Question de cours
Rappeler le principe de récurrence.

Exercice 1 : Types en OCaml
Donnez le type (s'il existe) des fonctions OCaml suivantes :

```
let f x = x + x

let g t = t +. (f t)

let h x y z =
   let aux x = x*.x in (aux y, x+z)
```

Exercice 2 : La fonction d'Ackerman-Péter
La fonction d'Ackerman-Péter est définie récursivement par :

$$A(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ A(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ A(m-1, A(m,n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

1. Combien d'appels récursifs sont nécessaires pour calculer A(1,2) ?
2. Ecrire la fonction d'Ackerman en OCaml et donner son type.
3. Combien vaut $A(1,n)$? Montrer par récurrence que $\forall n \geq 0, A(2,n) = 2n+3$

Exercice 3 : Comptage dans une liste
1. Ecrire une fonction de type `('a * 'b) list -> 'a -> bool` qui renvoie vraie si l'entier passé en argument apparait comme premier élément d'un couple de la liste.
2. Ecrire une fonction de type `int list -> int` qui renvoie le plus grand entier de la liste passée en argument.
3. Ecrire une fonction de type `int list -> int` qui renvoie le plus petit entier de la liste passée en argument.
4. Ecrire une fonction de type `int list -> int -> int` qui, pour une liste d'entier l et un entier n passés en argument, renvoie le nombre d'occurrences de n dans l.
5. Ecrire une fonction de type `int list -> (int * int) list` qui pour une liste de nombres renvoie une liste de couples (entier, nombre d'occurrences)

Question de cours
Qu'est-ce que la pile d'appel récursive ?

Exercice 1 : Récurrence
Montrer que la propriété suivante est vraie pour tout $n \in \mathbb{N}$:

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Exercice 2 : Évaluateur d'expression arithmétique
On souhaite pouvoir stocker des expressions arithmétiques simples dans un type sommes. On donne dans un premier temps le type `expr1` :

```
type expr1 =
  | Entier of int
  | Addition of int * int
```

1. Comment écrire 3+4 avec le type `expr1` ? Peut-on écrire 3+4+5 ?

On donne ensuite le type `expr2` :

```
type expr2 =
  Entier of int
| Addition of expr2 * expr2
```

2. Ecrire la fonction `eval : expr2 -> int` qui à partir d'une expression donne son résultat.
3. Ajouter un moyen d'écrire des multiplications et modifier la fonction `eval` en conséquence.

Exercice 3 : Filtrer des listes
1. Ecrire une fonction de type `int -> bool` qui renvoie vraie si l'entier passé en argument est pair.
2. Ecrire une fonction de type `int -> int list` qui renvoie une liste contenant les entiers de 1 à n inclus, où n est le premier argument.
3. Ecrire une fonction de type `int -> int list` qui renvoie une liste contenant les entiers pairs de 1 à n inclus, où n est le premier argument.
4. On souhaite généraliser le filtrage d'une liste précédent selon n'importe quel prédicat. Ecrire une fonction `filter` de type `('a -> bool) -> 'a list -> 'a list` qui renvoie la liste filtrée selon le prédicat.
5. Réécrire la fonction **3.** à l'aide de la fonction `filter`.

Exercice de cours
1. Ecrire la fonction récursive factorielle `fact : int -> int`.
2. Faire un schéma de l'évolution de la pile d'appel de `fact 3`.

Exercice 1 : Types en OCaml
Donnez le type (s'il existe) des fonctions OCaml suivantes :

```
let f x y = x::y

let g x = x x

let h x y t = [x;y] :: t
```

Exercice 2 : Coefficient binomial
Pour k et n des entiers naturels tel que $k \leq n$, on définit $\binom{n}{k}$ (qui se lit "k parmi n"), qui est le nombre de parties à k éléments dans d'un ensemble à n éléments. Le coefficient binomial se calcule à l'aide de la formule :

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

1. Ecrire une fonction calculant le coefficient binomial de deux entiers n et k. Donner le type de la fonction.

Pour calculer un coefficient binomial, on peut aussi utiliser la formule de Pascal : $\binom{n+1}{k+1} = \binom{n}{k} + \binom{n}{k+1}$
2. Combien vaut $\binom{n}{0}$?
3. Ecrire une fonction récursive calculant le coefficient binomial de deux entiers n et k. Donner le type de la fonction.
4. Montrer par récurrence sur la n la propriété : $\forall 0 \leq k \leq n, \binom{n}{k} = \binom{n}{n-k}$.

Exercice 3 : La fonction map
1. Ecrire une fonction de type `int -> int` qui renvoie le double de l'entier passé en argument.
2. Ecrire une fonction de type `('a -> 'b)-> 'a list -> 'b list` qui prend en entrée une fonction et une liste, puis renvoie une nouvelle liste où la fonction a été appliquée à chaque élément de la liste initiale. **3.** Ecrire une fonction de type `int list -> int list` qui renvoie une liste des doubles d'une liste d'entier donné en argument.