représenter l'expression $(vrai \wedge faux) \vee (faux \Rightarrow (vrai \Rightarrow faux))$.
3. Définissez une fonction récursive `evalue` qui prend un argument de type `expression` et renvoie un élément de type `bool` indiquant si l'expression booléenne est vraie.

Exercice 4

On s'intéresse à un objet mathématique appelé "arbre binaire". Un arbre binaire est un ensemble de données (dans notre cas des entiers) hierarchisées : chaque donnée est stockée dans un noeud, les noeuds sont reliés par des liens de parenté (le noeud le plus haut est appelé la racine, il a des fils qui ont eux même des fils et ainsi de suite). Pour plus d'informations, voir `http://fr.wikipedia.org/wiki/Arbre_binaire`.

Les arbres binaires peuvent être définis récursivement : un arbre binaire est soit l'arbre vide, soit constitué d'un noeud (associé à un entier) avec deux fils (chacun des fils étant lui-même un arbre binaire).

1. Définissez un type `arbre`.
2. Définissez une fonction `taille` prenant en argument un arbre binaire et renvoyant son nombre de noeuds.
3. Définissez une fonction `hauteur` prenant en argument un arbre binaire et renvoyant sa hauteur, c'est à dire la distance entre la racine et son descendant le plus éloigné.
4. Définissez une fonction `appartient` prenant en arguments un arbre binaire et un entier et indiquant si l'entier apparait dans l'arbre binaire.

Exercice 5

Un graphe est un objet mathématique qui représente des points (appelés "sommets") reliés entre eux (un lien entre deux sommets est appelé "arête"). Les sommets de nos graphes seront numérotés par des entiers. Une arête sera donc représentée par deux entiers (le sommet de départ et celui d'arrivée). Nos graphes seront non orientés, c'est à dire qu'une arête qui va du sommet 1 au sommet 2 est la même chose qu'une arête qui va du sommet 2 au sommet 1.

1. Définissez un type `sommet`, un type `arete` et un type `graphe`.
2. Avec le type `graphe` qu'on vient de définir, il est possible d'avoir des arêtes faisant référence à des sommets inexistants. Écrivez une fonction `check_graphe` prenant en argument un graphe et vérifiant que ce n'est pas le cas.
3. Définissez une fonction `adjacents` qui prend en arguments un graphe et un sommet de ce graphe et renvoie la liste des sommets adjacents au sommet donné en argument (c'est à dire reliés à ce sommet par une arête).
4. (difficile) Définissez une fonction `connexe` qui prend en argument un graphe et renvoie la liste de ses composantes connexes.