BCPST	Informatique : D.S. n°1	Samedi 1 décembre

Vous devrez rendre 3 copies :

Une copie pour les exercices 1 et 2
Une copie pour l'exercice 3
Une copie pour l'exercice 4

Une partie du barème est donnée sur la syntaxe du langage Scilab et sur votre interprétation de l'algorithme nécessaire pour répondre à chaque problème. Vous devrez donc expliquer chaque programme par un texte en français.

Exercice 1 (3 points)
Exécuter le programme suivant pas à pas en affichant le contenu de chacune des variables en considérant les entrées au clavier a=13 et b=3.
Dire ensuite ce qu'il réalise (pour la variable *s* et à partir de *a* et de *b*)

```
a=input('le premier')
b=input('le second')
r=0;
while a<>0 do
if a/2==floor(a/2) then a=a/2; b=b*2;
else r=r+b; a=(a-1)/2;b=b*2;end
end
disp(r,'Le résultat est ')
```

Exercice 2 (3 points)
Le nombre 60 a pour carré 3600 ; si on enlève les deux derniers chiffres de ce carré, on obtient le nombre 36 qui est lui-même un carré ; ceci reste valable si on remplace 60 par n'importe quel nombre divisible par 10.
Le nombre 31 a pour carré 961 ; si on enlève les deux derniers chiffres de ce carré, on obtient le nombre 9 qui est lui-même un carré, on peut donc écrire : $31^2 = 3^2 \times 10^2 + 61$.
Ecrire un programme en scilab permettant de trouver tous les nombres entiers non multiples de 10 et inférieurs à 1 000, tels que si on enlève les deux derniers chiffres à leur carré, on obtient encore un carré.

Exercice 3 Croissance microbienne (7 points)
On étudie la croissance d'une population microbienne idéale, qui se reproduit sans contrainte d'environnement. Chaque microbe suit la règle suivante :

- à la première seconde, le microbe est très jeune,
- à la deuxième seconde, le microbe est jeune,
- à partir de la troisième seconde, le microbe est adulte,
- dès qu'un microbe est adulte, il produit un nouveau microbe à chaque seconde.

Pour cet exercice, on suppose qu'à la seconde 1 la population est constituée d'un unique microbe, très jeune.

1) Déterminer la population microbienne lors des 6 premières secondes.

2) On peut exprimer le nombre de microbes à la seconde *i* grâce à la suite de Fibonacci :

Fibonacci(1) = 1
Fibonacci(2) = 1
Fibonacci(i) = Fibonacci(i - 1) + Fibonacci(i - 2), pour $i \geqslant 2$

Ecrire un programme Scilab qui demande à l'utilisateur un entier *n* puis permet de calculer et d'afficher les valeurs de la suite de Fibonnaci pour les indices de 1 à *n*.

3) Transformer le programme précédent pour trouver au bout de combien de secondes la population microbienne compte plus d'un million d'individus.

4) Donner une nouvelle version du programme permettant de n'afficher que la taille de la population à une minute exactement ?

Exercice 4 (7 points)
1) Ecrire une fonction [x]=pgcd(a,b) permettant d'obtenir le pgcd *x* (Plus Petit Commun Diviseur) entre deux entiers *a* et *b*.

2) Ecrire une fonction [x]=premiers(a,b) déterminant si les deux entiers sont premiers entre eux ou non et renvoyant en ce sens une valeur de *x* égale à 0 ou 1 (ou mieux la valeur %t ou %f). Deux nombres entiers sont dits premiers entre eux si le pgcd de ces deux entiers est égal à 1.

3) Deux joueurs A et B jouent ensembles à un jeu :
Chacun des deux joueurs choisit un nombre entier au hasard entre 1 et 1000. Si les deux nombres sont premiers entre eux, c'est le joueur A qui gagne, sinon c'est B qui gagne. On veut savoir si le jeu est équitable.
a) A l'aide des fonctions précédentes, écrire une fonction []=partie(), effectuant une partie et affichant le nom du vainqueur.
b) Réécrire la fonction précédente de manière à jouer 100 parties et donner, en l'affichant, une première réponse à la question : " Le jeu est-il équitable?".

Correction du DS n°1

Exercice 1

r = 0 a =13 b = 3
r = 3 a = 6 b = 6
r = 3 a = 3 b = 12
r = 15 a = 1 b = 24
r = 39 a = 0 b = 48

Et le programme affiche Le résultat est : 39. Ce programme calcule le produit des deux nombres a et b (par la multiplication dite Russe ou Ethiopienne).

