1 Calcul de puissances

⋙ **Question 1** *Écrire une fonction* carre : int -> int *qui calcule le carré d'un entier.*

Fonction récursives : Caml étant un langage fonctionnel, on définit très souvent des fonctions de manière récursive. Une fonction récursive est une fonction qui s'appelle elle-même. Par exemple, voici trois versions différentes de la fonction qui calcule de manière récursive la factorielle :

```
let rec fact n =
  if n = 0
    then 1
    else n*(fact (n-1))
;;

let rec fact =
  fun
      0 -> 1
    | n -> n*(fact (n-1))
;;

let rec fact n =
  match n with
      0 -> 1
    | n -> n*(fact (n-1))
;;
```

⋙ **Question 2** *Écrire une fonction récursive* puissance : int -> int -> int *pour calculer k à la puissance n.*

⋙ **Question 3** *Imaginez une façon plus rapide d'implémenter la fonction* puissance.

2 La suite de Fibonacci

La suite de Fibonacci est la suite $(u_n)_{n\in\mathbb{N}}$ définie par
$\begin{cases} u_0 = 0 \text{ et } u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \end{cases}$

⋙ **Question 4** *Calculez à la main les valeurs de u_2, u_3, u_4, u_5 et u_{10}.*

⋙ **Question 5** *Écrire une fonction récursive* fibo : int -> int *qui calcule la valeur de u_n.*

⋙ **Question 6** *Calculer en fonction de n combien d'appels seront faits à la fonction* fibo.

⋙ **Question 7** *Écrire une fonction récursive* fibo2 : int -> int * int *qui renvoie le tuple (u_n, u_{n-1}).*

⋙ **Question 8** *Complexité de cette fonction ?*

Un corrigé

▶ Question 1

```
let carre n =
  n*n
;;
```

▶ Question 2

```
let rec puissance k n =
  match n with
      0 -> 1
    | n -> k * (puissance k (n-1))
;;
```

▶ Question 3

```
let rec puissance k =
  fun 0 -> 1
  | n ->
    let temp = (puissance k (n / 2)) in
      if n mod 2 = 0
      then carre temp
      else (carre temp) * k
;;
```

▶ Question 4

n	2	3	4	5	6	7	8	9	10
u_n	1	2	3	5	8	13	21	34	55

▶ Question 5

```
let rec fibo =
fun 0 -> 0
  | 1 -> 1
  | n -> (fibo (n-1)) + (fibo (n-2))
;;
```

▶ Question 6

On montre par récurrence que la complexité de cette fonction est $O(2^n)$.

▶ Question 7

```
let rec fibo2 =
fun 1 -> (0,1)
  | n -> let a,b = fibo2 (n-1) in (b, a+b)
;;
```

▶ Question 8

Complexité : $O(n)$.