بِسمِ اللَّهِ الرَّحمَنِ الرَّحِيمِ
وَ عَلَى اللَّهِ فَليَتَوَكَّلِ المُتَوَكِّلون

صَدَقَ اللَّهُ العَظِيم

Programmation Maple© : *Corrigé de concours*

Annonce

Dans le cadre des activités organisées par le BDE des CPGE Med V Casablanca, et dans le but d'intéresser au mieux les élèves aux journées d'informatiques prevues aux CPGE Med V, Casablanca les 22-23 Mars, Mr Mamouni professeur de mathématiques en classes MPSI à le plaisir dinviter les élèves des CPGE Med V, et toute autre personne interessée à deux séances dinitiation à la programmation et l'algorithmique.

1) **1ère séance** : vendredi 7 Mars 2008, 16h30-18h :
   a) Intitulé : Initiation sur des exemples simples
   b) Contenu :
      - Cours de programmation
      - Tester si un nombre est premier
      - Décomposer un entier en base b
      - Methode de dichotomie
      - Calcul approché de π
      - Cryptographie RSA
2) **2ème séance** : Vendredi 14 Mars 2008, 16h30-18h :
   a) Intitulé : Programmation avancéé.
   b) Contenu :
      - Corrigé de l'épreuve d'informatique, Centrale-Sup Elec, 2004.
      - Corrigé de l'épreuve d'informatique, X-Polytechnique, 2003.

Language de programmation : Maple ©

Mamouni My Ismail
Professeur de mathématiques
MPSI 2, CPGE Med V
Casablanca.

Corrigé de l'épreuve d'informatiques, X-Polytechnique, 2003.

```
> 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
> end if;
> 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
> end if;
> 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
> end if;
> 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
> end if;
> 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),
> 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
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:
```

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 di

```
> 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 end if;
> 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 end if;
> 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 end if;
> 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 end if;
> od;
> nNE:=nops([aNE]);
> RETURN(seq([aNE[i],bNE[i]],i=1..nNE));
> end:
```

```
> frontiereSO(a,b);
```

$$[1,1]$$

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

```
\begin{Maple Warning}{\small{Warning, the name changecoords has been redefined}}\end{Maple Wa
```

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]
```

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

```
> P1:=[frontiereNO(a,b),frontiereNE(a,b),frontiereSE(a,b),frontiereSO(a,b),frontiereN
```

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

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

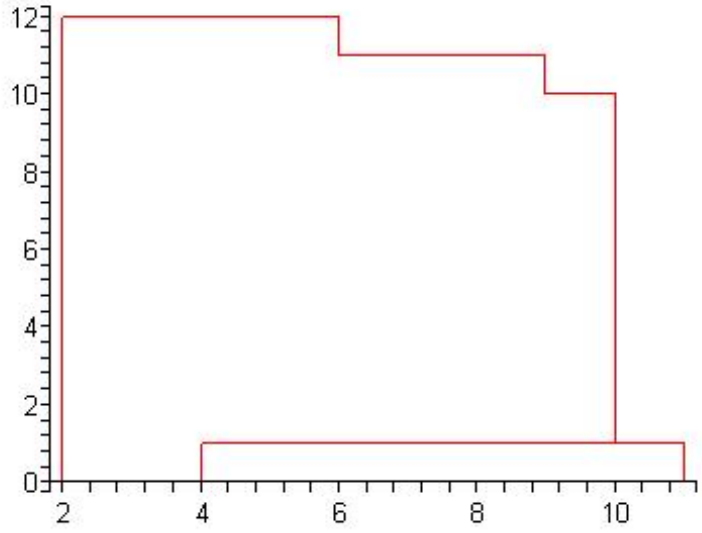

FIG. 1 – image

Corrigé de l'épreuve d'informatique, Centrale-Sup Elec., 2004.

```
> restart:
```

On commence par déclarer les écoles, le nombre de postes ouverts pour chaque école

```
> Ecoles:=[E1,E2,E3];N:=[1,2,1];candidats:=[C1,C2,C3,C4];
```

$$Ecoles := [E1, E2, E3]$$
$$N := [1, 2, 1]$$
$$candidats := [C1, C2, C3, C4]$$

Puis le classement de chaque école pour les candidats

```
> E1:=[C2,C4];E2:=[C4,C3,C1,C2];E3:=[C1,C2,C3];
```

$$E1 := [C2, C4]$$
$$E2 := [C4, C3, C1, C2]$$
$$E3 := [C1, C2, C3]$$

Les voeux de chaque candidat classés dans l'ordre de préference, la 1ére case répresente u

```
> C1:=[1,3,2];C2:=[2,3,2,1];C3:=[3,2,3];C4:=[4,1,2];
```

$$C1 := [1, 3, 2]$$
$$C2 := [2, 3, 2, 1]$$
$$C3 := [3, 2, 3]$$
$$C4 := [4, 1, 2]$$

Pour chaque école, on va chercher les candidats à éliminer, en commençant bien sûr par l

```
> choix_ecole:=proc(i) local Ecole,j,Nbr,k,candidat,a,b,NewEcole;
> Ecole:=op(i,Ecoles):NewEcole:=NULL:
> for j from nops(Ecole) to 1 by -1 do
> Nbr:=0;
```

```
> for k from j-1 to 1 by -1 do
> candidat:=op(k,Ecole):
> if op(2,candidat)=i then Nbr:=Nbr+1:
> else
> fi:
> od:
> if Nbr>=op(i,N) then print(‘Le candidat eliminé est numéro‘=op(1,op(j,Ecole)));
> else print(‘Le candidat non eliminé ‘=op(1,op(j,Ecole)));NewEcole:=NewEcole,C[op(1,
> fi:
> od:
> print(‘Le choix de l’école est ‘=[NewEcole]);RETURN([NewEcole]);
> end:
```

On applique notre petit programme pour le 1ére école

```
> E[1]:=choix_ecole(1);
```

‘Le candidat non eliminé ‘ $= 4$
‘Le candidat non eliminé ‘ $= 2$
‘Le choix de l’école est ‘ $= [C_4, C_2]$
$E_1 := [C_4, C_2]$

On applique notre petit programme pour le 2éme école

```
> E[2]:=choix_ecole(2);
```

‘Le candidat non eliminé ‘ $= 2$
‘Le candidat non eliminé ‘ $= 1$
‘Le candidat non eliminé ‘ $= 3$
‘Le candidat non eliminé ‘ $= 4$
‘Le choix de l’école est ‘ $= [C_2, C_1, C_3, C_4]$
$E_2 := [C_2, C_1, C_3, C_4]$

On applique notre petit programme pour le 3éme école

```
> E[3]:=choix_ecole(3);
```

‘Le candidat eliminé est numéro‘ $= 3$
‘Le candidat eliminé est numéro‘ $= 2$
‘Le candidat non eliminé ‘ $= 1$
‘Le choix de l’école est ‘ $= [C_1]$
$E_3 := [C_1]$

Maintenant chaque candidat va éliminer toute école qu'elle juge inutile, c'est à dire pour l

```
> for i from 1 to nops(Ecoles) do
> liste_principale[i]:=[seq(op(j,E[i]),j=1..N[i])];
> od;
```

$$liste\_principale_1 := [C_4]$$
$$liste\_principale_2 := [C_2, C_1]$$
$$liste\_principale_3 := [C_1]$$

On s'inspirant de cet exemple on inverse les role écoles-candidats pour trouver les écoles à