BCPST	Informatique : D.S. n°1	Samedi 26 novembre

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 (Extrait modifié en Scilab de l'épreuve C, Mathématiques 2000, du concours "Agro") (4 points)

1) Dans le problème, la fonction suivante permettait de construire une simulation de variables aléatoires prenant un nombre fini de valeurs. On a p réel tel que $0 < p < 1$

a) Que fait la fonction suivante :

```
function [a]=MACHIN1(p);
Y=rand();
if Y<=p then a=1; else a=0; end
```

b) Quelle est la probabilité d'obtenir 1 après son exécution ?

2) On considère cette autre fonction du problème :

```
function [i]=TRUC1(n);
    x=floor(n*rand()) + 1;
    i=0; y=floor(n*rand()) + 1;
    while x==y & i <100 do
    i=i+1;
    y=floor(n*rand()) + 1;
    end
```

a) Que fait cette fonction ?

b) Quelle est la probabilité que i soit égal à 100 à l'issue de l'exécution de cette fonction ?

Exercice 2 (5 points)

1) Construire un programme Scilab permettant de déterminer si une année donnée au clavier est une année bissextile ou non suivant la règle qui est rappelée ci-dessous :

Si une année A n'est pas divisible par 4, l'année n'est pas bissextile. Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

2) Modifiez le programme précédent de manière à l'insérer dans un programme que vous écrirez permettant de réaliser les tâches suivantes :

- demande à l'utilisateur deux années extrêmes. Le programme Scilab devra vérifier que la deuxième année lue au clavier est strictement supérieure à la première. Sinon, l'utilisateur est prié de recommencer. la saisie jusqu'à ce qu'elle soit correcte.
- dans un deuxième temps, le programme affichera toutes les années bissextiles entre ces deux dates.

Exercice 3 (5 points)

1) On définit la suite $(x_n)n\in \mathbb{N}$ de la manière suivante : x_0 et $x_{n+1} = 4x_n(1 - x_n)$.

Ecrire un programme qui

-demande à l'utilisateur la valeur de x_0 et celle de n.

- calcule x_n en fonction des valeurs données ci-dessus.

Modifier celui-ci pour qu'il affiche les valeurs successives $x_0, x_1, \ldots, x_n$ de la suite.

2) Ecrire un programme qui calcule la somme des carrés des n premiers entiers impairs, n étant donné au clavier par l'utilisateur. Par exemple, $1^2 + 3^2 + 5^2 + 7^2 + 9^2 = 165$.

Exercice 4 Approche d'une racine carrée par Théon de Smyrne (6 points)

Méthode : Si $m \geqslant 1$ et $n \geqslant 1$ deux entiers et $n^2 = 2m^2 + 1$, on démontre que $\frac{n}{m} - \frac{1}{m} < \sqrt{2} < \frac{n}{m}$.

Si $m \geqslant 1$ et $n \geqslant 1$ deux entiers et $n^2 = 2m^2 - 1$, on démontre que $\frac{n}{m} < \sqrt{2} < \frac{n}{m} + \frac{1}{m}$.

Ainsi, plus m augmente, meilleure est l'approximation de $\sqrt{2}$ obtenue.

Si un couple $(x,y) = (n,m)$ vérifie l'une des deux égalités $^{(1)}$ $x^2 = 2y^2 + 1$ ou $^{(2)}$ $x^2 = 2y^2 - 1$, le couple $(x,y) = (2n + m\ ;\ n + m)$ vérifie l'autre.
En partant ainsi d'un couple qui convient (par exemple (2,3) puisque $3^2 = 2\times2^2 + 1$) on obtient un autre couple qui convient et dont le deuxième élément est plus grand (ici le couple (7,5)).
Les fractions $\frac{n}{m}$ obtenues sont alors des valeurs approchées par défaut ou par excès de $\sqrt{2}$ à $\frac{1}{m}$.

1) Construire le programme Scilab permettant de :

- demander à l'utilisateur la précision voulue pour le calcul d'une valeur approchée de $\sqrt{2}$.

