Option Informatique

TP n°1 : des fonctions récursives Caml

Important : gardez une copie de votre code. L'extension d'un fichier Caml est `.ml`

Note : même lorsque ce n'est pas demandé, écrire des fonctions auxiliaires n'est jamais une mauvaise idée.

Exercice 1. *Exemples à dérouler à la main et tester.*

On propose les fonctions récursives suivantes. Pour chacune, on va l'appeler sur l'argument 3.
Anticiper le résultat, puis tester afin de vérifier. Et si le résultat diffère de celui attendu, expliquer.

```
let rec compte1 n =
    print_int n;
    compte1 n+1;;
compte1 3;;
```

```
let rec compte2 n =
    if n>0 then
    begin
        print_int n;
        compte2 (n-1)
    end;;
compte2 3;;
```

```
let rec compte3 n =
    if n>0 then
    begin
        compte3 (n-1);
        print_int n
    end;;
```

```
let rec compte4 n =
    if n>0 then
    begin
        print_int n;
        compte (n-1);
        print_int n
    end;;
```

```
let rec compte5 n =
    if n>0 then
    begin
        compte5 (n-1);
        print_int n;
        compte5 (n-1)
    end;;
```

Combien d'entiers l'appel `compte5 10` écrira-t-il ?

Exercice 2. *Binomiaux.*

On rappelle que les coefficients binomiaux $\binom{n}{k}$ vérifient les propriétés suivantes :

- $\forall n \in \mathbb{N},\ \binom{n}{0} = 1$;
- $\forall k > 0,\ \binom{0}{k} = 0$;
- $\forall n > 0, \forall k > 0,\ \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

Écrire une fonction récursive permettant de calculer les binomiaux à l'aide de ces relations.

Exercice 3. Nombres de Hamming

Un nombre de Hamming est un entier qui ne comporte que des 2, 3 et 5 dans sa décomposition en facteurs premiers. Écrire une fonction `hamming : int -> bool` qui détermine si un entier donné est ou pas un nombre de Hamming.

Exercice 4. Iterées d'une fonction

Écrire une fonction `itere : ('a -> 'a) -> int -> 'a -> 'a` prenant comme argument une fonction f et un entier n et retournant la fonction $\underbrace{f \circ f \circ \cdots \circ f}_{n \text{ fois}}$.

Exemple : d'après le cours de maths, `itere sin n 1.` *doit être de plus en plus proche de* $0.$ *lorsque* n *augmente.*

Exercice 5. *Dichotomie pour le TVI.*

Écrire une fonction récursive `dichotomie : (float -> float) -> float -> float -> float -> float` qui prend comme argument une fonction f `: float -> float`, et trois flottants a, b et ε tels qu'on ait $a < b$ et $f(a)f(b) < 0$, et retourne une approximation à ε près d'une racine de f calculée par dichotomie (on ne demande pas de vérifier qu'on a bien $a < b$ et $f(a)f(b) < 0$).

Par exemple `dichotomie sin 1. 4. 0.001` doit donner une approximation de π à 10^{-3} près.

Remarque : on pourrait utiliser des boucles mais restons dans l'esprit du TP.

Exercice 6. *Algorithme d'Euclide.*

On rappelle que l'algorithme d'Euclide est basé sur les deux propriétés suivantes :

- $a \wedge b = b \wedge (a \bmod b)$ où $(a \bmod b)$ désigne le reste dans la division euclidienne de a par b ;
- $d \wedge 0 = d$ pour $d \geqslant 0$.

1. Écrire une fonction **récursive** `pgcd : int→int→int` permettant de calculer le pgcd de deux entiers positifs à l'aide de l'algorithme d'Euclide.
2. *Question optionnelle :* Si vous avez fait tout ce qui précède (wow), modifier la fonction précédente pour qu'elle retourne un triplet (d, u, v) tel que $d = a \wedge b$ et $au + bv = d$.

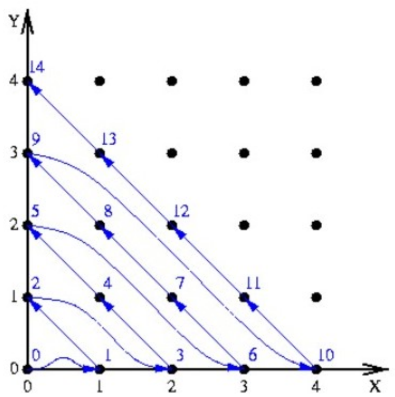
   Indication : Si $a = bq + r$ et si on a trouvé (d, u, v) tels que $d = a \wedge b = b \wedge r$ et $d = bu + rv$, alors $d = bu + (a - bq)v = av + b(u - qv)$.

Exercice 7. *Exercice optionnel :* Test de primalité.

1. Écrire une fonction **récursive** `aux : int → int → bool` telle que : `aux n d` renvoie `true` si et seulement si n n'admet pas de diviseur k de même parité que d et tel que $d \leqslant k < n$.
2. En déduire une fonction `est_premier : int → bool` (non récursive, mais qui fait appel à la fonction `aux`) qui teste si un entier donné en argument est premier.

Exercice 8. *Exercice optionnel :* Bijection $\mathbb{N}^2 \simeq \mathbb{N}$.

On numérote chaque point du plan de coordonnées $(x, y) \in \mathbb{N}^2$ par le procédé suggéré sur la figure ci-dessous :



1. Écrire une fonction récursive `numero : int → int → int` telle que `numero x y` renvoie le numéro du point de coordonnées (x, y).
2. On peut trouver une formule explicite pour `numero x y`. Sauriez-vous le faire ?