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

Vous devrez rendre 3 copies : Une copie pour les exercices 1 et 2
Une copie pour les exercices 3 et 4
Une copie pour l'exercice 5

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)
Voici un programme écrit en Scilab :

```
a=input('Entrez un premier entier');
b=input('Entrez un second entier');
while b<>0 do
        r=modulo(a,b);
        a=b;
        b=r;
    end;
disp(a);
```

1) Faire tourner le programme précédent en prenant A=12 et B=21
2) De même en prenant A=15 et B=20
Que se passe-t-il ?
3) Que fait ce programme ?

Exercice 2 (4 points)
Le problème est d'afficher tous les nombres de trois chiffres qui sont égaux à la somme du cube de leurs chiffres. Ces nombres sont les nombres d'Armstrong.
Exemple : $153 = 1^3 + 5^3 + 3^3$.

Après avoir correctement expliqué les variables utilisées et leur rôle, écrire **un algorithme** :
- qui fait parcourir un entier parmi les entiers à trois chiffres.
- qui calcule la somme du cube de ses chiffres et affiche les nombres solutions.

Exercice 3 : (Extrait modifié en Scilab de l'épreuve C, Mathématiques 2000, du concours "Agro") (3,5 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 procédure ?
b) Quelle est la probabilité que i soit égal à 100 à l'issue de l'exécution de la procédure ?

Exercice 4 (3,5 points)
1) Après avoir expliqué votre méthode, écrire un programme Scilab permettant de déterminer la somme des diviseurs d'un nombre entier demandé à l'utilisateur.
2) Transformer l'algorithme précédent de manière à trouver les paires de *nombres amis* entre 1 et 2000.

Deux nombres *m* et *n* sont dits *amis* si la somme des diviseurs de *m* (autres que *m*) est égale à *n* et la somme de tous les diviseurs de *n* (autres que *n*) est égale à *m* ; par exemple 220 et 284 sont des *nombres amis*.

Exercice 5 (6 points)
1) Ecrire une fonction **[c]=division(a,b)** permettant de déterminer si la division des deux entiers a par b est un nombre entier ou non. On pourra renvoyer c=%t dans l'affirmative et c=%f sinon.
2) L'algorithme de Göbel :
Soit (u_n) la suite définie par son premier terme $u_1 = 2$ et la relation de récurrence :

$$u_n = \frac{u_{n-1}(u_{n-1} + n - 1)}{n} \text{ pour } n \geqslant 2.$$

Utilisez la fonction précédente pour déterminer si tous les termes de la suite sont des nombres entiers. Pour cela :

a) Vous demanderez à l'utilisateur le nombre de termes à vérifier

b) Dès qu'un terme de la suite n'est pas entier, vous devrez afficher "Un terme n'est pas entier" et vous donnerez son rang.

c) Si tous les termes demandés sont des entiers, vous afficherez le message "Tous les termes de la suite jusqu'au rang demandé sont des nombres entiers.".

BCPST	Informatique : D.S. n°2	Samedi 12 mars

Vous devrez rendre 3 copies : **Une copie pour l'exercice 1**
Une copie pour les exercices 2 et 3
Une copie pour l'exercice 4

Exercice 1 Jeu de Nim

Le jeu se joue à deux : toi (l'élève) contre l'ordinateur (le prof). C'est chacun à son tour de jouer.
Le tapis comporte une seule rangée de 21 allumettes.
Lorsque c'est à ton tour de jouer, tu dois enlever une, deux ou trois allumettes.
Celui qui retire la dernière allumette perd la partie.

Il s'agit de te laisser jouer le premier et de te faire perdre à tous les coups en construisant la fonction []=Nim() suivante :
tant que la partie n'est pas finie

- afficher le message 'Il reste xxxx allumettes, vous devez en retirer entre 1 et 3' où xxxx représente le nombre d'allumettes restantes.
- demander à l'utilisateur un nombre entre 1 et 3. Redemander ce nombre s'il ne se trouve pas entre 1 et 3.
- l'ordinateur doit enlever à son tour des allumettes (il faut à cet instant enlever le complément à 4 d'allumettes, si vous enlevez1 allumette, l'ordinateur en enlève 3, si vous en enlevez 2 l'ordinateur 2 et si vous en enlevez 3 l'ordinateur en enlève 1).
- S'il ne reste plus d'allumette après votre tour de jeu, afficher : L'ordinateur à gagné. Si c'est après le tour de l'ordinateur (???), afficher : L'ordinateur a perdu.