Le programme devra vérifier que la précision donnée est strictement plus petite que 1 sinon, l'utilisateur est prié de recommencer

- Partir du couple (2,3) donné dans la méthode ci-dessus pour obtenir les couples suivants et afficher les valeurs approchées successives donnée par la méthode de Théon de Smyrne. (vous pourrez simuler quelques étapes à la main de la méthode avant de construire le programme).

2) On souhaite désormais obtenir une valeur approchée de $\sqrt{3}$. Par analogie avec la méthode précédente, il semble que les valeurs approchées successives sont initialement obtenues par l'équation $n^2 = 3m^2 + 1$.

a) Construire un programme permettant de trouver un couple de deux entiers compris entre 1 et 10 solutions.

Rechercher sur votre copie, à la main ou avec une calculatrice un tel couple solution.

b) A partir de ce couple solution, construire un autre programme permettant de déterminer une valeur approchée de $\sqrt{3}$ à 10^{-4} près à l'aide de la méthode de Théon de Smyrne.

Correction du DS n°1

Exercice 1

1) a) rand() permettant d'obtenir un nombre aléatoire entre 0 et 1 (plus précisément dans [0;1[), la fonction machin renvoie la valeur 1 si le tirage est inférieur à *p*, 0 sinon.
Cette fonction correspondant à la simulation d'une variable aléatoire de Bernoulli.
b) La probabilité d'obtenir 1 est donc égale à *p*.
2) a) *x* = floor(*n*×rand()) + 1 permet d'obtenir un nombre entier quelconque entre 1 et *n*.
La boucle en while permettant d'incrémenter *i* de 1 en 1 et cela pour les valeurs de *i* entre 0 et 99, cette boucle est réalisée au maximum 100 fois. Le second test d'exécution concerne la nécessité que le tirage de *x* soit le même que celui de *y*.
Cette boucle compte ainsi dans *i* le nombre de fois que le tirage de *y* est successivement égal au tirage de *x* donné initialement.
b) Pour que *i* soit égal à 100, il faut que le tirage de *y* coïncide 100 fois avec celui de *x* donc avec la probabilité de $\frac{1}{n^{100}}$

Exercice 2

1) le programme repose sur deux tests :

```
n=input('entrez une année');
if n/4-floor(n/4)<>0 then disp('cette année n''est pas bissextile')
elseif (n/100==floor(n/100)) & (n/400<>floor(n/400)) then disp('cette année n''est pas bissextile')
    else disp('cette année est bissextile');  end
```

2) Il est possible d'utiliser la négation du test précédent pour ne considérer que les années bissextiles :

```
i1=input('entrez une première année');
i2=input('entrez une deuxième année plus grande que la première');
while i1>=i2 do
i1=input('entrez une première année');
i2=input('entrez une deuxième année plus grande que la première');
end
disp('les années bissextiles concernées sont :')
for n=i1:i2 do
if n/4-floor(n/4)==0 & ((n/100<>floor(n/100)) | (n/400==floor(n/400))) then disp(n); end
end
```

Exercice 3

1) Il s'agit du principe classique de cacul des différents termes d'une suite :

```
x0=input('entrez le terme initial de la suite')
n=input('entrez l''indice n du terme recherché')
x=x0;
for i=1:n do x=4*x*(1-x); end
disp(x)
```

Pour la seconde version, seul le placement du disp apporte la solution ou bien de permettre les affichages successifs des différents calculs :

```
x0=input('entrez le terme initial de la suite')
n=input('entrez l''indice n du terme recherché')

x=x0
for i=1:n do x=4*x*(1-x);
disp(x)
end
```

2) Vous deviez reconnaître la construction d'une somme :

```
n=input('combien de nombres impairs souhaitez-vous considérer ?')
s=0;
for i=0:(n-1) do
s=s+(2*i+1)^2;
end
disp(s)
```

