REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Concours Nationaux d'Entrée aux Cycles de Formation d'Ingénieurs
Session 2012

الجمهورية التونسية
وزارة التعليم العالي والبحث العلمي

المناظرات الوطنية للدخول
إلى مراحل تكوين المهندسين
دورة 2012

Concours Toutes Options

Barème : EXERCICE : 5 points PROBLEME 1 : 6 points PROBLEME 2 : 9 points

Alternative de correction de l'épreuve d'informatique

Barème sur 100

EXERCICE 1 (20 / 100)

3 pts

1.
```
> f:=(i::posint,j::posint)->piecewise((i=j and i<5),1-(1/2)^(5-i),(i=5
and j=5),1,(i=j+1),(1/2)^(5-j),0);
```
ou bien
```
> f:=(i::posint,j::posint)->if (i=j and i<5)then 1-(1/2)^(5-i)
                elif (i=5 and j=5) then 1
                elif (i=j+1) then (1/2)^(5-j)
                else 0 fi;
```

1 pt

2.
```
> A:=matrix(5,5,f);
```
ou bien
```
A:=matrix(5,5);
for i to 5 do for j to 5 do A[i,j]:=f(i,j) od:od:
```

1 pt

3.
```
> with(linalg):
  > d:=det(A);
```

2 pts

4.
```
> L:=[eigenvals(A)];
```

1 pt

5.
```
> if d<>0 then AI:=inverse(A) else ERROR(`A non inversible`) fi;
```

2 pts (1 + 1)

6.
```
> jordan(A,'p'); evalm(p);
```

2 pts

7.
```
> I5:=diag(1$5);
```
ou bien
```
> I5:=Matrix(5,5,shape=identity);
```
ou bien
```
> g:=(i,j)->piecewise(i=j,1,0); I5:=matrix(5,5,g);
```
ou bien
```
> I5:=array(identity,1..5,1..5);
```

2 pts

8.
```
> P:=unapply(det(X*I5-A),X);
```

4 pts

9.
```
> equal(evalm(P(A),Matrix(5,5));
```
ou bien
```
> equal(evalm(P(A),matrix(5,5,0));
```
N.B: on accepte également les structures de programmation de Maple (boucles/procédure).

2 pts (1 + 1)

10.
```
> b:=Vector(5,[1,0,0,0,0]);
  > linsolve(A,b);
```

PROBLEME1 (30 / 100)

8 pts

1.

1ère méthode	2ème méthode sans création d'une liste intermédiaire

> Chgsgn:=proc(L::list)
local L1,i,cs;
L1:=[NULL]:
for i to nops(L) do
if L[i]<>0 then
L1:=[op(L1),L[i]];fi;od;
cs:=0;
for i to nops(L1)-1 do
if L1[i]*L1[i+1]<0 then
cs:=cs+1 fi;od;
return(cs);
end proc:
N.B: on peut travailler avec une séquence
L1:=NULL:
for i to nops(L) do
if L[i]<>0 then
L1:=L1,L[i];fi;od;
for i to nops([L1])-1 do
if L1[i]*L1[i+1]<0 then
cs:=cs+1 fi;od;
return(cs);
end proc:
	> Chgsgn:=proc(L::list)
	local n,i,k,cs;
	cs:=0;
	k:=1;
	n:=nops(L);
	while L[k]=0 and k<n do k:=k+1 od;
	j:=k+1;
	while j<=n do
	while L[j]=0 and j<=n do j:=j+1 od;
	if L[k]*L[j]<0 then cs:=cs+1;
	fi;
	k:=j;
	j:=j+1;
	od;
	return(cs);
	end proc;

8 pts

2.

```
> List_Poly:=proc(P::polynom)
local LP,i,n;
n:=degree(P,x);
LP:=P,diff(P,x):
for i from 2 to n do
 LP:=LP,-rem(LP[i-1],LP[i],x);od;
return([LP]);
end proc:
```

6 pts

3.

```
> Nb_Chgs:=proc(P::polynom, r::numeric)
local L,n,i;
L:=List_Poly(P);
n:=nops(L);
L:=[seq(subs(x=r,L[i]),i=1..n)]:
return(Chgsgn(L));
end proc:
```

4 pts

4.

```
> Nb_Racine_Simple:=proc(P::polynom, a::numeric, b::numeric)
 if subs(x=a,P)=0 then ERROR(a,"annulle P donner une autre borne"  );
elif subs(x=b,P)=0 then ERROR(b,"annulle P donner une autre borne"  );
else  NBCHGS(P,a)-NBCHGS(P,b) fi:
end proc;
```

4 pts

5.

```
> NB_racine_simple(P/gcd(P,diff(P,x)),a,b);
```

PROBLEME2 (50 / 100)

N.B: on accepte également la syntaxe **E** pour le passage par valeur et **S** ou **E/S** pour le passage par variable .