Exercice 2 Le carré de Polybe

Polybe est un historien grec (env. 200 - 125 av. J.-C.) qui est à l'origine du premier procédé de chiffrement par substitution. C'est un système de transmission basé sur un carré de 25 cases (on peut agrandir ce carré à 36 cases, afin de pouvoir ajouter les chiffres ou pour chiffrer des alphabets comportant davantage de lettres, comme l'alphabet cyrillique):

	1	2	3	4	5
1	a	b	c	d	e
2	f	g	h	i	j
3	k	l	m	n	o
4	p	q	r	s	t
5	u	v	x	y	z

En français, on supprime le W, qui sera le cas échéant remplacé par V. En anglais, on agrège le I et le J.
Chaque lettre peut être ainsi représentée par un groupe de deux chiffres: celui de sa ligne et celui de sa colonne. Ainsi "e"=15, "u"=51, "n"=34, ...Polybe proposait de transmettre ces nombres au moyen de torches. Une torche à droite et cinq à gauche pour transmettre la lettre "e" par exemple. Ce procédé permettait donc de transmettre des messages sur de longues distances. On peut aussi transmettre les coordonnées des lettres en tapant des coups sur un mur, sur la tuyauterie, etc.

Construire la fonction **[ch]=carre_polybe(c)** qui effectue les tâches suivantes :
- Construction d'un tableau t qui est le tableau de type 5×5 ci-dessus.
- Pour une chaîne de caractère c de la forme correspondant à l'exemple c='1.3 3.5 1.4 1.5 4.3', la transformer, à l'aide du tableau t, en la chaîne de caractères ch qui correspond au codage par le carré de Polybe.

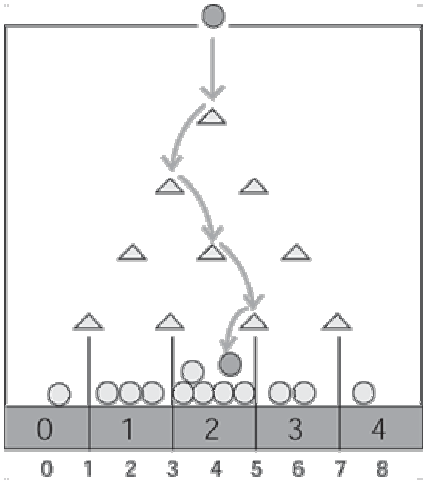
Exercice 3

Les nombres 46 et 96 possèdent une propriété intéressante : leur produit ne change pas lorsqu'on permute leurs chiffres.

En effet 46*96 = 4416 = 64*69.

On demande d'établir s'il existe d'autres couples de nombres de deux chiffres possédant la même propriété en les trouvant tous.

Exercice 4 La table de Galton



Il s'agit dans cet exercice de simuler à *n* reprises l'expérience suivante :

Une bille est lâchée au dessus du système ci-contre. Elle peut aller avec une chance sur deux à gauche ou à droite du premier triangle, elle peut ensuite, toujours avec une chance sur deux, aller à gauche ou à droite des triangles des rangées inférieures.

Il s'agit de dénombrer, après *n* lâchés de billes le nombre de billes dans chacune des cases 0 ou 1 ou 2 ou 3 ou 4.

On peut envisager le tableau de taille 5×9 suivant dont les * indiquées seront les positions possibles des billes

				*				
			*		*			
		*		*		*		
	*		*		*		*	
*		*		*		*		*

Considérez la variable *x* égale à 1 (première ligne) et *y* à 5 (5ème colonne).

Suivant le résultat d'un tirage aléatoire, la variable *x* sera augmentée de 1 et la variable *y* sera augmentée ou baissée de 1. En répétant ce tirage aléatoire trois autres fois, vous aurez ainsi simulé une expérience.

Un tableau de taille 1×9 correspondant à la dernière ligne recevra alors le résultat de la position finale.

Construire une fonction **[]=Galton();** réalisant les instructions suivantes :

- Demande à l'utilisateur du nombre de simulations à réaliser.
- Simulation des expériences
- Tracé de la courbe associée aux résultats (nombre de billes dans un numéro en fonction de ce numéro).

