TD 2

Tableaux et chaines de caractères

Correction

Exercice 1 : Écrire une fonction `somme` qui renvoie la somme des valeurs d'un tableau d'entiers.

```
val somme : int array -> int = <fun>
```

Correction

Listing 1 – La fonction `somme`

```
let somme tab =
  let som = ref 0 in
  let taille = Array.length tab in
  for k = 0 to taille-1 do
    som := !som + tab.(k) ;
  done;
!som;;
```

Exercice 2 : Écrire la fonction `min_max` qui donne le plus grand et le plus petit élément d'un tableau :

```
val min_max : 'a array -> 'a * 'a  = <fun>
```

Correction

Listing 2 – La fonction `min_max`

```
let min_max tab =
  let mini = ref tab.(0) in
  let maxi = ref tab.(0) in
  let taille = Array.length tab in
  for i = 1 to taille -1 do
    if tab.(i) < !mini then
        mini := tab.(i) else
    if tab.(i) > !maxi then
        maxi := tab.(i) ;
  done;

(!mini , !maxi );;
```

Exercice 3 : Écrire la fonction `cat` qui concatène deux vecteurs.

```
val cat : 'a array -> 'a array -> 'a array = <fun>
```

Correction

Listing 3 – La fonction `cat`

```
let cat tab_1 tab_2 =
  let taille_1 = Array.length tab_1 in
```

```
  let taille_2 = Array.length tab_2 in
  let tab = Array.make (taille_1 + taille_2 ) (tab_1.(0)) in
  for i = 0 to taille_1 -1 do
      tab.(i) <- tab_1.(i) ;
  done ;
  for i = taille_1 to taille_1 + taille_2 -1 do
       tab.(i) <- tab_2.( i- taille_1  ) ;
  done ;
tab;;
```

Exercice 4 :

1. Écrire la fonction `map_tab` qui applique une fonction à tous les éléments du tableau, et renvoie le tableau modifié.

```
val map_tab : ('a -> 'a) -> 'a array -> 'a array
map_tab (fun x-> x*x) [| 7;4;1;|];;
- : int array = [|49; 16; 1|]
```

Correction

Listing 4 – La fonction `map_tab`

```
let map_tab fonction tab =
  let taille = Array.length tab in
  for i = 0 to taille -1 do
    tab.(i) <-fonction tab.(i) ;
  done;
  tab;;
```

2. Écrire la fonction `alea_tab` qui construit un tableau de taille n de nombres aléatoire inférieur à p. On utilisera la fonction `Random.int` : `int`$->$ `int` qui renvoie un nombre aléatoire entre 0 et l'argument (exclus).

```
val alea_tab : int -> int -> int array
alea_tab 10 8;;
- : int array = [|0; 3; 3; 6; 4; 6; 3; 0; 7; 0|]
```

Correction

Listing 5 – La fonction `alea_tab`

```
let alea_tab n p =
  let result = Array.make n p in
  map_tab (Random.int) result;;
```

Exercice 5 :

1. Écrire la fonction `permut` qui échange les éléments en position i et j d'un tableau.

```
val permut : 'a array -> int -> int -> unit = <fun>
```

Correction

Listing 6 – La fonction permut

```
let permut tab i j =
  let temp = tab.(i) in
  tab.(i) <- tab.(j) ;
  tab.(j) <- temp ;;
```

2. Écrire une fonction permut_tab qui permute aléatoirement les éléments d'un tableau.

```
val permut_tab : 'a array -> 'a array = <fun>
permut_tab [| 8;1;3;4;5;7 |];;
- : int array = [|3; 8; 4; 7; 1; 5|]
```

Correction

Listing 7 – La fonction permut_tab

```
let permut_tab tab =
  let taille = Array.length tab in
  for i = taille -1 downto 1 do
    let p = Random.int (i+1) in
    permut tab i p ;
  done;
tab;;
```

Exercice 6 :

1. Écrire la fonction eratosthene qui, pour un entier n donné ($n > 2$), renvoie un tableau de taille n de booléen informant si l'indice i est un nombre premier.

```
val eratosthene : int -> bool array
eratosthene 10;;
- : bool array =
 [|false; false; true; true; false; true; false; true; false; false|]
```

Correction

Listing 8 – La fonction eratosthene

```
let eratosthene n =
  let tab_result = Array.make n true in
  tab_result.(0) <- false ;
  tab_result.(1) <- false ;
  for i = 2 to n-1 do
    let k = ref 2 in
    while (!k * i < n)&& tab_result.(i) do
       tab_result.(!k * i) <- false ;
       incr(k)
    done ;
  done;
  tab_result ;;
```

2. Améliorer le code précédent pour avoir en même temps le nombre de nombre premier du tableau.

```
eratosthene_et_nb : int -> bool array * int
```

Correction

Listing 9 – La fonction `eratosthenes_et_n`

```
let eratosthene_et_nb  n =
  let tab_result = Array.make n true in
  tab_result.(0) <- false ;
  tab_result.(1) <- false ;
  let compteur = ref 0 in
  for i = 2 to n-1 do
    let k = ref 2 in
    if tab_result.(i) then incr(compteur) ;
    while (!k * i < n)&& tab_result.(i) do
       tab_result.(!k * i) <- false ;
       incr(k)
    done ;
  done;
  (tab_result , !compteur ) ;;
```