TD n° 6 : Programmation impérative

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 :

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

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

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

OCAML
```
let e5 = let x = ref 0 in let y = x in x := x + 1; y;;
```

OCAML
```
let e6 =
  let x = ref 10;
  while !x > 0 do
    print_int !x;
    print_char ` `;
    decr x
  done
```

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

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

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

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

EXERCICE 3 *Changement de base*

On considère les fonctions suivantes :

`nombre_chiffres : int -> int -> int`

OCaml
```
let nombre_chiffres b n =
  let m = ref n in
  let nb = ref 0 in
  while !m > 0 do
    incr nb;
    m := !m / b
  done;
  !nb
```

`ecriture_base : int -> int -> int array`

OCaml
```
let ecriture_base b n =
  let nb = nombre_chiffres b n in
  let chiffres =  Array.make nb 0 in
  let m = ref n in
  for i = 0 to nb - 1 do
    chiffres.(i) <- !m mod b;
    m := !m / b
  done;
  chiffres
```

et `lecture_base : int -> int array -> list`

OCaml
```
let lecture_base b chiffres =
  let nb = Array.length chiffres in
  let n = ref 0 in
  for i = nb - 1 downto 0 do
    n := !n * b + chiffres.(i)
  done;
  !n
```

Pour chacune :

- exhiber un invariant de boucle, le prouver,
- prouver la terminaison,
- prouver la correction.

EXERCICE 4 *Recherche dichotomique*

1. Écrire une fonction `recherche_dicho : int array -> int -> int` tel que `recherche_dicho tab x` effectue une recherche dichotomique de `x` dans le tableau trié `tab` et renvoie un indice de position à laquelle se trouve `x` s'il est dans le tableau et `-1` sinon.

   On utilisera trois références `ind_g`, `ind_d` et `ind_m` pour effectuer le parcours dichotomique (pour, respectivement, « indice gauche », « indice droit » et « indice milieu »).
2. Décrire un invariant de boucle pour l'algorithme précédent, et le prouver. En déduire la correction de la fonction.