Programmation fonctionnelle (HAI102I)

Examen de session 2 Juin 2022

Licence Informatique
Département Informatique
Faculté des Sciences de Montpellier
Université de Montpellier

Le seul document autorisé est une feuille A4 recto-verso. L'examen dure 2h. Le sujet comporte 2 pages et il y a 3 exercices.
Dans ce qui suit, vous avez le droit d'utiliser les fonctions des questions précédentes même si vous n'êtes pas parvenu à les écrire. Dès qu'une fonction prendra plusieurs arguments (au moins deux), vous écrirez la fonction sous sa forme curryfiée.

Exercice 1 (6 Pts)

1. Écrire la fonction `prodFonc`, qui étant données une fonction `f` et une liste `[e1; ...; en]`, calcule `(f e1) * ... * (f en)` en utilisant uniquement les accesseurs sur les listes `List.hd` et `List.tl`. Quelle est la signature de cette fonction ?
2. Réécrire la fonction `prodFonc` en utilisant cette fois le filtrage sur les listes (utilisation de `match` ou de `function`).
3. Réécrire une troisième version de la fonction `prodFonc` mais en utilisant uniquement l'itérateur `List.fold_right`.
   On rappelle que `(List.fold_right f [a1; ...; an] init)` est équivalent à l'expression `(f a1 (f a2 (... (f an init) ...)))`.

Exercice 2 (7 Pts)

Les nombres de Mersenne sont les nombres entiers qui s'écrivent $2^n - 1$, où n un entier naturel non nul. On notera M_n le nombre de Mersenne de rang n égal à $2^n - 1$. Les premiers nombres de Mersenne sont $M_1 = 1$, $M_2 = 3$, $M_3 = 7$, $M_4 = 15$ etc.

1. Écrire la fonction `puis2`, qui étant donné un entier n, calcule l'entier 2^n.
   Vous utiliserez pour cela la fonction `x ** y`, qui étant donnés deux nombres flottants `x` et `y` calcule le nombre flottant $\texttt{x}^{\texttt{y}}$. Il vous faudra également utiliser les fonctions de conversion `float_of_int` (de `int` vers `float`) et `int_of_float` (de `float` vers `int`).
2. Écrire la fonction `nbMersenne`, qui étant donné un entier strictement positif n, calcule M_n, le nombre de Mersenne de rang n. Par exemple `(nbMersenne 3)` vaut 7.
3. Écrire la fonction `estMersenne`, qui étant donné un entier `p`, teste si `p` est un nombre de Mersenne, c'est-à-dire s'il existe un entier naturel non nul n tel que $\texttt{p} = M_n$.
   Pour ce faire, vous écrirez, au préalable, la fonction `estMersenneRec`, qui étant donnés deux entiers `p` et `n`, teste s'il existe un nombre de Mersenne égal à `p` de rang supérieur ou égal à `n`. Par exemple, `(estMersenneRec 15 3)` vaut `true` car 15 est un nombre de

Mersenne de rang supérieur à 3, alors que `(estMersenneRec 15 5)` s'évalue en `false` car $M_5 = 31 > 15$.

4. Écrire la fonction `listeMersenne`, qui étant donné un entier `n`, rend la liste composée des `n` premiers nombres de Mersenne (l'ordre de cette liste est libre et peut être croissant ou décroissant sur les nombres de Mersenne).

Exercice 3 (8 Pts)

On définit le type somme `chaine`, représentant les chaînes de caractères, de la façon suivante :

```
type chaine =
  | Vide
  | Cons of char * chaine
```

1. Écrire la fonction `stringVersChaine`, qui étant donnée une chaîne de caractères `s` de type `string` (type prédéfini d'OCaml), rend la valeur de type `chaine` correspondant à `s`.
   Exemples d'évaluation :
   - `(stringVersChaine "")` s'évalue en `Vide`.
   - `(stringVersChaine "L1")` s'évalue en `Cons ('L', Cons ('1', Vide))`.

   Pour ce faire, vous aurez besoin des 3 fonctions suivantes sur le type `string` :
   - `(String.length s)` rend la taille de la chaîne de caractères `s`.
   - `(String.get s i)` rend le caractère à l'indice `i` dans la chaîne de caractères `s`.
   - `(String.sub s pos len)` rend la sous-chaîne de la chaîne de caractères `s` qui commence à la position `pos` et a la longueur `len`.
2. Écrire la fonction `chaineVersString`, qui étant donnée une chaîne de caractères `ch` de type `chaine`, rend la chaîne de caractères de type `string` correspondant à `ch`.
   Exemples d'évaluation :
   - `(chaineVersString Vide)` s'évalue en `""`.
   - `(chaineVersString (Cons ('L', Cons ('1', Vide))))` s'évalue en `"L1"`.

   Pour ce faire, vous aurez besoin de la fonction `Char.escaped`, qui étant donné un caractère `c`, rend la chaîne de caractères (de type `string`) contenant `c`. Par exemple, `(Char.escaped 'L')` s'évalue en `"L"`.
3. Écrire la fonction `longChaine`, qui étant donnée une chaîne de caractères `ch` de type `chaine`, calcule le nombre de caractères de `ch`.
   Exemples d'évaluation :
   - `(longChaine Vide)` vaut `0`.
   - `(longChaine (Cons ('L', Cons ('1', Vide))))` vaut `2`.
4. Écrire la fonction `iemeCar`, qui étant donnés une chaîne de caractères `ch` de type `chaine` et un entier `i`, rend le caractère d'indice `i` dans la chaîne `ch` (le premier caractère d'une chaîne non vide est d'indice 0). S'il n'y a pas de caractère d'indice `i` dans `ch`, vous devrez lever l'exception `Invalid_argument` avec le message de votre choix.
   Exemples d'évaluation :
   - `(iemeCar (Cons ('L', Cons ('1', Vide))) 1)` s'évalue en `'1'`.
   - `(iemeCar (Cons ('L', Cons ('1', Vide))) 4)` lève une exception.