Corrigé feuille d'exercices n°18 de l'option d'informatique.

Exercice n°1

```
let quadrant M =
let res = ref 0 and x = M.x and y = M.y in
     if x >= 0. && y < 0. then 1
else if x > 0. && y >= 0. then 2
else if x <= 0. && y > 0. then 3
else 4;;
```

Exercice n°2

```
let OrdreRadar M1 M2 =
  match (quadrant M1) - (quadrant M2) with
   x when x < 0 -> -1  (* M1 est rencontré avant M2 *)
 | x when x > 0 -> 1 (* M2 est rencontré avant M1 *)
 | x -> (* M1 et M2 sont dans le même quadrant *)
          let x1 = M1.x and y1 = M1.y and x2 = M2.x and y2 = M2.y in
          let d = x1 *. y2 -. x2 *. y1 in
               if d = 0. then 0
                          else (if d > 0. then  -1 else 1);;
```

Exercice n°3

```
let genere_pts n =
let l = ref [] in
for i = 1 to n do
   l := {x=float_of_int(random__int(size_x()));
         y = float_of_int(random__int(size_y()))}::!l
done;
!l;;
```

Exercice n°4

```
let rec affiche_pts = function
  [] -> ()
| M::Ms -> fill_circle (int_of_float M.x) (int_of_float M.y) 2;
           affiche_pts Ms;;
```

Exercice n°5

```
let rec point_depart = function
 [] -> failwith "liste vide"
| [M] -> M
| M::Ms -> let M1 = point_depart Ms in
              if M.x < M1.x then M
    else if M.x > M1.x then M1
    else if M.y > M1.y then M else M1;;
```

Exercice n°6

```
let ptnul = {x=0.001;y=2000.1};; (* point vide *)

let dist M = abs_float M.x +. abs_float M.y;;

let diff M1 M2 =
  {x = M2.x -. M1.x; y= M2.y -. M1.y};;

let rec point_suivant M dir l =
let Pt_suivant = ref ptnul in
let ll = ref l in
for i = 1 to list_length l do
   let M1 = hd !ll in
    (match OrdreRadar (diff M M1) dir with
    1 ->( if !Pt_suivant = ptnul && M1 <> M
             then Pt_suivant := M1
             else if (OrdreRadar (diff M M1) (diff M !Pt_suivant) = -1 && M1 <> M)
                      then Pt_suivant := M1);
  | 0 -> if dist (diff M M1) > dist (diff M !Pt_suivant)
                            then Pt_suivant := M1;
  | _ -> (); );
   ll := tl(!ll);
done;
!Pt_suivant;;
```

Exercice n°7

```
let rec point_SO = function
 [] -> failwith "liste vide"
| [M] -> M
| M::Ms -> let M1 = point_SO Ms in
             if M.x < M1.x then M
    else if M.x > M1.x then M1
    else if M.y < M1.y then M else M1;;

let envconvexe l =
let poly = ref [] and dir = ref {x = 0.;y= -1.} in
let M = point_depart l and M' = point_SO l in
if (M <> M') then poly := [M';M] else poly := [M];
let M1 = ref M' and M2 = ref (point_suivant M' !dir l) in
while not(M = !M2) do
    poly := !M2::!poly;
    dir := diff !M1 !M2; M1 := !M2;
    M2 := point_suivant !M1 !dir l;
done;
M::!poly;;
```

Exercice n°8

```
let dessine_polygone l =
let rec dessine_poly = function
  [] -> ()
| M::Ms -> lineto (int_of_float M.x) (int_of_float M.y);
           dessine_poly Ms in
let M = hd l in
moveto (int_of_float M.x) (int_of_float M.y);
dessine_poly (tl l);;
```