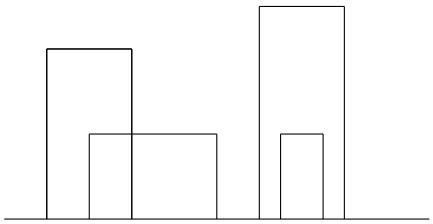
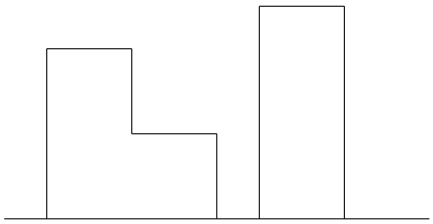
Option Informatique en Sup MPSI
TP : des immeubles à l'horizon

▶ Vous trouverez sur le serveur Web du laboratoire d'informatique (`newbox/optinfosup/index.html`) un pointeur vers un ensemble de fichiers qui pourront vous être utiles.

▶ Face à nous se profilent les immeubles d'un quartier moderne. Voici un exemple avec quatre immeubles ≪tranparents≫ :



Chaque immeuble est un parallélépipède[1] ; les immeubles les plus proches cachent, en tout ou en partie, ceux qui sont plus éloignés. Pour un observateur placé à l'infini, l'ensemble des immeubles apparaît comme une figure géométrique, délimitée par des traits horizontaux et verticaux. Voici comment notre exemple lui apparaît :



▶ Un repère orthonormé $(O; i, j)$ ayant été fixé, chaque immeuble est défini par un triplet (g, h, d) où g est l'abscisse[2] du bord gauche, d l'abscisse du bord droit et h la hauteur. Pour tout immeuble, nous avons $g < d$ et $h > 0$. Ainsi, la cité représentée plus haut peut être décrite par la liste :

`[(20,20,50);(65,20,75);(10,40,30);(60,50,80)]`.

Question 1 • Rédigez en Caml une fonction :

```
cité_valide : int*int*int list -> bool
```

spécifiée comme suit : `cité_valide l` indique si la liste ℓ décrit bien une famille d'immeubles.

▶ Notre objectif est de construire le contour vu par l'observateur situé à l'infini. Cette ligne géométrique peut être décrite par une liste $\ell = (\ell_0, \ldots, \ell_{2n})$ de nombres, de longueur $2n + 1 \geqslant 3$, vérifiant les conditions suivantes :

- pour $0 \leqslant i \leqslant n$, ℓ_{2i} est l'abscisse du i-ième segment vertical ;
- pour $0 \leqslant i < n$, ℓ_{2i+1} est l'ordonnée du i-ième segment horizontal.

Par exemple, la liste décrivant le contour associé à la cité dessinée plus haut serait :

`[10;40;30;20;50;0;60;50;80]`

▶ Une liste $(x_i)_{0 \leqslant i \leqslant 2n}$ est dite *valide* si elle vérifie les deux conditions suivantes :

[1] et non un *parallépipède* comme le dit l'auteur du sujet d'informatique MP/PC de Polytechnique 2005

[2] et non l'*abcisse*, comme l'écrit l'auteur précédemment cité

- $x_{2i} < x_{2i+2}$ pour $0 \leqslant i < n$;
- $x_{2i+1} \geqslant 0$ pour $0 \leqslant i < n$.

Elle est dite *réduite* si elle vérifie de plus la condition $x_{2i+1} \neq x_{2i+3}$ pour $0 \leqslant i < n-1$.

Question 2 • Rédigez en Caml une fonction :

```
contour_valide : int list -> bool
```

spécifiée comme suit : `contour_valide x` indique si la liste x est valide, au sens qui vient d'être défini.

Question 3 • Rédigez en Caml une fonction :

```
contour_réduit : int list -> bool
```

spécifiée comme suit : `contour_réduit x` indique si la liste x est réduite, au sens qui vient d'être défini.

Question 4 • Rédigez en Caml une fonction :

```
dessine_contour : int list -> unit
```

spécifiée comme suit : `dessine_contour x` dessine le contour défini par la liste x (supposée valide et réduite). Vous utiliserez les fonctions graphiques décrites en annexe.

Question 5 • Rédigez en Caml une fonction :

```
réduire_contour : int list -> int list
```

spécifiée comme suit : `réduire_contour x` réduit la liste x (supposée valide).

Question 6 • Rédigez en Caml une fonction :

```
contour_of_cité : int*int*int list -> int list
```

spécifiée comme suit : `contour_of_cité l` calcule le contour associé à la cité ℓ. Remarque : il suffit de savoir ajouter un immeuble...

Question 7 • Rédigez en Caml une fonction :

```
latex_of_cité : int*int*int list -> string -> unit
```

spécifiée comme suit : `latex_of_cité l s` écrit dans le fichier de nom s la liste des instructions LaTeX permettant de dessiner le contour de la cité décrite par la liste ℓ. Vous trouverez sur le site Web un exemple, un résumé de la syntaxe et le nécessaire pour ≪emballer≫ le tout.

Annnexe : instructions graphiques

▶ Le plus simple est de donner un exemple !

```
#open "graphics" ;; (* placer au début du programme *)

let dessine_maison() =
 moveto 100 100 ;
 lineto 300 100 ; lineto 300 200 ;
 lineto 200 250 ; lineto 100 200 ; lineto 100 100 ;
 moveto 100 200 ; lineto 300 200
;;

open_graph "" ;;
dessine_maison() ;;
```

FIN