BCPST	Informatique : D.S. n°3	Samedi 4 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 (7 points)
1) A l'aide d'un programme Scilab, déterminer le nombre à quatre chiffres *N*, s'écrivant en base 10 sous la forme $\overline{abcd}$ tel que l'addition suivante soit vérifiée :

```
  b d a c
+ d c b a
---------
  a b c d
```

2) On remarque que $19^2 + 89^2 = 8282$.
Ecrire un programme Scilab déterminant la quantité de nombres bégues (à 4 chiffres de la forme *abab*) égaux à la somme de deux carrés, en dehors de 8282 ?

Exercice 2 Le jeu du pendu (7 points)
Le problème est de réaliser en langage Scilab un ensemble de fonctions permettant de jouer au jeu du pendu. Il s'agit dans un premier temps de demander à un utilisateur la chaîne à découvrir, d'afficher ensuite ses caractères extrêmes et de demander à un autre utilisateur les lettres nécessaires pour compléter les caractères manquants. Un compteur donnant le nombre de coups sera construit.
Exemple :
On donne à Scilab la chaîne ch="Bien_le_bonjour". Celui-ci affiche alors "B---_--_------r". Scilab demande alors les caractères les uns après les autres en affichant au fur et à mesure l'état de la chaîne à rechercher. Si l'utilisateur propose le caractère "o", le second affichage sera "B---_--_-o-o-r".

1) Ecrire une fonction **[ch]=saisie()** permettant la saisie d'une chaîne de caractères vérifiant les conditions suivantes :
→ elle ne doit être composée que de majuscules ou de minuscules (rappel : les codes ascii obtenus par l'instruction ascii("…") des majuscules sont entre 65 et 90 et ceux des minuscules vont de 97 à 122) ou du caractères _ (souligné) de code ascii 97.
→ Le premier et le dernier caractère ne doivent pas être le caractère _ (souligné).
2) Ecrire une fonction **[]=devoiler(ch)** qui consiste à dévoiler au fur et à mesure la chaîne :
→ construire un tableau de même longueur que la chaîne, coder par 0 les caractères cachés et par 1 les caractères visibles (au début du jeu, le premier et le dernier caractère sont codés par ainsi que tous les _ (soulignés)).
→ tant qu'il reste un caractère caché, demander à l'utilisateur un caractère, regarder s'il est dans la chaîne. Si c'est le cas, coder sa place (ou ses places) dans le tableau par 1, augmenter le compteur du nombre de tentatives de 1 et enfin afficher la chaîne trouvée à ce moment du jeu.

Exercice 3 Un problème d'endémie (7 points)
Un individu vit dans un lieu où il est susceptible d'attraper une maladie par piqûre d'insecte. Il peut être dans l'un des trois états suivants : immunisé (I), malade (M), non malade et non immunisé (S).
D'un mois à l'autre, son état peut changer suivant les règles suivantes :
→ étant immunisé, il peut le rester avec une probabilité 0,9 ou passer à l'état S avec une probabilité 0,1.

→ étant dans l'état S, il peut le rester avec une probabilité 0,5 ou passer à l'état M avec une probabilité 0,5.
→ étant malade, il peut le rester avec un probabilité 0,2 ou passer à l'état I avec une probabilité 0,8.
Les questions suivantes devront être traitées sous Scilab et les réponses formulées devront l'être aussi sous Scilab (même si vous pouvez expliquer votre démarche par un texte clair).

1) Ecrire la matrice de transition (on considèrera respectivement 1, 2 et 3 les états I, M et S), que lon notera A .
2) On suppose qu'au départ un individu est immunisé. Calculer A^2 et en déduire la probabilité que cet individu :
→ soit malade au bout de 2 mois.
→ soit immunisé au bout de 2 mois.
3) Calculer la probabilité pour que, au bout de 1 an, de 2 ans, de 3 ans, un individu soit malade dans chacun des trois cas suivants :
→ au départ, il est immunisé,
→ au départ, il est on malade et non immunisé,
→ au départ, il est malade.
(ceux d'entre vous qui ont une calculatrice et qui ont le temps de mettre en place les calculs pourront faire une remarque).
4) Comment doit être initialement un individu pour qu'il soit certain d'être malade au bout de 1 an ?
(ceux d'entre vous qui ont une calculatrice et qui ont le temps de mettre en place les calculs pourront faire une remarque).