Prise en main

Manipulation des types

Executer `1+1;;`. A quoi servent les `;;` ?
Stocker la valeur 1073741823 dans une variable `a`. Ajouter `1` à `a` sans changer `a`. De même, multiplier `a` par 2 sans changer `a`.
Calculer le quotient et le reste de la division entière de 5 par 2.
Executer `1. + 1.;;` observer, expliquer, corriger.
Executer `let b = float_of_int a;;`, puis ajouter `1` à `b` sans changer `b`. De même, multiplier `b` par 2 sans changer `b`.
Calculer $\log_2 b$ (la fonction `log` de Caml calcule le logarithme népérien).
Tester la fonction `int_of_float` sur différentes valeurs, puis sur `2.**63.`.
Exécuter `a=1073741823 ;;`. A quoi correspond `=` en Caml ?
Executer `not (1=2);; "zo" = "zé" ;; (2 < 3) & (3*4 = 2*6);;`
`true || false ;;` et `1. < 2 ;;`.
Caml distingue les caractères et les chaines de caractères. Executer `'z';;` et `"z";;` et observer la différence.
Stocker la chaine `premier tp` dans une variable `s` puis exécuter `s.[0];;`.
Modifier la chaine pour obtenir `Premier TP`.

Fonctions

Ecrire une fonction `incremente` qui à x associe $x+1$ et prévoir son type.
Ecrire deux fonctions `carre_int` et `carre_float` qui mettent au carré respectivement un entier et un flottant passé en paramètre.
Ecrire une fonction `somme` prenant en paramètre un *couple* d'entiers et qui renvoie leur somme.
Ecrire une fonction `echange` échangeant l'ordre des éléments dans un couple.
Ecrire une fonction `first` qui renvoie le premier élément d'un couple.
Ecrire une fonction `fourth` qui renvoie le quatrième élément d'un quadruplet.
Ecrire une fonction `somme2` *curryfiée* faisant la somme de deux entiers.
Ecrire une fonction `couple` curryfiée qui prend deux arguments x et y et renvoie le couple (x, y).

Définitions globales et locales

Prévoir la réponse de l'interprète de commande des commandes suivantes, puis vérifier.
Remarque : l'indentation est facultative et n'est utilisée ici que pour la lisibilité de l'énoncé.

```
let x = 5 in
   let y = 1 - x in
      x + y;;

let x = 5 and y = 1 - x in
   x + y;;

let a = 1 in
   let a = a + 1 in
      let b = a + 1 in a + b;;
```

```
let a = 1 in
   let a = a + 1 and b = a + 1 in a + b;;

let a = 2 ;;
let f x = a * x ;;
let a = 3 in f 1 ;;

let a = 3 and f x = a * x;;
f 1 ;;

let a =
   let a = 3 and b = 2 in
      let a = a + b and b = a - b in
         a - b;;

let b = 2 in a - b * b;;
```

Typage

1. Déterminer sans utiliser l'interprète de commandes le type des expressions suivantes :

a) `fun f x y -> f x y ;;`
b) `fun f g x -> g (f x) ;;`
c) `fun f g x -> (f x) + (g x);;`
d) `fun x y z -> (x y) z ;;`
e) `fun x y z -> x (y z) ;;`
f) `fun x y z -> x y z ;;`
g) `fun x y z -> x (y z x) ;;`
h) `fun x y z -> (x y) (z x);;`

2. Ecrire une fonction Caml pour chacun des types suivants :

a) `int -> int`
b) `int * int -> int`
c) `int -> int * int`
d) `(int -> int) -> int`

Définitions de fonctions

a) L'opérateur des différences finies Δ associe à toute suite $(u_n)_{n\in\mathbb{N}}$ la suite $(u_{n+1} - u_n)_{n\in\mathbb{N}}$. Écrire une fonction delta qui réalise cette transformation. On commencera par en donner son type.

b) Ecrire une fonction `curry` qui transforme une fonction à deux variables non curryfiée en une fonction curryfiée, puis une fonction `uncurry` qui réalise la transformation inverse.

c) Ecrire une fonction `diviseur` testant si un premier entier en divise un second.

d) Ecrire une fonction `ndigits` donnant le nombre de chiffres d'un entier.

e) Ecrire la fonction cosinus hyperbolique et sa réciproque.

Devoir Surveillé numéro 1
D'APRÈS LE CONCOURS E3A (2002-2013)
Mardi 7 mai 2019
Durée : 2 heures

Consignes générales
- Les programmes doivent être systématiquement précédés d'une brève explication de leur fonctionnement (sauf éventuellement ceux de moins de 4 lignes).
- Vous êtes libres de définir autant de fonctions intermédiaires que vous le souhaitez pour répondre à une question.
- Les conversions de type entre listes et tableaux ou entre entiers et chaines de caractères ne sont pas autorisées. Tout mélange de syntaxe entre ces structures de données sera automatiquement sanctionné par un 0 à la question.
- Les exercices sont de difficultés comparables et ne sont présentés dans aucun ordre particulier.

