Mamouni My Ismail
Professeur agrégé de mathématiques
Enseignant en classes de MP
CPGE My Youssef
Rabat, Maroc
mamouni.myismail@gmail.com
myismail.chez.com

ECOLE POLYTECHNIQUE
ECOLE SUPERIEURE DE PHYSIQUE ET CHIMIE INDUSTRIELLES
CONCOURS 2003
FILIERE MP-OPTION SCIENCES INDUSTRIELLES
FILIERE PC
EPREUVE FACULTATIVE D'INFORMATIQUE
****

l'enclos du robot
Frontière Sud-Ouest

```
>  restart;
```

Question 1.

```
>  sudouest:=proc(P,Q) local x,y;
>  x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
>  if x[1]<=x[2] and y[1]<=y[2] then 1
>  else 0
>  fi;
>  end:
>  nordouest:=proc(P,Q) local x,y;
>  x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
>  if x[1]<=x[2] and y[1]>=y[2] then 1
>  else 0
>  fi;
>  end:
>  sudest:=proc(P,Q) local x,y;
>  x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
>  if x[1]>=x[2] and y[1]<=y[2] then 1
>  else 0
>  fi;
>  end:
>  nordest:=proc(P,Q) local x,y;
>  x:=[P[1],Q[1]]:y:=[P[2],Q[2]]:
>  if x[1]>=x[2] and y[1]>=y[2] then 1
>  else 0
>  fi;
>  end:
```

Question 2

```
>  echange:=proc(a,b,i,j) local p,q,a1,b1;
>  p:=min(i,j);q:=max(i,j);
>  if p=1 then
>  if q=nops(a) then
>  a1:=[a[q],seq(a[k],k=2..nops(a)-1),a[1]];
> b1:=[b[q],seq(b[k],k=2..nops(b)-1),b[1]];
> else
> a1:=[a[q],seq(a[k],k=2..q-1),a[1],seq(a[k],k=q+1..nops(a))];
> b1:=[b[q],seq(b[k],k=2..q-1),b[1],seq(b[k],k=q+1..nops(b))];
> fi;
> else
> if q=nops(a) then
> a1:=[seq(a[k],k=1..p-1),a[q],seq(a[k],k=p+1..q-1),a[p]];
> b1:=[seq(b[k],i=1..p-1),b[q],seq(b[k],k=p+1..q-1),b[p]];
> else
> a1:=[seq(a[k],k=1..p-1),a[q],seq(a[k],k=p+1..
> q-1),a[p],seq(a[k],k=q+1..nops(a))];
> b1:=[seq(b[k],k=1..p-1),b[q],seq(b[k],k=p+1..q-1),b[p],seq(b[k],k=q+1.
> .nops(b))];     fi;
> fi;
> RETURN([a1,b1]);
> end:
```

```
Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `i` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local

Warning, `k` in call to `seq` is not local
```

Question 3 c'est le point P1

Question 4

Ecrivons d'abord la fonction testSO qui retourne 0 si un point donné est dans la frontière SudOuest, et une valeur non nulle dans le cas contraire

```
> testSO:=proc(a,b,i) local a1,b1,N,j;
> a1:=op(1,echange(a,b,1,i));
> b1:=op(2,echange(a,b,1,i));
> N:=0:for j from 2 to nops(a) do
> N:=N+sudouest([a1[j],b1[j]],[a1[1],b1[1]]);
> od;
> RETURN(N);
> end:
```

Terminons enfin par récupérer les points qui se trouvent sur la frontière SudOuest, c'est à dire pour lesquels la valeur retournée par testSO est 0

```
> frontiereSO:=proc(a,b) local aSO,bSO,i;
> global nSO;
> aSO:=NULL:
> bSO:=NULL:
> for i from 1  to nops(a) do
> if testSO(a,b,i)=0 then
> aSO:=aSO,a[i];bSO:=bSO,b[i];
> else fi;
> od;
> nSO:=nops([aSO]);
> RETURN(seq([op(i,aSO),op(i,bSO)],i=1..nSO));
> end:
```

Question 5

```
> testNO:=proc(a,b,i) local a1,b1,N,j;
> a1:=op(1,echange(a,b,1,i));
> b1:=op(2,echange(a,b,1,i));
> N:=0:for j from 2 to nops(a) do
> N:=N+nordouest([a1[j],b1[j]],[a1[1],b1[1]]);
> od;
> RETURN(N);
> end:
> frontiereNO:=proc(a,b) local aNO,bNO,i;
> global nNO;
> aNO:=NULL:
> bNO:=NULL:
> for i from 1  to nops(a) do
> if testNO(a,b,i)=0 then
> aNO:=aNO,a[i];bNO:=bNO,b[i];
> else fi;
> od;
> nNO:=nops([aNO]);
> RETURN(seq([op(i,aNO),op(i,bNO)],i=1..nNO));
> end:
```