Exercice 2

```
 for n=11:999 do
 if n/10<>floor(n/10) then     x=floor(n^2/100);
                               if x==floor(sqrt(x))^2 then disp(n); end
 end
 end
```

Exercice 3

1) Notons TJ, J, A les états respectifs de chaque individu.
A la seconde 1: TJ
soit 1 individu
A la seconde 2 : J
soit 1 individu
A la seconde 3 : A, TJ
soit 2 individus
A la seconde 4 : A, TJ, J
soit 3 individus
A la seconde 5 : A, TJ, J, A, TJ
soit 5 individus
A la seconde 6 : A, TJ, J, A, TJ, A, TJ, J
soit 8 individus

2)
```
n=input('Donnez un entier plus supérieur à 1");
 u=1,v=1,
 for i=3:n do
        w=u+v;  disp(w); u=v;  v=w;
 end
```

3)
```
u=1, v=1, w=2;n=3;
 while w<1E6 do
     w=u+v;n=n+1; u=v;  v=w;
 end
 disp(n,'Le nombre de secondes est')
```

4)
```
u=1,v=1,
 for i=3:60 do
   w=u+v;  u=v;  v=w;
 end
 disp(w,'La population à 1 minute est ')
```

Exercice 4

function [x]=**pgcd**(a,b)

```
if a<b then c=a; else c=b; end
for i=1:c do
  if modulo(a,i)==0 & modulo(b,i)==0 then x=i; end
end
endfunction

function [x]=premiers(a,b)
if pgcd(a,b)==1 then x=%t; else x=%f; end
endfunction

function []=partie()
A=floor(1000*rand())+1;B=floor(1000*rand())+1;
if premiers(A,B)==%t then disp('Le gagnant est A'); else disp('Le gagnant est B'); end
endfunction

function []=partie2()
s=0
for i=1:100 do
A=floor(1000*rand())+1;B=floor(1000*rand())+1;
if premiers(A,B)==%t then s=s+1; end
end
if s==50 then disp('Le jeu semble équitable')
        elseif s>50 then disp('Le jeu semble favorable pour A')
                        else disp('Le jeu semble favorable pour B')
end
endfunction
```

Remarque : Bien entendu cette conclusion n'est pas définitive, il s'agit d'une expérience. Il faudrait la répéter pour étudier la fluctuation d'échantillonnage ou bien l'appliquer pour un nombre bien plus grand que 100 parties.

BCPST	Informatique : D.S. n°2	Samedi 22 mars

Vous devrez rendre 2 copies :

Une copie pour les exercices 1 et 2

Une copie pour les exercices 3 et 4

Une partie du barème est donnée sur la syntaxe du langage Scilab et sur votre interprétation de l'algorithme nécessaire pour répondre à chaque problème. Vous devrez donc expliquer chaque programme par un texte en français.

Exercice 1 (5 points)

1) Un nombre est dit automorphe s'il se termine par lui-même quand on l'élève au carré,

par exemple : $25^2 = 625$

Trouver tous les nombres automorphes inférieurs à 100.

2) Au triplet (a, b, c) de nombres réels on fait correspondre le triplet $(a+b, b+c, c+a)$ et on réitère le procédé.

Par exemple avec (2, -1, 0.5), on trouve successivement (1, -0.5, 2.5) (0.5, 2, 3.5) (2.5, 5.5, 4). Est-il possible, au bout d'un certain nombre d'opérations, de retrouver le triplet de départ ? Au bout de combien d'opérations ?

Construire un programme Scilab permettant, après avoir demandé à l'utilisateur les valeurs de a, de b et de c, d'afficher le nombre d'étapes nécessaire pour retrouver le triplet initial. Vous vérifierez que le programme s'arrête si plus de 500 étapes sont nécessaires.

Exercice 2 (5 points)

On rappelle ici que rand() fournit un nombre réel aléatoire dans [0;1[.

1) Le saut de puce

Une puce se déplace sur une droite en faisant un saut, soit en avant, soit en arrière. Au bout de 100 sauts, à quelle distance de son point de départ se trouve-elle?

On note x la position de la puce et on suppose qu'un saut lui permet de se déplacer de 1.

Construire un programme Scilab permettant de répéter 100 fois :

- choisir un nombre aléatoire
- permettre suivant sa valeur de décider si la puce avance ou recule
  (les deux premières étapes peuvent être réunies)
- actualiser la valeur de x

puis de répondre à la question initiale.

