Option Informatique en Sup MPSI
TP : des immeubles à l'horizon, le corrigé

Question 1 • Deux versions :

```
let rec cité_valide = function
 | [] -> true
 | (g,h,d)::q -> (g<d) & (h>0) & (cité_valide q) ;;

let cité_valide =
 let immeuble_valide (g,h,d) = (g<d) & (h>0) in
 for_all immeuble_valide ;;
```

Question 2 • La traduction en Caml est immédiate :

```
let rec contour_valide = function
 | [g;h;d] -> g<d & h>0
 | g::h::d::q -> g<d & h>=0 & contour_valide (d::q)
 | _ -> false ;;
```

Question 3 • Ici encore, la traduction en Caml est immédiate :

```
let rec contour_réduit = function
 | [g;h;d] -> g<d & h>0
 | g::h::d::h'::q -> g<d & h>=0 & h <> h' & contour_réduit (d::h'::q)
 | _ -> false ;;
```

Question 4 • À chaque itération, nous dessinons un trait vertical, puis un trait horizontal. Avant la première itération, nous traçons un trait horizontal menant de l'origine $(0,0)$ au bas du bord gauche du contour : c'est le point $(t,0)$ où t est la tête de liste. Après la dernière itération, nous dessinons le bord vertical droit du contour. Précisons ce qui doit être fait à la k-ième itération : nous connaissons l'abscisse maximale x et l'ordonnée y du dernier segment horizontal tracé. La liste commence par les valeurs h et d ; nous traçons un segment vertical menant de (x,y) à (x,h) puis un segment horizontal menant de (x,h) à (d,h). Nous remplaçons x et y par d et h, et sommes prêts pour la $(k+1)$-ième itération. Une fois la liste épuisée, il reste à tracer un segment vertical menant du point courant (x,y) au point $(x,0)$.

```
let dessine_contour l =
 clear_graph() ;
 let k = 10 in
 let rec aux x y = function
  | h::d::q -> lineto x h ; lineto d h ; aux d h q
  | [] -> lineto x 0
  | _ -> failwith "liste incorrecte"
 in moveto (hd l) 0 ; aux (hd l) 0 (tl l) ;;
```

Question 5 • À nouveau, la traduction en Caml est immédiate :

```
let rec réduire_contour = function
 | g::h::d::h'::d'::q when h=h' -> réduire_contour (g::h::d'::q)
 | g::h::q -> g::h::(réduire_contour q)
 | [x] -> [x]
 | _ -> failwith "liste incorrecte" ;;
```

Question 6 • La question délicate ! Il suffit, comme l'indiquait l'énoncé, de savoir insérer un nouvel immeuble dans un contour. L'immeuble est décrit par (g,h,d). Nous allons le présenter successivement à chacun des «blocs» définis par le contour ; un tel bloc est décrit par (g',h',d'). Il existe au plus 25 cas de figure, puisque g (resp. d) peut occuper au plus cinq places différentes par rapport à g' et d'. En fait, il n'y a que treize cas à examiner, plus deux cas possibles quand on arrive en fin de liste.

```
let rec insère_immeuble (g,h,d) = function
 | [] -> [g;h;d]
 | g'::q when d<g' -> g::h::d::0::g'::q
 | g'::q when d=g' -> g::h::g'::q
 | g'::h'::d'::q when g<g' & g'<d & d<d' -> g::h::g'::(max h h')::d::h'::d'::q
 | g'::h'::d'::q when g<g' & d=d' -> g::h::g'::(max h h')::d'::q
 | g'::h'::d'::q when g<g' & d>d' ->
   let q' = insère_immeuble (d',h,d) (d'::q) in g::h::g'::(max h h')::q'
 | g'::h'::d'::q when g=g' & d<d' -> g::(max h h')::d::h'::d'::q
 | g'::h'::d'::q when g=g' & d=d' -> g::(max h h')::d::q
 | g'::h'::d'::q when g=g' & d>d' ->
   let q' = insère_immeuble (d',h,d) (d'::q) in g::(max h h')::q'
 | g'::h'::d'::q when g'<g & g<d & d<d' -> g'::h'::g::(max h h')::d::h'::d'::q
 | g'::h'::d'::q when g'<g & g<d & d=d' -> g'::h'::g::(max h h')::d'::q
 | g'::h'::d'::q when g'<g & g<d' & d'<d ->
   let q' = insère_immeuble (d',h,d) (d'::q) in g'::h'::g::(max h h')::q'
 | g'::h'::d'::q when g'<g & g=d' & d'<d ->
   let q' = insère_immeuble (d',h,d) (d'::q) in g'::h'::q'
 | g'::h'::d'::q when g'<d' & d'<g ->
   let q' = insère_immeuble (g,h,d) (d'::q) in g'::h'::q'
 | [x] when x=g -> [g;h;d]
 | [x] when x<g -> [x;0;g;h;d]
 | _ -> failwith "liste incorrecte"
;;

let rec contour_of_cité = function
 | [] -> []
 | (g,h,d)::q -> insère_immeuble (g,h,d) (contour_of_cité q) ;;
```

La liste obtenue n'est pas réduite ; nous aurions pu effectuer la réduction au fur et à mesure que nous insérions les immeubles, mais ceci aurait encore alourdi la programmation.

Question 7 • Reprenons le cadre utilisé à la question 4, en l'adaptant au nouvel objectif.

```
let latex_of_contour c =
 let ps = print_string and pi = print_int in
 let print_put x y =
  ps "\\put(" ; pi x ; ps "," ; pi y ; ps ")" in
 let vline g b h = match h-b with
  | 0 -> ()
  | k -> print_put g b ; ps "{\\line(0," ; pi (if k>0 then 1 else -1) ;
     ps "){" ; pi (if k>0 then k else -k) ; ps "}}\n"
 and hline h g d = match d-g with
  | 0 -> ()
  | k -> print_put g h ; ps "{\\line(1,0){" ; pi k ; ps "}}\n" in
 let rec aux t = function
  | [g;h;d] -> vline g t h ; hline h g d ; vline d 0 h ; hline 0 d (d+10)
  | g::h::d::q -> vline g t h ; hline h g d ; aux h (d::q)
  | _ -> failwith "contour mal construit"
 in hline 0 0 (hd c) ; aux 0 c ;;
```

FIN