TRAVAUX PRATIQUES V

Récursivité en OCaml

Les codes doivent être compilés avec `ocamlopt` et testés !

A Révisions du TP précédent

1. Écrire une fonction `better_print_int` qui affiche un entier (pris en argument) puis un retour à la ligne. Vous n'utiliserez pas `Printf.printf` .

B Pour débuter

Définition 1.
Pour qu'un identifiant puisse être défini à l'aide de lui-même, on ajoute le mot-clef rec :

```
let rec <identifiant> = <expression qui peut contenir l'identifiant>
```

Voici par exemple deux fonctions récursives :

```
let rec demo = fun n ->
  if n > 0 then demo (n-1) + 1 else 0

let rec double = fun n ->
  if n > 0 then double (n-1) + 2 else 0
```

1. Que font les deux fonctions `demo` et `double` ?
2. Écrire une fonction récursive `factorielle` qui prend en argument un entier `n` et calcule sa factorielle $n!$.
   On précise que $n! = 1.2.3....(n-1).n$, et que cette fonction vérifie :

$$n! = \begin{cases} n.(n-1) & \text{si } n > 0 \\ 1 & \text{sinon} \end{cases}$$

3. **a.** Écrire une fonction récursive `ligne_etoiles : int -> unit` qui prend en argument un entier n, s'évalue à unit et a pour effet de bord d'afficher n caractères `*` . Vous pourrez réduire l'instance « afficher n étoiles » à l'instance « afficher n-1 étoiles ».
   b. Écrire une fonction récursive `triangle_bas : int -> unit` qui affiche un triangle de base n pointe vers le bas comme ceci pour n=4 :

```
* * * *
* * *
* *
*
```

   c. Écrire une fonction récursive `triangle_haut : int -> unit` qui comme effet secondaire affiche un triangle de base n pointe vers le haut comme ceci pour n=4 :

```
*
* *
```

```
* * *
* * * *
```

d. (Bonus, plus difficile) Écrire une fonction récursive `carre_creux : int -> unit` qui a pour effet secondaire d'afficher les contours d'un carré `n` × `n`. Voici un exemple pour `n=4` :

```
* * * *
*     *
*     *
* * * *
```

4. Écrire une fonction récursive `somme_f : (int->int) -> int -> int` qui prend en argument une fonction `f : int->int` et un entier positif `n` et qui calcule $\sum_{i=0}^{n} f(i)$.

C Un peu de maths

C.1 Maths du supérieur

5. **a.** Les nombres de Catalan forment une suite d'entiers naturels définis par :

$$C_0 = 1, \quad \forall n \geq 1, C_{n+1} = \frac{2(2n+1)}{n+2} C_n$$

Écrire une fonction `catalan` qui prend en argument un entier naturel n et qui calcule C_n.

6. **a.** La suite de Collatz associée à un $N \in \mathbb{N}$ est définie de la façon suivante :

$$u_0 = N \quad \text{et} \quad \forall n \in \mathbb{N}, u_{n+1} = \begin{cases} \frac{u_n}{2} \text{ si } u_n \text{ est pair} \\ 3 * u_n + 1 \text{ sinon} \end{cases}$$

On appelle *temps de vol* le plus petit entier k tel que $u_k = 0$ (il dépend donc de N car ildépend de u_0). Écrire une fonction `temps_de_vol : int -> int` qui prend en argument N et renvoie son temps de vol.

b. (*Problème ouvert de recherche*) Prouver que votre fonction termine.

C.2 Maths de primaire

Les mathématiques du supérieur, c'est bien. Mais l'école primaire, c'est mieux [1] !

7. **a.** Écrire une fonction `prod : int -> int -> int` qui prend en argument deux entiers positifs `x` et `y` et calcule `x*y` sans utiliser `*` .

b. Écrire une fonction `div : int -> int -> int` qui prend en argument deux entiers positifs `x` et `y` et renvoie `x/y` (division entière) sans utiliser `/` ni `mod` .

c. Écrire une fonction `reste : int -> int -> int` qui prend en argument deux entiers positifs `x` et `y` et renvoie `x mod y` (le reste de `x` divisé par `y`) sans utiliser `/` ni `mod` .

D Pour occuper les plus rapides

Au choix :

8. Avancez sur France-IOI ou Prologin ou un autre projet informatique personnel.

9. Allez lire la définition de la récursivité terminale dans le cours sur la complexité. Réécrivez les fonctions de ce TP pour qu'elles soient récursives terminales et ainsi optimiser leur complexité spatiale.

1. Et c'est un peu plus de l'informatique.