Corrigé de la feuille d'exercices n°4.

1.

```
let minmax t a b =
let i = ref a and j = ref a in
  for k = a+1 to b do
    if t.(k) < t.(!i) then i:= k;
    if t.(k) > t.(!j) then j:= k;
  done;
(!i,!j);;

let permute t i j =
let temp = t.(i) in
t.(i) <- t.(j);
t.(j) <- temp;;
```

Première version du programme qui en marche pas.

```
let tri_selection t =
let n = vect_length t in
let a = ref 0 and b = ref (n-1) in
while (!b > !a) do
  let (i,j) = minmax t (!a) (!b) in
     permute t !a i;
     permute t !b j;
     incr a; decr b;
done;;
```

Deuxième version avec le bug corrigé.

```
let tri_selection t =
let n = vect_length t in
let a = ref 0 and b = ref (n-1) in
while (!b >= !a) do
  let (i,j) = minmax t (!a) (!b) in
     permute t !a i;
     if (!a = j) then permute t i !b
                 else permute t j !b;
     incr a; decr b;
done;
t;;

tri_selection : 'a vect -> 'a vect = <fun>
```

2. triangle de Pascal version tableaux de tableaux.

```
let pascal n=
let t = make_vect (n+1) [||] in
for i = 0 to n do
t.(i) <- make_vect (i+1) 1;
for j = 1 to i-1  do
   t.(i).(j) <- t.(i-1).(j-1) + t.(i-1).(j)
   done;
done;
t;;
```

3.

```
let map_vect f t =
let n = vect_length t in
let tt = make_vect n (f t.(0)) in
for i = 1 to vect_length t -1 do
     tt.(i) <- f(t.(i))
done;
tt;;

#map_vect (function x -> x*x+1) [|1;2;3;4|];;
- : int vect = [|2; 5; 10; 17|]
```

4.

```
let retourne t =
let n = vect_length t in
for i = 0 to n/2-1 do
   permute t i (n-1-i);  (* ne pas oublier de s'arrêter à la moitié du tableau. *)
done;
t;;
```

5. on réécrit la fonction "`make_matrix`" juste pour voir comment elle peut être écrite

```
let make_matrix n p x =
let M = make_vect n [||] in
for i = 0 to n-1 do
    M.(i) <- make_vect p x
done;
M;;
```

Produit externe sur les matrices.

```
let prodext a M =
let n = vect_length M and p = vect_length M.(0) in
let P = make_matrix n p M.(0).(0) in
   for i = 0 to n-1 do
     for j = 0 to p-1 do
        P.(i).(j) <- a*P.(i).(j);
     done;
   done;
P;;
```

Produit matriciel.

```
let produit M N =
let n = vect_length M and p = vect_length M.(0)
and p1 = vect_length N and q = vect_length N.(0) in
if p <> p1 then failwith "dimension de matrices incompatibles"
 else let P = make_matrix n q M.(0).(0) in
   for i = 0 to n-1 do
     for j = 0 to q-1 do
       let c = ref (M.(i).(0) * N.(0).(j)) in
          for k = 1 to p-1 do
              c := !c + M.(i).(k) * N.(k).(j)
          done;
        P.(i).(j) <- !c;
     done;
   done;
P;;
```

6.

```
let appartient x t =
let trouve = ref false in
for i = 0 to vect_length t -1 do
   if x = t.(i) then  trouve := true
done;
!trouve;;

appartient 5 [|4;7;8;3;1;20;39|];;

appartient "un" [|"par";"un";"beau";"matin";"d'avril"|];;
```

```
let ajoute x t =
let n = vect_length t in
let tt = make_vect (n+1) x in
for i = 1 to n do
   tt.(i) <- t.(i-1)
done;
tt;;

let intersect ens1 ens2 =
match ens1=[||] with
  true -> [||];   (* cas où ens1 = ∅ *)
| false -> (let n = vect_length ens1 in
          let inter = ref [||] in
             for i = 0 to n-1 do
                if appartient ens1.(i) ens2
                      then inter := ajoute ens1.(i) !inter;
             done;
          !inter)
;;

let réunion ens1 ens2 =
match ens1=[||] with
  true -> ens2;   (* cas où ens1 = ∅ *)
| false -> (let n = vect_length ens1 in
          let inter = ref (copy_vect ens2) in
             for i = 0 to n-1 do
                if not(appartient ens1.(i) ens2)
                      then inter := ajoute ens1.(i) !inter;
             done;
          !inter)
;;

let verifie t =
let appartient x t a b =
let trouve = ref false in
for i = a to b do
    if x = t.(i) then trouve := true
done;
!trouve in
let ensemble =ref true in
let n = vect_length t-1 in
for i =0 to n do
  if appartient t.(i) t (i+1) n then ensemble := false;
done;
!ensemble;;
```