2) Le saut de kangourou

Un kangourou se déplace soit en faisant un saut vers le nord soit en faisant un saut vers l'ouest. Au bout de 100 sauts, à quelle distance de son point de départ se trouve-il ?

Construire un programme Scilab permettant de répondre à cette question. Il n'est pas interdit de faire "un peu" de géométrie.

Exercice 3 (5 points)

1) On considère sur l'ensemble des chaînes constituées de 0 et de 1 la transformation qui consiste à remplacer toute occurrence d'un '0' par la chaîne '01' et toute occurrence d'un '1' par la chaîne '10'. On définit la suite de Thue-Morse en partant de la chaîne '0' :

u_0 = '0'
u_1 = '01'
u_2 = '0110'
u_3 = '01101001'
u_4 = '0110100110010110'

Construire un programme Scilab permettant d'afficher le $20^{ème}$ terme de cette suite.

2) Facultatif, une application des suites de Thue-Morse :

On a $1 + 4 + 6 + 7 = 2 + 3 + 5 + 8$ et $1^2 + 4^2 + 6^2 + 7^2 = 2^2 + 3^2 + 5^2 + 8^2$.

Comment choisir les 8 termes de ces deux égalités : considérons le terme u_3 de la suite de Thue-Morse que nous superposons aux premiers entiers à partir de 1. Ceux désignés par 0 de la suite de Thue-Morse seront des termes du membre de gauche, les autres du membre de droite comme le montre le tableau ci-dessous :

u_3	*0*	*1*	*1*	*0*	*1*	*0*	*0*	*1*
Entiers =	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
A =	1			4		6	7	
B =		2	3		5			8

On peut toujours séparer les entiers compris entre 1 et 2*m* en deux classes *A* et *B* comportant le même nombre d'éléments de façon que la puissance $k^{ème}$ des éléments de *A* soit égale à la puissance $k^{ème}$ des éléments de *B*, pour tout $k < m$.

Ecrire un programme Scilab permettant de visualiser (puis de vérifier) la relation obtenue pour $k = 3$ et $m = 8$

Exercice 4 (5 points)

Il s'agit dans cet exercice de faire une étude statistique sur les mots ou les lettres employés dans un texte.

1) Construire la fonction [n]=mots(c) permettant de connaître le nombre de mots dans la chaîne *c*. On suppose que cette chaîne contient des lettres parmi celles de l'alphabet, sans accent, des espaces, des virgules ou des points. Pour ces deux derniers caractères, ils sont collés au mot précédent et un espace existe après eux excepté pour le point final.

2) Construire la fonction [m]=mot_moyen(c) permettant de connaître la taille moyenne d'un mot. Vous pourrez utiliser la fonction mot(c). Plusieurs méthodes sont envisageables.

3) Construire la fonction [m,M]=min_et_max(t) permettant de déterminer le minimum et le maximum d'un tableau t de taille quelconque.

4) Construire la fonction [t]=repartition_voyelles(c) permettant de construire le tableau t de taille 6 donnant le nombre de voyelles (*a, e, i, o, u, y*) contenues dans la chaîne c.

5) Construire la fonction []=frequence(c) permettant d'afficher la voyelle la plus fréquente ainsi que celle la moins fréquente contenues dans la chaîne *c*. Il n'est pas interdit d'utiliser les fonctions précédentes.

Correction du DS n°2

exercice 1

```
function []=automorphe()
for n=1:99
        q=n^2
        d=q-n
        if n<10 then, if d/10==floor(d/10) then, disp(n), end
                else, if d/100==floor(d/100) then, disp(n), end
        end
end

function [n]=triplet(a,b,c)
x=a+b, y=b+c, z=a+c
n=1
while n<501 & (x<>a | y<>b | z<>c) do
        u=x+y, v=y+z, w=x+z
        x=u, y=v, z=w
        n=n+1
end
```

exercice 2

```
function []=puce()
x=0
for i=1:100
        x=(-1)^int(2*rand())+x
end
 if x>0 then, disp('la puce a parcouru une distance de '+string(x)+'.')
        else, if x<0 then disp('la puce a reculé d''une distance de '+string(-x)+'.')
                        else, disp('la puce est revenu au départ')
                end
end

function []=kangourou()
p=0, q=0
for i=1:100
        y=int(2*rand())
        if y==0 then, p=p+1, else q=q+1, end
end
x=sqrt(p^2+q^2)
disp('le kangourou a parcouru une distance de '+string(x)+'.')
```

Exercice 3

1) Programme Scilab :

```
u='0';
for i=2:19 do
        d="";
        for j=1:length(u) do
                if part(u,j)=='0' then d=d+'01'; else d=d+'10'; end
        end
        u=d;
end
disp(u)
```

2) Soit $k = 3$ et $m = 8$; il faut donc $2m = 16$ caractères dans un des termes de la suite de Thue-Morse ce qui correspond à u_4.

