

## Séance 4: MUTATION ET STYLE IMPÉRATIF

L3 – Université Nice Sophia Antipolis

### Exercice 1 (Tableaux, ☆)

1. Écrivez une fonction *moyenne* : `float array -> float` qui renvoie la moyenne d'un tableau de flottants. La fonction renvoie une erreur avec `invalid_arg` si le tableau est de longueur 0.
2. Écrivez une fonction *scalaire* : `float array -> float array -> float` qui calcule le produit scalaire de deux tableaux de même longueur. On rappelle que le produit scalaire de  $(a_1, \dots, a_n)$  et  $(b_1, \dots, b_n)$  vaut  $\sum_{i=1}^n a_i b_i$ .

□

### Exercice 2 (Tri sélection, \*\*)

1. Définissez la fonction *swap* : `'a array -> int -> int -> unit` telle que `swap t i j` échange les valeurs d'indice  $i$  et  $j$  de  $t$ .
2. Définissez la fonction *argmin* : `int -> 'a array -> int` telle que `argmin k [a0;...;an]` renvoie l'indice du plus petit  $a_i$  pour  $i$  compris entre  $k$  et  $n$  (les  $k$  premiers éléments ne sont pas pris en compte). Par exemple, `argmin 2 [10;2;5;1;4]` renvoie 3.
3. Déduisez-en une fonction *tri\_selection* : `'a array -> unit` qui trie un tableau en appliquant le tri sélection.

□

### Exercice 3 (Style impératif et style itératif, \*\*)

Programmez en style impératif, autrement dit sans utiliser la récurrence, uniquement à l'aide de boucles, les fonctions suivantes

1. la fonction *factorielle*: `int -> int` qui calcule la factorielle d'un nombre
2. la fonction *somme*: `int list -> int` qui calcule la somme d'une liste;
3. la fonction *argmin* : `'a list -> int` qui calcule l'indice du minimum d'une liste non vide. En cas de liste vide, une erreur est levée avec `invalid_arg`.
4. Reprogrammez ces fonctions en style itératif, autrement dit sans utiliser de référence et avec des fonctions récursives terminales.

□

### Exercice 4 (Listes mutables cycliques, \*\*)

On considère le type suivant

```
1 type mlist =  
2   | Empty  
3   | Cons of int ref * mlist ref
```

1. Définissez la fonction *somme* : `mlist -> int`

2. Définissez la fonction `set` : `mlist -> int -> int -> unit` telle que `set l i n` remplace l'entier à l'indice  $i$  par  $n$ , et fait une erreur avec `invalid_arg` si l'indice n'est pas correct.
3. Définissez la fonction `cycle` : `mlist -> unit` qui transforme une liste acyclique non vide en une liste cyclique.
4. Définissez la fonction `clength` : `mlist -> unit` qui renvoie la longueur d'une liste cyclique ou acyclique.

□