Exercice 4

1) Les boucles en while sont utilisées ; l'une pour la demande de la précision l'autre pour le calcul des termes successifs :

```
pre=input('Quelle précision voulez-vous ?");
while pre>=1 do
pre=input('La précision doit être plus petite que 1");
end
n=3;m=2;
while 1/m>pre do
  x=n+2*m;
  y=n+m;
  n=x;m=y;
  disp(n/m)
end
```

2) a) Deux boucles en for pour rechercher les couples solutions :

```
for i=1:10 do
  for j=1:10 do
     if i^2==3*j^2+1 then disp(i,' ',j)
     end
  end
end
```

b) Le programme est très semblable au premier :

```
pre=1.E-4
n=2;m=1;
while 1/m>pre do
  x=n+3*m;
  y=n+m;
  n=x;m=y;
  disp(n/m)
end
```

BCPST	Informatique : D.S. n°2	Samedi 18 mars

Vous devrez rendre 2 copies : Une copie pour l'exercice 1
Une copie pour l'exercice 2

Exercice 1

Le but de cet exercice est de travailler sur une chaîne de caractères représentant un texte. Vous supposerez que ce texte contient des caractères en majuscules, des espaces, des virgules, des points, des apostrophes.

1) Construire la fonction [tab]=occurrences(c) déterminant dans un tableau tab le nombre d'occurrences de chacune des 26 lettres de l'alphabet contenues dans la chaîne c. (Rappel : str2code('A') retourne -10 dans Scilab ainsi que str2code('Z') retourne -35).

2) Construire la fonction [a,b]=bornes(tab) déterminant le minimum et le maximum des occurrences du tableau tab donné en entrée.

3) Construire la fonction [tab_bis]=trier(tab) permettant de construire le tableau tab_bis, tableau associé à un tri croissant du tableau tab. Ce tableau ne doit pas contenir les valeurs dans l'ordre croissant du tableau tab mais les positions de ces valeurs données dans l'ordre croissant.

Exercice 2

Un générateur de Charabia latin

Nous nous proposons de réaliser un générateur de charabia latin (jeu utilisé par les petits enfants de langue anglaise), c'est-à-dire un programme qui modifie un mot du français en un mot d'un charabia latin. Cette transformation s'effectue en plaçant la première lettre du mot à la fin, et en y ajoutant la lettre "a". Ainsi, le mot "tortue" devient "ortueta", "Scilab" devient "cilabsa", et ainsi de suite.

Ecrivons maintenant un programme Scilab qui lira une phrase en français et écrira son équivalent en charabia latin. Pour des raisons de simplicité, nous ne tiendrons pas compte du problème des lettres majuscules et des signes de ponctuation.

L'écriture du programme traduisant le fonctionnement de ce jeu sera facilitée par la construction des modules suivants :

1) Ecrire la fonction **[ch]=Entrez_Texte** permettant d'entrer le texte en français représenté par la chaîne de caractères **ch** (vérifiez que le texte comporte moins de 80 caractères).

2) Ecrire la fonction **[n]=Nombre_mots** permettant de déterminer le nombre **n** de mots de la phrase **ch** en utilisant le nombre d'espaces.

3) Ecrire la fonction **[m]=recherche_mot(ch,n)** permettant d'affecter à la chaîne **m** le **n**$^{\text{ième}}$ mot de la phrase **ch**.

4) Ecrire la fonction **[ma]=Transf_Franc_Charabia(mf)** permettant de transformer le mot français de la chaîne **mf** en son équivalent en charabia latin dans la chaîne **ma**.

5) Ecrire le programme complet permettant de transformer une phrase complète en son équivalent en charabia latin.

