Programmation Fonctionnelle : TD2

Université de Tours

`Département informatique de Blois`

Expressions booléennes, prédicats, conditions, structures

*
* *

Appropriation du cours

Revenons sur le cours 2, qui se trouve sous Celene : testez en Ocaml les exemples ou exercices suivants :

- La fonction `divisiblePar` (définition/code et application à des arguments).
- La fonction `bissextile` (définition/code et application à des arguments).
- La fonction `secondeSuivante` (définition/code et application à des arguments), ***dans les 2 versions***, et en testant aussi une version dans laquelle vous mettez les motifs (pattern) dans l'ordre inverse pour le switch, afin de constater que c'est bien différent.
- Une fonction `addFraction` qui fait la somme de 2 fractions (définition et application à des arguments) avec définition d'un nouveau type `fraction` ***et*** sans définition de nouveau type, en utilisant un couple d'entiers $(n_1, n_2) \in$ `int * int`.

Problème 1

On s'intéresse ici à la modélisation de relations d'ordre élémentaires pour les fractions.

1. Sur l'opérateur $=$:
   (a) Écrire la spécification d'une fonction `egalite1` de deux fractions à l'aide du type `fraction` défini précédemment.
   (b) Écrire la spécification d'une fonction `egalite2` avec la notation sous forme de couple d'entiers. Quelle différence notez-vous ?
   (c) Écrire le code des fonctions `egalite1` et `egalite2`.
2. Sur l'opérateur $\leq$:
   (a) Écrire la spécification et le code d'une fonction `plusGrand` qui prend en paramètre deux fractions f_1 et f_2 et qui retourne Vrai si et seulement si $f_1 < f_2$.
   (b) Écrire la spécification et le code d'une fonction `plusGrandEgal` qui prend en paramètre deux fractions f_1 et f_2 et qui retourne Vrai si et seulement si $f_1 \leq f_2$.
3. Sur l'opérateur $\sum$:
   (a) Écrire une liste `list_frac` de 3 fractions quelconques à l'aide du type `fraction`.
   (b) Écrire la spécification et le code d'une fonction `sommeFraction` qui retourne la somme, sous forme de fraction, de toutes les fractions définies dans une liste.

Problème 2

1. Définir un type `couleur` où une couleur $c \in \{\text{Rouge}, \text{Bleu}, \text{Jaune}, \text{Blanc}, \text{Noir}\}$

2. Définir un type `peinture` qui peut-être une *Couleur* (`couleur`), un *Numéro* défini par un entier (`int`) ou un *Mélange* qui est un couple de deux peintures (`peinture * peinture`).

   Exemple:

   ```
   let p1 = Couleur Bleu;;
   let p2 = Melange (p1, Numero 0);;
   let p3 = Melange (p2, Couleur Rouge);;
   ```

3. On souhaite définir un nouveau type `potPeinture` comprenant une *teinte* (`peinture`), une *contenance* en litres (`float`) et un *prix* en euros (`float`).

   Exemple:

   ```
   let pp1 = {teinte = p1; contenance = 1.5; prix = 20.5};;
   let pp2 = {teinte = p3; contenance = 2.0; prix = 25.3};;
   let pp3 = {teinte = p2; contenance = 2.25; prix = 19.0};;
   ```

4. Écrire la spécification et le code d'une fonction `scalePP` qui prend en paramètre un pot de peinture et une contenance en litre et renvoie le prix du pot de peinture pour cette contenance.

   Exemple. `scalePP pp1 2.0` $\Rightarrow$ `27.33`

5. Relation d'ordre sur des pots de peinture :

   (a) Écrire la spécification et le code d'une fonction `plusGrandContenance` qui retourne vrai si et seulement si la contenance de pot de peinture pp_1 est plus petite ou egale que la contenance d'un pot de peinture pp_2.

   Exemple. `plusGrandContenance pp2 pp3` $\Rightarrow$ `True`

   (b) Écrire la spécification et le code d'une fonction `plusGrandPrix` similaire à celle de la question 5. (a) mais pour le prix.

   Exemple. `plusGrandContenance pp2 pp3` $\Rightarrow$ `False`

6. Sur le tri des pots de peinture :

   (a) Écrire la spécification et le code d'une fonction `sortPeinturePrix` qui prend en entrée trois pots de peinture et retourne ces pots (dans un triplet), ordonnées par ordre croissant de prix.

   (b) Généraliser la fonction pour trier maintenant selon la contenance.

   (c) Comment faire pour trier un 4-uplet ? Un 5-uplet ? Un n-uplet ? Combien d'arrangements (ici, façons de ranger les pots de peinture) total dénombre t-on pour n éléments ?

   (d) En programmation fonctionnelle, quelle structure semble plus adaptée pour des opérations telles que les tris ? Pourquoi ?