A va contenir l'expression de gauche, *B* celle de droite,

a va contenir la valeur des termes de gauche, *b* celle de droite

```
A="";B="";a=0;b=0;u='0110100110010110';
for i=3:2:16 do
    if part(u,i)=='0' then A=A+"+"+string(i)+'^3';a=a+i^3;
                     else  B=B+"+"+string(i)+'^3';b=b+i^3;
    end
end
disp("A="+A)
disp("B="+B)
disp("Pour le membre de gauche, on trouve"+string(a))
disp("Pour le membre de gauche, on trouve"+string(b))
```

Exercice 4

1) On ajoute un espace en fin de chaîne pour avoir une situation semblable pour les rencontre du point dans toute la chaîne.
Le nombre de mots est alors le nombre d'espaces dans la chaîne.

```
function [n]=mots(c)
c=c+' ';
n=0
for i=1:length(c) do
    if part(c,i)==' ' then n=n+1;end
end
```

2) Tout d'abord, on cherche le nombre de mots et on construit un tableau tab qui parcourt la chaîne et qui remplit au fur et à mesure le tableau tab du nombre de caractères de chaque mot.

```
function [m]=mot_moyen(c)
tab=zeros(1,mots(c))
j=1;s=1;s1=s;  //on suppose que le premier caractère n'est pas un espace
for i=2:length(c) do
    if part(c,i)<>' ' | part(c,i)<>',' |part(c,i)<>'.' then if s==0 then tab(j)=s1;j=j+1;else s=s+1;s1=s;end
                                                          else s=0; //fin d'un mot
    end
end
somme=0;
for i=1:length(tab) do
    somme=somme+tab(i);
end
m=somme/length(tab)
```

```
function [m]=mot_moyen_plus_simple(c)
//il suffit de compter le nombre de mots, d'enlever à la longueur totale de c le nombre d'espaces,
//de points, de virgules pour en faire la moyenne.
n=mots(c);s=0;
for i=2:length(c) do
    if part(c,i)==' ' | part(c,i)==',' |part(c,i)=='.' then s=s+1;s1=s;end
end
m=(length(c)-s)/n;
```

BCPST	Informatique : D.S. n°3	Samedi 14 juin

Vous devrez rendre 2 copies :

Une copie pour les exercices 1 et 2

Une copie pour les exercices 3 et 4

Une partie du barème est donnée sur la syntaxe du langage Scilab et sur votre interprétation de l'algorithme nécessaire pour répondre à chaque problème. Vous devrez donc expliquer chaque programme par un texte en français.

Exercice 1 (Extrait, avec adaptation, de l'épreuve de mathématiques aux concours de cette année, 4 points)
Dans la fonction Scilab suivante, la fonction rand() renvoie un nombre réel aléatoire de l'intervalle [0,1[.

```
function [T]=tirage(n)
for i=1:n+1
    T(i)=1+floor(n*rand());
end
```

Puis le programme suivant est donné :

```
T=tirage(20000);
i=1;
coincide=0;
while coincide==0 do                    // début de la première boucle while
    i=i+1;
    S=0;
    while S<i-1 & coincide==0 do        // début de la deuxième boucle while
        S=S+1;
        if T(S)==T(i) then
            coincide=1;
        end
    end                                 // fin de la deuxième boucle while
end                                     // fin de la première boucle while
U=i;
for n=1:i do
    disp(T(n))
end
disp('U = ')
disp(U)
disp('S = ')
disp(S)
```

1) Que fait la fonction tirage ?
2) Que représentent les variables U et S à la fin du programme ?
3) Pourquoi est-il certain que le nombre de passages dans la première boucle while est fini ?

Exercice 2 (matrices de permutation, 6 points)
Une matrice de permutation est une <u>matrice carrée</u> qui vérifie les propriétés suivantes :

- les coefficients sont 0 ou 1
- il n'y a qu'un seul 1 par ligne
- il n'y a qu'un seul 1 par colonne

Ainsi la matrice $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ est une matrice de permutation.

