TP ACT n°2
Philippot Grégoire
Pelage François-Xavier

--Q1
comme la base du rectangle est sur l'axe x, on peut en déduire qu'un rectangle de surface maximal à nécessairement deux sommets de la forme (xi
, 0),(xj , 0) avec $0 \leq i < j \leq n - 1$.

h étant la hauteur max du plan,
on aurait :
surface = h * (xj-xi)

```
// x entier >= 0, y entier >= 0, n entier >= 0, l liste de
coordonnées int surface (Int x , Int y, Int n, List l){
        int hmax =0;
        int h=1;
        int long_x;
        int long_y;
        for (i=0;i<n;i++)
                { for(j=i+1;j<n;j++)
                {
                        hmax = l[i+1][1] // hauteur du point suivant
                        while(l[i+1][1]>l[i+1+h][1]){
                                h++; //chercher si des points qui suivent sont dans le rectangle
                        }
                        long_x= l[i+h+1][0]-l[i][0]
                        h=1;
                        long_y= hmax;
                        if (res > long_x * long_y)
                                { res= long_x *
                                long_y
                        }end if
                }end for
        }end for
        return res
}
```

la complexité est en $\Theta(n^3)$
Nous n'avons pas réussi à tester notre code sur la plate forme
Pour n=100 000 au vu de la complexité que nous avons, nous pouvons en déduire qu'il ne se fera pas exécuté en 1s.

--Q2
Oui car on peut toujours s'occuper séparément des élément de la liste. Ici on coupera la liste au point dont l'ordonnée est la plus petite puis on calculera leur surfaces et on en ressortira la surface maximale.

// x entier >= 0, y entier >= 0, n entier >= 0, l liste de coordonnées int div_pour_regner(Int x , Int y, Int n, List l){

```
        int hmin = l[2][1]
        int index = -1
        for(i = 0; i<n ; i++){
                if  (hmin  >=  l[i][2])
                        {  hmin  =  l(i)
                        [1] index = i
                }end if
        }end for
        List div = div(points,index) // div renvoyant une structure contenant les 2 parties de
la liste(avant et après l'index
        int res = max(surface(div[0]),surface(div[1]))); // surface max entre les 2 parties de la liste
        return res;
}
```

La complexité dans le meilleurs des cas est en
Θ(nlogn) la complexité dans le pire des cas est en
$\Theta(n^3)$.