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

50 Cinquantenaire

الجمهورية التونسية
وزارة التعليم العالي والبحث العلمي والتكنولوجيا

المناظرات الوطنية للدخول إلى مراحل تكوين المهندسين
دورة 2009

Concours Nationaux d'Entrée aux Cycles de Formation d'Ingénieurs
Session 2009

Concours Toutes Options
Corrigé de l'épreuve d'Informatique

Barème : EXERCICE 1 : 4 points, EXERCICE 2 : 6 points, PROBLEME : 10 points

Le barème est sur 40

EXERCICE 1 *(8 points)*

1) (1 pt)

```
> f:=x->piecewise(x<-1,0,x<1,(x+1)/2,0);
```

Rq : On accepte pour cette question la définition avec la structure *if*.

```
> f:=x->if x<-1 then 0 elif x<1 then (x+1)/2 else 0 fi;
```

2) (0.5 pt)

```
> plot(f(x),x=-Pi..Pi);
```

3) (1 pt)

```
> an:=(1/Pi)*int(f(x)*cos(n*x),x=-infinity..infinity);
  bn:=(1/Pi)*int(f(x)*sin(n*x),x=-infinity..infinity);
```

4) (1 pt)

```
> a:=unapply(an,n);
  b:=unapply(bn,n);
```

5) (1 pt)

```
> limit(a(n),n=0);
  limit(b(n),n=0);
```

6) (1 pt)

```
> L1:=[seq([i,a(i)],i=1..20)];L2:=[seq([i,b(i)],i=1..20)];
```

Rq : On accepte pour cette question la création des deux listes L1 et L2 avec les structures itératives.

7) (1 pt)

```
> plot([L1,L2],style=point);
```

8) (1 pt)

```
> SF:=(x,m)->a(0)/2+sum(a(k)*cos(k*x)+b(k)*sin(k*x),k=1..m);
```

9) (0.5 pt)

```
> plot([SF(x,2),SF(x,20),f(x)],x=-Pi..Pi);
```

EXERCICE 2 *(12 points)*

1) (3 pts) (en-tête = 0.5 pt, initialisation de A1, C1 et P = 0.75 pt, traitement de calcul de P = 1.5 pt, retour du résultat = 0.25)

```
> calcul_pol:=proc(M::matrix,n::posint)
 local   A1,C1,P,Ai,Ci,i;
 A1:=M;
 C1:=-trace(A1);
 P:=x^n+C1*x^(n-1);
 for  i  from  2  to  n  do
    Ai:=multiply(matadd(A1,C1*diag(1$n)),M);
    Ci:=-trace(Ai)/i;
    P:=P+Ci*x^(n-i);
    A1:=Ai;C1:=Ci;
 od;
 return(P);
 end proc;
```

Rq : le type des paramètres et la déclaration des variables locales sont facultatifs.

2)

2.1) (0.5 pt)

```
> with(linalg);
```

2.2) (1 pt)

```
> f:=(i,j)->if i=j then 0 else 1 fi;
```

2.3) (1 pt)

```
> M:=matrix(4,4,f);
```

2.4) (1 pt)

```
> Id:=diag(1$n);
```

Rq : On accepte pour cette question la création de ***Id*** en utilisant la commande ***identity*** ou bien les structures itératives.

2.5) (1 pt)

```
> P1:=charpoly(M,x);
```

2.6) (1 pt)

```
> P2:=calcul_pol(M,4);
```

2.7) (0.5 pt)

```
> evalb(P1=P2);
```

3) (3 pts) (en-tête = 0.5 pt, test sur les degrés de P et Q = 0.75 pt, traitement de vérification = 1.5 pt, retour du résultat = 0.25)

```
> comparaison:=proc(P,Q::polynom)
 local   etat,i;
 if   degree(P)=degree(Q)   then   etat:=true
                            else   etat:=false   fi;
 i:=0;
 while   i<=degree(P) and etat=true   do
    if   coeff(P,x,i)<>coeff(Q,x,i)   then   etat:=false   fi;
    i:=i+1;
 od;
 return(etat);
 end proc;
```