Correction du DS n°2

Exercice 1

```
function [tab]=occurrences(c)
tab=zeros(1,26);
for i=1:length(c)
        j=str2code(part(c,i))
        if j>=-35 & j<=-10 then,tab(-9-j)=tab(-9-j)+1, end
end
function [a,b]=bornes(tab)
a=tab(1); b=tab(1)
for i=2:length(tab)
        if tab(i)<a then, a=tab(i), else, if tab(i)>b then, b=tab(i), end, end
end

function [tab_bis]=trier(tab)
[a,b]=bornes(tab); tab_bis=zeros(1,26); k=a, j=1
while j<=26
        for i=1:26 do if tab(i)==k then, tab_bis(j)=i, j=j+1, end; end
        k=k+1
end
```

Exercice 2

```
function [ch]=Entrez_texte
n=100
while n>80 do
        ch=input('Entrez un texte de moins de 80 caractères',"string");
        n=length(ch)
end

function [n]=Nombre_Mots(ch)
k=length(ch); s=0
for i=1:k do
        if part(ch,i)==' ' then s=s+1; end
end;n=s+1;

function [m]=Recherche_Mot(ch,n)
k=Nombre_Mots(ch)//si il n'y a pas assez de mots on renvoie la chaîne vide
if n>k then m=''; else    tab=1:k-1;j=1;
                          for i=1:length(ch) do
                          //on range dans le tableau tab les positions des blancs de la chaîne ch
                                  if part(ch,i)==' ' then tab(j)=i;j=j+1; end
                          end
                          if n==1 then m=part(ch,1:tab(1)-1); end
                          if n==k then m=part(ch,tab(k-1)+1:length(ch)); end
                          if (n<>1)&(n<>k) then m=part(ch,tab(n-1)+1:tab(n)-1); end
                  end

function [ma]=Transf_Franc_Charabia(mf)
ma=part(mf,2:length(mf))+part(mf,1)+'a'

function programme
chf=Entrez_texte(); n=Nombre_Mots(chf); cha=''
for i=1:n do
        mof=Recherche_Mot(chf,i)
        moa=Transf_Franc_Charabia(mof)
        cha=cha+moa+' '
end
disp(cha)
```

BCPST **Informatique : D.S. n°3**

Samedi 17 juin

Vous devrez rendre 3 copies : **Une copie pour l'exercice 1**
Une copie pour l'exercice 2
Une copie pour l'exercice 3

Exercice 1

1) Conjecture d'Erdös

Pour tout entier $n > 1$, il existe trois entiers x, y et z tels que :

$$\frac{4}{n} = \frac{1}{x} + \frac{1}{y} + \frac{1}{z}$$

Construire un programme permettant de vérifier cette conjecture pour $n \in \{1; 2; \ldots; 100\}$, dans ce cas, vous supposerez que les entiers x, y et z sont à rechercher parmi les entiers entre 1 et n^2.
S'il existe un entier pour lequel l'égalité n'est pas vérifiée, vous ferez afficher "La conjecture est fausse" sinon vous ferez afficher "La conjecture semble vraie".

Remarque : *C'est un cas particulier de fractions égyptiennes où l'on cherche à écrire un rationnel comme somme d'un nombre donné d'inverses d'entiers, conjecture vérifiée pour* $n \leqslant 10^8$.
De même existe la conjecture de Sierpinski qui suppose que pour $n > 1$, *il existe trois entiers naturels a, b et c tels que* $\frac{5}{n} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c}$.

2) La loi binomiale

Ecrire la fonction [t]=binomiale(n,p) en Scilab permettant de construire le tableau t contenant l'ensemble des valeurs de la loi binomiale de paramètres (n,p) où n est un entier et p un réel de]0;1[.

Exercice 2 Le nombre d'Or

Le nombre d'Or, noté φ, représente une proportion idéale pour certains. Il vérifie, entre autres, la relation suivante :

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ldots}}}}}$$

on peut définir une suite (x_i) de valeurs approchées de φ, convergeant vers φ, de la façon suivante :

$x_1 = 1$

$x_2 = 1 + \frac{1}{1} = 2$

$x_3 = 1 + \cfrac{1}{1 + \frac{1}{1}} = 1,5$

$x_4 = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{1}}} = 1,666\ldots$

Ecrire une fonction en Scilab qui calcule une approximation de φ en retournant le $n^{\text{ième}}$ élément de cette suite. La fonction recevra donc comme paramètre un entier n.