Les matrices de permutations carrées de taille n sont en bijection avec les <u>permutations</u> de l'ensemble $\{1, 2, ..., n\}$. Si σ est une telle permutation, la matrice correspondante est A_σ de terme général a_{ij} tel que:

$$a_{i,j} = \delta_{i,j} = \begin{cases} 1 \text{ si } j = \sigma(i) \\ 0 \text{ sinon} \end{cases}$$

On démontre que la matrice transposée de A_σ est A_σ^{-1}.

1) Écrire une fonction TesterMatrice permettant de tester si une matrice M passée en argument est une matrice de permutation.

2) La trace de A_σ est égale au nombre d'entiers i tels que $\sigma(i) = i$, c'est-à-dire au nombre de points fixes de A_σ.
Ecrire une fonction TraceMatrice permettant de déterminer la trace de la matrice M.

3) Multiplier à gauche une matrice M par une matrice de permutation élémentaire revient à faire l'échange de deux lignes de M. Ecrire une fonction qui prend comme paramètre une matrice carrée M de taille n, demande à l'utilisateur les lignes i et j à transposer, puis construit la matrice de transposition élémentaire P telle que la matrice PM aient les lignes i et j échangées (par rapport à M).

Exercice 3 (4 points)

D'après les sondages, si des élections avaient lieu dans une province d'un certain pays aujourd'hui, 70% de ceux qui ont voté Libéral aux dernières élections voteraient encore Libéral, 20% voteraient Conservateur et 10% voteraient Néodémocrate. De la même façon, de ceux qui ont voté Conservateur, 10% voteraient Libéral, 80% Conservateur et 10% Néodémocrate et de ceux qui ont voté Néodémocrate, 30% voteraient Libéral, 30% Conservateur et 40% Néodémocrate.

Supposons que ces pourcentages demeurent constants d'une élection à l'autre et notons $X_n = \begin{pmatrix} l_n \\ c_n \\ d_n \end{pmatrix}$, $n \geqslant 0$, où l_n est le pourcentage de vote Libéral à la $n^{\text{ième}}$ élection, c_n est le pourcentage de vote Conservateur à la $n^{\text{ième}}$ élection et d_n est le pourcentage de vote Néodémocrate à la $n^{\text{ième}}$ élection.

Pour chacune des questions suivantes, vous donnerez uniquement les lignes de programme Scilab permettant de donner la réponse. Avec d'éventuels commentaires supplémentaires, permettant de les expliquer. Aucun calcul "à la main" est demandé.

1) Quelle est la matrice de transition T, telle que $X_{n+1} = T.X_n$?

2) Si aux dernières élections, les pourcentages du vote étaient 45%, 40% et 15% respectivement pour les Libéraux, les Conservateurs et les Néodémocrates, que seront-ils cette fois-ci ? A l'élection suivante ?

3) Ecrire une fonction Scilab permettant de déterminer à long terme, c'est-à-dire pour un grand nombre d'élections, quels seront les pourcentages du vote Libéral, Conservateur et Néodémocrate, en supposant que la matrice de transition est constante ?

Exercice 4 (6 points)

Soit l'algorithme suivant :
(1) entrer au clavier un entier naturel
(2) l'élever au carré
(3) ajouter au résultat le nombre initialement choisi
(4) à ce dernier résultat ajouter 41
(5) afficher ce dernier résultat

1) Ecrire un programme en Scilab permettant de réaliser cet algorithme.
(Vous devrez vous assurer que l'entier qui est proposé par l'utilisateur est effectivement positif.)

2) Le programme suivant consiste à vérifier si, au départ, on choisit un nombre entier compris entre 0 et 39, alors on obtient toujours un nombre premier.
Vous devez construire le programme en respectant les étapes successives qui sont décrites ci-après :
a) construire une boucle en la variable ***i*** permettant de parcourir les entiers entre 0 et 39.
b) reprendre la partie du programme réalisé en 1) permettant d'affecter à la variable ***x*** le résultat de l'algorithme pour la valeur initiale ***i***.
c) Tester si cet entier ***x*** est un nombre premier ou non.
(Un nombre est premier si, parmi tous les nombres entiers entre 1 et lui-même, il n'est divisible que par 1 et lui-même).
d) Vous compterez en la variable **somme** le nombre d'entiers donnant par cet algorithme un nombre premier. Suivant le résultat de cette variable, affichez si tous les nombres choisis entre 0 et 39 produisent un nombre premier.

