TD n° 3 : À la découverte de l'impératif

EXERCICE 1 (*Incrémentation et décrémentation*) Écrire les fonctions `incr` et `decr` qui permettent respectivement d'incrémenter et de décrémenter une référence sur un entier. Quel est leur type ?

EXERCICE 2 (*Attention : il y a des erreurs !*) Prévoir la réponse de CAML pour chacune des phrases suivantes :

```
let f1 x =
  let y = !x + 1 in
  y
;;
```

```
let f2 x y =
  for i = x to 10 do
    y := x :: !y
  done;
  !y
;;
```

```
let e3 = if true then "Hello !";;
let e4 = if true then print_string "Bye...";;
```

```
let e5 =
  let x = ref 10;
  while !x > 0 do
    print_int !x;
    print_char ' ';
    decr x
  done
;;
```

```
let e6 = let x = ref 0 in let y = x in x := x + 1; y;;
```

```
let f7 x y =
  if !x > 0 then
    decr x;
    y := !y + !x
  else
    incr x
;;
```

```
let foo = ref succ;;
let bar x = !foo x;;
bar 2;;
foo := function x -> x + 2;;
bar 2;;
```

```
let p = ref 0 in let q = ref p in !(!q) + (p := 2; !(!q));;
```

```
let clock = ref 0;;
let time () = incr clock; !clock;;
time ();;
time () + time ();;
time () - time ();;
```

EXERCICE 3 Écrire une fonction `puissance : int -> int -> int` qui implémente l'exponentiation naïve de manière itérative. Justifier sa terminaison, sa correction et calculer sa complexité.

EXERCICE 4 Écrire deux fonctions `somme_elements` et `produit_elements` qui calculent la somme et le produit des éléments d'un tableau. Quel est leur type ?

EXERCICE 5 Écrire une fonction `appartient : 'a vect -> 'a -> bool` qui vérifie si un élément donné est présent dans un tableau. Montrer que cette fonction termine, est correcte et calculer sa complexité.

EXERCICE 6 Écrire une fonction `echanger : 'a vect -> int -> int` qui échange le contenu de deux cases d'un tableau.

EXERCICE 7 Écrire des versions impératives du tri insertion puis du tri rapide. Les deux fonctions seront de type `'a vect -> 'a vect`. On pourra utiliser la fonction de l'exercice précédent.