Exercice 3

1) Extrait de l'épreuve de la banque Agro-Véto, Mathématiques A, 2006

a) Ecrire un algorithme fournissant le produit de deux matrices appartenant à $M_4(\mathbb{R})$, l'algèbre des matrices carrées d'ordre 4 à coefficients dans $\mathbb{R}$.

b) Combien d'opérations (additions et multiplications) sont-elles nécessaires ?

2) Un centre de conditionnement physique pense installer une piscine à l'intérieur de ses murs. Un sondage auprès des membres actuels indique que tous les membres continueront à utiliser la salle de conditionnement physique et que 65% des membres utiliseront aussi la piscine. Parmi les nouveaux membres qui devraient s'inscrire au centre une fois la piscine installée, 78% utiliseront la piscine et 59% utiliseront la salle de conditionnement physique. Le centre compte présentement 440 membres.

a) Créer la matrice de transition A qui décrit l'utilisation prévue de la salle de conditionnement physique et de la piscine en fonction des anciens et des nouveaux membres une fois la piscine ouverte.

b) En supposant, uniquement pour cette question, que 200 nouveaux membres s'inscriront, écrire les instructions Scilab utilisant la matrice A pour afficher le nombre de personnes utilisant la salle de conditionnement physique ainsi que le nombre de personnes utilisant la piscine.

c) La piscine ne sera construite que si au moins 600 personnes s'y inscrivent tout en gardant 800 personnes pour la salle de conditionnement physique.

Ecrire les instructions Scilab nécessaires pour afficher le nombre de nouveaux membres à inscrire pour réaliser cette fréquentation de la piscine.

Correction du DS N°3

Exercice 1

1) function []=erdos()

```
n=2; t=%t
while t==%t & n<=100
        a=4/n
        test=%f
        x=1
        while test==%f & x<=n^2
                b=a-1/x
                y=1
                while test==%f & y<=n^2
                        c=b-1/y
                        z=1
                        while test==%f & z<=n^2
                                if c==1/z then, test=%t, else, z=z+1, end
                        end
                        if test==%f then, y=y+1, end
                end
                if test==%f then, x=x+1, end
        end
        if test==%t then,disp('pour n= '+string(n)+': '+string(x)+'-'+string(y)+'-'+string(z)),n=n+1,else t=%f, end
end
if n>100 then, disp ('la conjecture semble vraie'), else, disp('la conjecture est fausse'), end
```

2) function [t]=binomiale(n,p)

```
t=zeros(n+1)
for k=0:n do
//calcul des combinaisons par la formule
//k_parmi_n=n/k*(n-1)/(k-1)/...(n-(k-1))/(k-(k-1))
        pro=1
        for i=0:k-1 do
        pro=pro*(n-i)/(k-i)
        end
        t(k+1)=pro*(p^k)*((1-p)^(n-k))
end
```

Exercice 2

```
function [x]=nombre_d_or(n)
x=1
for i=1:n-1 do
        x=1+1/x
end
```

Exercice 3

function [C]=exo3_a(A,B)

```
//a)
C=zeros(l,c)
for i=1:4 do
        for j=1:4 do
                for k=1:4 do
                        C(i,j)=C(i,j)+A(i,k)*B(k,j)
                end
        end
end

//b)
//Il y a 4*4*4 boucles et chacune de ces boucles contient une addition et une multiplication
```

//donc en tout il y a 64+64=128 opérations de type addition et multiplication.

function exo3_b()

```
//a)
A=[1 0.59;0.65 0.78]

//b)
B=[440;200];
C=A*B
disp('il y a '+string(C(1,1))+' personnes utilisant la salle')
disp('il y a '+string(C(2,1))+' personnes utilisant la piscine')

//c)
C=[800;600]
B=A\C; disp(B)
disp('il faut '+string(floor(B(2,1))+1)+' nouveaux membres')
```