PROBLEME *(20 points)*

1) (1 pt)

```
Procédure  saisie(variable w : réel)
Début
    Répéter
      Lire(w)
    Jusqu'à  w >= 0  ET  w < 1
Fin
```

2) (3.5 pts)

```
Procédure  convert(w : réel , variable Tw : TABGR)
variable  R : réel
          i, k : entier
Début
    Si  2*w < 1  Alors  Tw[1] ← 0
                 Sinon  Tw[1] ← 1
    Fin Si
    R ← w
    i ← 1
    Répéter
      R ← 2*R - Tw[i]
      i ← i+1
      Si  R = 0  Alors
         Pour  k  de  i  à  N  Faire
            Tw[k] ← 0
         Fin Pour
         Sinon
            Si  2*R < 1  Alors  Tw[i] ← 0
                         Sinon  Tw[i] ← 1
            Fin Si
      Fin Si
    Jusqu'à  R = 0  OU  i = N
Fin
```

3) (2.5 pts)

```
Procédure  plusgrand(Tx, Ty : TABGR) : booléen
variable  i : entier
          fini : booléen
Début
     plusgrand ← vrai
     fini ← faux
     i ← 0
     Tant que  fini = faux  ET  i < N  Faire
        i ← i + 1
        Si  Tx[i] > Ty[i]  Alors  fini ← vrai
           Sinon
           Si  Tx[i] < Ty[i]  Alors  fini ← vrai
                                     plugrand ← faux
           Fin Si
        Fin Si
     Fin Tant que
Fin
```

N.B : On accepte la conversion des deux tableaux *Tx* et *Ty* en deux réels *x* et *y* avec la formule donnée dans l'énoncé $x = \sum_{i=1}^{N} x_i 2^{-i}$.

4) (3.5 pts)

```
Procédure  foisdeux(Tx : TABGR , variable Ty : TABGR , variable tropgrand : booléen)
variable  i : entier
Début
     Si  Tx[1] = 1  Alors
        tropgrand ← vrai
        Sinon
        tropgrand ← faux
        Pour  i  de  1  à  N-1  Faire
           Ty[i] ← Tx[i+1]
        Fin Pour
        Ty[N] ← 0
     Fin Si
Fin
```

N.B : On accepte pas la conversion.

5)

5.1) (3.5 pts)

```
Procédure  itère(Tx,Ty : TABGR , variable Tz : TABGR , variable tropetit : booléen ,
                                  variable erreur : booléen)
Début
    foisdeux(Tx,Tz,erreur)
    Si  erreur = faux  Alors
        Si  plusgrand(Tz,Ty) = faux  Alors
            tropetit ← vrai
            Sinon
            tropetit ← faux
            difference(Tz,Ty,Tz)
        Fin Si
    Fin Si
Fin
```

5.2) (3.5 pts)

```
Procédure  divise(Tx,Ty : TABGR , variable Tz : TABGR , variable correct : booléen)
variable  i : entier
          tropetit,erreur : booléen
Début
    Si  Ty[1] = 1  OU  plugrand(Tx,Ty)  Alors
        correct ← faux
        Sinon
        correct ← vrai
        Pour  i  de  1  à  N  Faire
            itère(Tx,Ty,Tx,tropetit,erreur)
            Si  tropetit  Alors
                Tz[i] ← 0
                Sinon
                Tz[i] ← 1
            Fin Si
        Fin Pour
    Fin Si
Fin
```

6) (2.5 pts)

Si la contrainte $x<y<1/2$ n'est pas respectée, il faut tout d'abord décaler vers la droite les chiffres de l'écriture en base 2 de y jusqu'à ce que le résultat, appelons le y', soit strictement inférieur à 1/2 (Si y est de type TABGR, il est inférieur à 1 on ne décale donc son écriture que d'une position au plus). On décale ensuite les chiffres de l'écriture de x jusqu'à ce que le résultat, appelons-le x', soit strictement inférieur à y'. On calcule x'/y' à l'aide de la procédure précédente. Si on a décalé y de p positions et x de q positions, x/y est égal au résultat obtenu multiplié par 2^{q-p}.