1. Nombres de Hamming

La suite de Hamming est la suite croissante des entiers n'admettant que 2, 3 ou 5 comme éventuels diviseurs premiers. Cette suite commence donc par la séquence

1 2 3 4 5 6 8 9 10 12 15 16

Etant donné $N \in \mathbb{N}$, on veut construire un tableau `T` de taille N contenant les N premières valeurs de cette suite. On utilise pour cela le principe suivant :
1. Initialement, `T` est rempli de 0, à l'exception de sa première case (d'indice 0) qui contient un 1.
2. Le tableau étant en cours de remplissage, l'élément suivant à ajouter est à choisir parmi les nombres suivants : $2T[i_2]$, $3T[i_3]$, $5T[i_5]$ où i_2 (resp. i_3 et i_5) est l'indice du plus petit élément du tableau n'ayant pas son double (resp. triple, quintuple) dans le tableau.

A titre d'exemple, si `T` est dans l'état

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$T[k]$	1	2	3	4	5	6	8	9	10	12	15	16				

Alors, on a $i_5 = 3$, $i_3 = 5$ et $i_2 = 7$ et l'élément suivant est à choisir parmi $5T[3] = 20$, $3T[i_3] = 18$ et $2T[i_2] = 18$. On rajoute donc 18 en case 12, et il faut alors incrémenter i_2 et i_3.
En vous basant sur le principe ci-dessus, écrire un programme en Caml prenant en argument l'entier N et renvoyant un tableau contenant les N premiers nombres de Hamming. Donner la complexité de la fonction.

2. Sous-suite de trois nombres de somme maximale

Soit `L` une liste d'entiers contenant au moins 3 valeurs. On considère que les entiers sont indexés à partir de 0. On souhaite écrire une fonction qui renvoie l'indice du premier élément d'un triplet de somme maximale dans la liste. Par exemple pour la liste `[1;5;7;8;6;2;7;9;4]`, la fonction doit retourner 2, indice du premier élément du triplet $(7, 8, 6)$ de somme 21.
La fonction finale fera appel à une fonction auxiliaire récursive qui renvoie l'indice et la valeur de la somme. Pour l'écrire, on peut se poser les questions suivantes : dans quel cas n'y a-t-il pas de solution ? Une solution immédiate ?

3. Suites oscillantes

Soit E un ensemble fini et $f : E \longrightarrow E$ une fonction. On définit une suite $(u_n)_{n\in\mathbb{N}}$ en prenant

$$u_0 \in E \text{ et } \forall n \in \mathbb{N}, u_{n+1} = f(u_n)$$

1. Justifier que $(u_n)_{n\in\mathbb{N}}$ est nécessairement périodique à partir d'un certain rang, c'est-à-dire qu'il existe $p \in \mathbb{N}$ et $r \in \mathbb{N}^*$ tels que $u_{n+r} = u_n$ pour tout $n \geq p$.
2. Ecrire un programme qui prend en argument la fonction f et la valeur de u_0 et renvoie les plus petits entiers p et r pour lesquels la propriété précédente est vérifiée. Donner la complexité de la fonction. On pourra pour cela utiliser de la fonction `List.mem : 'a -> 'a list -> bool` de complexité linéaire qui indique si un élément passé en argument appartient ou non à une liste donnée.

4. Nombres parfaits

Soit n un entier naturel. On dit que n est un nombre parfait si $2n$ est la somme des entiers naturels diviseurs de n. Par exemple, l'entier 6 est parfait puisque $2 \times 6 = 12 = 6 + 3 + 2 + 1$.
Ecrire une fonction calculant la somme des diviseurs d'un entier. Ecrire ensuite un programme qui détermine la liste des nombres parfaits n tels que $n \leq 9999$.

5. Suites de Fibonacci

Une suite de Fibonacci généralisée est définie par $u_0 = a, u_1 = b, u_n = u_{n-1} + u_{n-2}$ pour $n \geq 2$. On se propose de comparer plusieurs méthodes calculant le n-ième élément de la suite.
1. Méthode récursive simple.
Ecrire la fonction `fibo1` prenant en arguments n, a, b des entiers et qui renvoie le n-ième élément de la suite de Fibonacci initialisée par a et b en utilisant la définition mathématique récursive. Donner la formule de récurrence vérifiée par la complexité de la fonction.
2. Méthode itérative
Pour la méthode itérative, on utilise une boucle avec trois variables locales : une variable contenant l'avant-dernier élément de la suite, une variable contenant le dernier élément, une variable contenant le nouvel élément.
Ecrire une fonction `fibo2` prenant en argument n, a, b des entiers et qui renvoie le n-ième élément de la suite de Fibonacci initialisée par a et b en utilisant la méthode itérative. Donner la complexité de la fonction.
3. Méthode récursive par matrices.
a) Montrer mathématiquement l'égalité suivante

$$\begin{pmatrix} u_n \\ u_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} a \\ b \end{pmatrix} \qquad (\star)$$

b) Ecrire une fonction de mutiplication de deux matrices carrées 2×2. Pour représenter les matrices on choisit d'utiliser des quadruplets d'entiers, tels que (a, b, c, d) représente la matrice $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$.
c) En utilisant le principe d'exponentiation rapide, écrire une fonction d'élévation à la puissance n d'une matrice carrée 2×2.
d) Ecrire une fonction `fibo3` prenant en argument n, a, b des entiers et qui renvoie le n-ième élément de la suite de Fibonacci initialisée par a et b en utilisant la formule $(\star)$. Quelle est la complexité de la fonction ?