3) Facultatif :
Cette propriété est vérifiée avec des nombres autres que 41. A l'instruction (4), en considérant un nombre Y au lieu de 41, on obtient des nombres premiers quand on choisit une valeur initiale comprise entre 0 et $Y - 2$.
Ces nombres Y sont appelés les nombres chanceux d'Euler (mathématicien suisse 1707-1783).
Ecrire un programme Scilab permettant de déterminer les nombres chanceux d'Euler inférieurs à 100. Il a été démontré en 1967 qu'il n'y en a pas d'autres que ceux proposés par Euler.

Correction du DS n°3

Exercice 1

1) La fonction tirage permet de construire un tableau de $n + 1$ valeurs contenant des entiers aléatoires entre 1 et n.

2) Les variables S et U représentent les deux premiers tirages dont les valeurs sont égales.

3) D'après le principe des tiroirs, il existe au moins deux valeurs égales dans le tableau. Il y a en effet plus de valeurs que de valeurs distinctes. Dons la boucle while en coincide==0 se terminera car il existera au moins deux valeurs identiques.

Exercice 2

1)
```
function [bool]=TesterMatrice(M)
bool=%t;n=sqrt(length(M));
for i=1:n do
        for j=1:n do
                if M(i,j)<>0 & M(i,j)<>1 then bool=%f;end
        end
end
for i=1:n do  //somme par ligne
        s=0;
        for j=1:n do
                s=s+M(i,j)
        end
        if s<>1 then bool=%f;end
end
for j=1:n do  //somme par colonne
        s=0;
        for i=1:n do
                s=s+M(i,j)
        end
        if s<>1 then bool=%f;end
end
```

2)
```
function [s]=TraceMatrice(M)
n=sqrt(length(M));s=0;
for i=1:n do
        s=s+M(i,j)
end
```

3) La matrice de transposition élémentaire échangeant la $i^{ème}$ ligne avec la $j^{ème}$ ligne est la matrice identité pour laquelle l'élément de position (i,i) est nul ainsi que l'élément (j,j). De plus les éléments de positions (i,j) et (j,i) sont égaux à 1.

```
function [M]=transposition(M)
n=sqrt(length(M));
i=input('Entrez le numéro de la première ligne');j=input('Entrez le numéro de la deuxième ligne');
P=zeros(n,n);
for k=1:n do P(k,k)=1;
P(i,i)=0;P(j,j)=0;P(i,j)=1;P(j,i)=1;
M=P*M;
```

Exercice 3

1)
```
T=[0.7 0.1 0.3;0.2 0.8 0.3;0.1 0.1 0.4]
```

2)
```
X=[0.45;0.4;0.15]
Y=T*X;disp(Y)
```

```
Z=T*Y;disp(Z)
```
3) On détermine deux évolutions à long terme et on regarde s'il y a une grande variation. On peut aussi déterminer le premier vecteur pour lequel la différence relative par rapport au précédent ne soit pas très grande.
```
T=[0.7 0.1 0.3;0.2 0.8 0.3;0.1 0.1 0.4];
X=[0.45;0.4;0.15];
Y=T*X;
d=abs((Y(1,1)-X(1,1))/X(1,1))+abs((Y(2,1)-X(2,1))/X(2,1))+abs((Y(3,1)-X(3,1))/X(3,1));
while d>0.01 do
X=Y;
Y=T*X;
d=abs((Y(1,1)-X(1,1))/X(1,1))+abs((Y(2,1)-X(2,1))/X(2,1))+abs((Y(3,1)-X(3,1))/X(3,1));
end
disp(X)
```

Exercice 4

1)
```
n=input('entrez un entier positif');
while n<0 | n<>floor(n) do
  n=input('entrez un entier positif !');
end
m=n^2;
m=n+m+41;
disp(m)
```

2)
```
somme=0;
for i=0:39 do
 x=i^2+i+41;
 s=0;
 for k=1:x do
  if floor(x/k)==x/k then s=s+1;end
 end
 if s==2 then somme=somme+1;end
end
disp(somme)
if somme==40 then disp('tous les entiers entre 0 et 39 produisent des entiers premiers');
          else disp('tous les entiers entre 0 et 39 ne produisent pas des entiers premiers');
end
```

3)
```
for n=1:100 do
somme=0;
for i=0:n do
 x=i^2+i+n+2;
 s=0;
 for k=1:x do
  if floor(x/k)==x/k then s=s+1;end
 end
 if s==2 then somme=somme+1;end
end
if somme==n+1 then disp(n);
end
end
```