PARTIE I

1. **4 pts**

```
Fonction Taille (NMAX : entier) : entier
variable e : entier
DEBUT
        REPETER
             Ecrire("donner un entier compris entre 2 et ", NMAX )
             Lire (e)
             JUSQU'A   e > 1 ET  e <= NMAX  ET e mod 2 = 0
        Retourner (e)
FIN
```

2. **6 pts**

```
Fonction Code_Car (T : TABCAR , c : caractère) : booléen
variable ind , i : entier
DEBUT
        ind ← 0
        i ← 1
        TANT QUE  ind = 0 and i <= 26  FAIRE
        Si  T[i] = c  Alors  ind ← i  Sinon  i ← i + 1  Fin Si
        Fin TANT QUE
        RETOURNER (ind)
FIN
```

3. **5 pts**

```
Procédure Saisie_Msg (n : entier , VAR T1 : TABMSG)
variable i : entier
DEBUT
        POUR  i  de  1  à  n  FAIRE
             REPETER
               Ecrire(" donner la ",i, "ième lettre")
               Lire (T1[i])
            JUSQU'A   Code_Car (T , T1[i]) <> 0
        FIN POUR
FIN
```

4. **6 pts**

```
Procédure SAISIE_MATC (VAR Mc : MATC)
variable i , j , A : entier
DEBUT
        REPETER
           POUR  i  de  1  à  2  FAIRE
            POUR  j  de  1  à  2  FAIRE
                  REPETER
                     Ecrire(" donner la case d'indices ",i, ",",j , " de la matrice de chiffrement ")
                     Lire (Mc[i , j])
                     JUSQU'A Mc[i , j] > 0
             FIN POUR
        FIN POUR
           A ← Mc [1 , 1] * Mc[2 , 2] – Mc [1 , 2] * Mc[2 , 1]
           JUSQU'A A mod 2 = 1 ET A mod 13 <> 0
FIN
```

5. **8 pts**

```
Procédure CHIFFRER (n : entier , T1 : TABMSG , Mc : MATC , T : TABCAR , VAR T2 : TABMSG)
variable k , A , B , C , D : entier
DEBUT
      POUR  k  de  1  à  n-1  (pas = 2)  FAIRE
            A ← Code_Car (T , T1 [k])
            B ← Code_Car (T , T1 [k + 1])
            C ← (Mc [1 , 1] * A + Mc [1 , 2] * B) mod 26
            D ← (Mc [2 , 1] * A + Mc [2 , 2] * B) mod 26
            T2 [k] ← T[C]
            T2 [k + 1] ← T[D]
      FIN POUR
FIN
```

PARTIE II

6. **5 pts**

```
Procédure Init_Tab (VAR E : TABE)
variable n , k  : entier
DEBUT
      k ← 0
      POUR  i  de  1  à  26  FAIRE
          Si  i mod 2 = 1  Alors  Si i mod 13 <> 0  Alors  k ← k + 1
                                                          E [k] ← i  Fin Si

      FIN POUR
FIN
```

7. **8 pts**

```
Procédure Cree_Md (Mc : MATC ,  VAR Md : MATC)
variable k , i : entier
         E : TABE
DEBUT
      Init_Tab (E)
      k ← (Mc [1 , 1] * Mc [2 , 2] – Mc [1 , 2] * Mc [2 , 1])
       i ← 1
       TANT QUE  (k * E [i]) mod 26 <> 1  FAIRE
            i ← i + 1
       FIN TANT QUE
      Md [1 , 1] ← (E [i] * Mc [2 , 2]) mod 26
      Md [1 , 2] ← (E [i]  * - Mc [1 , 2]) mod 26
      Md [2 , 1] ← (E [i]  * - Mc [2 , 1]) mod 26
      Md [2 , 2] ← (E [i]  * Mc [1 , 1]) mod 26
FIN
```

8. **8 pts**

```
Procédure DECHIFFRER (n : entier , T2 : TABMSG , Mc : MATC , T : TABCAR ,  VAR T1 :
TABMSG)
variable k , A , B , C , D : entier
         Md : MATC
DEBUT
      Cree_Md (Mc , Md)
      POUR  k  de  1  à  n – 1  (pas = 2)  FAIRE
            A ← Code_Car (T2 [k])
            B ← Code_Car (T2 [k + 1])
            C ← (Md [1 , 1] * A + Md [1 , 2] * B) mod 26
            D ← (Md [2 , 1] * A + Md [2 , 2] * B) mod 26
            T1 [k] ← T[C]
            T1 [k + 1] ← T[D]
      FIN POUR
FIN
```