Question 6

```
> testSE:=proc(a,b,i) local a1,b1,N,j;
> a1:=op(1,echange(a,b,1,i));
> b1:=op(2,echange(a,b,1,i));
> N:=0:for j from 2 to nops(a) do
> N:=N+sudest([a1[j],b1[j]],[a1[1],b1[1]]);
> od;
> RETURN(N);
> end:
> frontiereSE:=proc(a,b) local aSE,bSE,i;
> global nSE;
> aSE:=NULL:
> bSE:=NULL:
> for i from 1  to nops(a) do
> if testSE(a,b,i)=0 then
> aSE:=aSE,a[i];bSE:=bSE,b[i];
> else fi;
> od;
> nSE:=nops([aSE]);
> RETURN(seq([aSE[i],bSE[i]],i=1..nSE));
> end:
> testNE:=proc(a,b,i) local a1,b1,N,j;
> a1:=op(1,echange(a,b,1,i));
> b1:=op(2,echange(a,b,1,i));
> N:=0:for j from 2 to nops(a) do
> N:=N+nordest([a1[j],b1[j]],[a1[1],b1[1]]);
> od;
> RETURN(N);
> end:
> frontiereNE:=proc(a,b) local aNE,bNE,i;
> global nNE;
> aNE:=NULL:
> bNE:=NULL:
> for i from 1  to nops(a) do
> if testNE(a,b,i)=0 then
> aNE:=aNE,a[i];bNE:=bNE,b[i];
> else fi;
> od;
> nNE:=nops([aNE]);
> RETURN(seq([aNE[i],bNE[i]],i=1..nNE));
> end:
> frontiereSO(a,b);
```

$$[1, 1]$$

```
> with(plots):
```

Question 7

On définit maintenant la fonction SO qui permet de tracer la frontière SudOuest en joignant ses point à l'aide de la fonction plot

```
> tracer:=proc(P) local Pts,i;
> Pts:=NULL:
> for i from 1 to nops(P)-1 do
> Pts:=Pts,P[i],[P[i][1],P[i+1][2]];
> od;
> Pts:=Pts,P[nops(P)];
> end:
> P:=[[0,11],[2,0],[2,1],[3,8],[3,7],[4,8],[4,2
> ],[4,0],[5,3],[5,6],[6,9],[6,12],[6,11],[7,9],[9,3],[9,11],[10,8],[10,
> 6],[10,10],[11,1]];a:=[seq(P[i][1],i=1..nops(P))];b:=[seq(P[i][2],i=1.
> .nops(P))];
```

$$P := [[0, 11], [2, 0], [2, 1], [3, 8], [3, 7], [4, 8], [4, 2], [4, 0], [5, 3], [5, 6], [6, 9], [6, 12], [6, 11], [7, 9], [9, 3], [9, 11], [10, 8], [10, 6], [10, 10], [11, 1]]$$

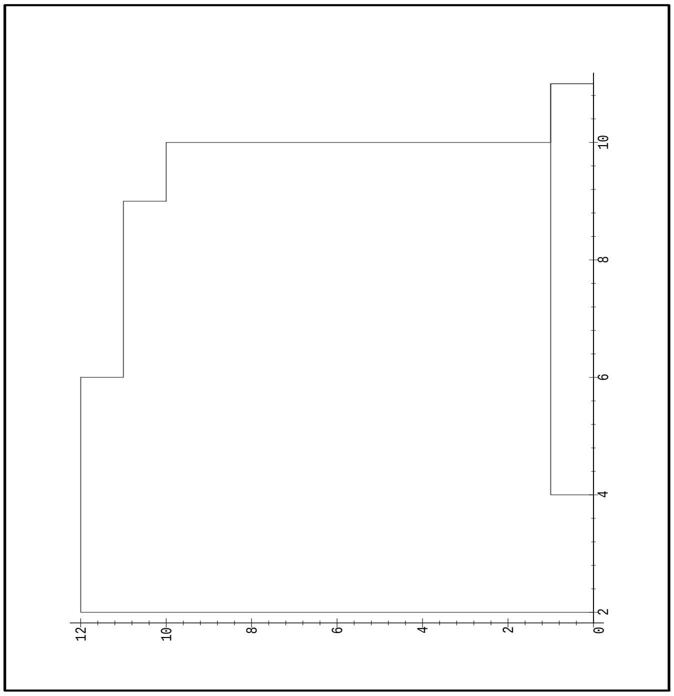
$$a := [0, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6, 6, 6, 7, 9, 9, 10, 10, 10, 11]$$

$$b := [11, 0, 1, 8, 7, 8, 2, 0, 3, 6, 9, 12, 11, 9, 3, 11, 8, 6, 10, 1]$$

```
>  P1:=[frontiereNO(a,b),frontiereNE(a,b),fronti
>  ereSE(a,b),frontiereSO(a,b),frontiereNO(a,b)];
```

$$P1 := [[6, 12], [6, 12], [9, 11], [10, 10], [11, 1], [4, 0], [11, 1], [2, 0], [6, 12]]$$

```
>  plot([tracer(P1)]);
```


12
10
8
6
4
2
0
2
4
6
8
10