BCPST	Informatique : D.S. n°1	Samedi 17 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

Exercice 1 (3,5 points)
Exécuter le programme suivant pas à pas en affichant le contenu de chacune des variables.
Dire ensuite ce qu'il réalise (pour la variable *p*)

```
-->p=1;i=1;
-->while i<5 do j=i+1;
-->p=p*j/i;
-->disp(p,' : p=',i,'étape')
-->i=i+1;
-->end
```

Exercice 2 (4,5 points)
Ecrire un programme en Scilab déterminant le plus petit entier naturel *n* admettant 15 diviseurs (pris parmi les entiers entre 1 et *n*)
Vous expliquerez ce programme et votre démarche par un texte en français.

Exercice 3 (5,5 points)
La devinette
On veut écrire un programme obéissant aux règles suivantes :
• l'ordinateur génère un entier aléatoire compris entre 0 et 999, demande ensuite à l'utilisateur de deviner ce nombre.
• Pour chaque entier proposé par l'utilisateur :
- l'ordinateur précise en cas d'échec s'il est plus petit ou plus grand que le nombre mystérieux, puis demande une nouvelle proposition à l'utilisateur.
- en cas de succès, il indique le nombre de propositions qui ont été nécessaire pour deviner le nombre mystérieux puis le programme s'arrête.

Sachant que Scilab possède la fonction prédéfinie rand(), donnant un nombre aléatoire entre 0 et 1, ainsi que la partie entière d'un réel positif int(x), écrire un programme en Scilab réalisant les objectifs ci-dessus.

Exercice 4 (6,5 points)
On démontre que la *série* de somme partielle $S_n = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$ tend vers $+\infty$ lorsque n tend vers $+\infty$.
Le problème consiste à trouver le nombre de termes de cette *série*, dite harmonique, nécessaire au dépassement d'une valeur arbitraire donnée :

$$1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} > \text{valeur}$$

1) Ecrire une fonction function [s]=harmonic(n) qui, pour un entier n passé en paramètre, calcule la somme $s = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$.
2) Ecrire un programme dans Scilab
- demandant à l'utilisateur une valeur strictement plus grande que 1 et plus petite ou égale à 8 (et répétant la saisie jusqu'à ce que la condition soit satisfaite)
- affichant le nombre de termes nécessaire au dépassement de cette valeur et le résultat de la somme (vous pourrez utiliser la fonction harmonic définie ci-dessus).

BCPST	Informatique : D.S. n°2	Samedi 9 mars

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) Ecrire une fonction ayant un tableau x comme paramètre et dont le résultat est la somme des éléments de x.

2) Ecrire une fonction ayant comme paramètres un tableau x et un entier n dont le résultat est la somme partielle des n premiers éléments de x : $\sum_{i=1}^{n} x(i)$

3) Ecrire une fonction ayant comme paramètre un tableau x et donnant le tableau y qui est affecté des sommes partielles des éléments de x ; on a ainsi $y(k) = \sum_{i=1}^{k} x(i)$

4) Ecrire une fonction ayant comme paramètre un tableau x de taille 10 et donnant le tableau y qui contient les moyennes mobiles des éléments de x, soit $y(k) = \frac{x(k - 1) + x(k + 1)}{2}$

Pour les extrémités du tableau, au lieu de $x(0)$, on prendra $x(10)$ et au lieu de $x(11)$, on prendra $x(1)$.

Exercice 2 : Naturels premiers chinois (5 points)

Il y a 2500 ans qu'en Chine fut émise la conjecture suivante :

Pour tout entier $n > 1$, n premier $\Leftrightarrow$ n divise $2^n - 2$

Nous appellerons *naturel premier chinois* ou encore *naturel pseudo-premier*, tout entier naturel $n > 1$ qui vérifie : n divise $2^n - 2$.

1) Ecrire un programme qui détermine tous les naturels pseudo-premiers jusqu'à 1000.

Fermat a montré en 1640 que pour tout naturel n :

n premier $\Rightarrow$ n divise $2^n - 2$

La réciproque est malheureusement fausse. (Serait-elle exacte que nous aurions là un excellent test pour reconnaître si un entier naturel est un nombre premier ou non)

2) Ecrire une fonction **[b]=premier(n)** qui détermine si un entier n donné en paramètre est un nombre premier ou non. Suivant l'entier n, la fonction renvoie alors le bouléen b contenant la valeur vrai ou faux.

3) Ecrire un programme qui écrit tous les entiers naturels premiers chinois non premiers, c'est-à-dire tous les contre-exemples de la relation établie par les chinois.

Exercice 3 (9 points)

Partie A

Vous allez dans un premier temps créer un ensemble de procédure et fonctions permettant de représenter les fractions et de travailler avec.

Les fractions seront représentées sous la forme d'un tableau de deux éléments, le premier contiendra le numérateur et le second le dénominateur. Exemple, le tableau [2,3] représente la fraction $\frac{2}{3}$.

1) Ecrire la fonction **[t]=lire** permettant la donnée d'une fraction (représentée par le tableau t) par un utilisateur au clavier. Vous devrez vous assurez que le dénominateur est non nul.

2) Ecrire la fonction **ecrire(f)** permettant de visualiser la fraction sous la forme p/q ou simplement p si q = 1. Pour l'affichage, vous pourrez utiliser l'instruction disp(string(f(1))+' / '+string(f(2))) dans le premier cas et adapter les autres affichages à partir de cette proposition.

Si q < 0, vous préférerez l'affichage de la forme -p/(-q) comme sur l'exemple suivant

Si f(1) = 8 et f(2) = -3, l'affichage sera -8/3

3) Ecrire la fonction **[c]=pgcd(a,b)** permettant de déterminer le pgcd (plus grand commun diviseur) des deux nombres a et b.
Vous pourrez utiliser par la méthode suivante ; si ce n'est pas le cas, vous expliquerez votre démarche par un texte clair.
Si un des deux nombres est nul, l'autre est le pgcd sinon il faut remplacer le plus grand par la différence entre le plus grand et le plus petit et laisser le plus petit inchangé.
Puis recommencer ainsi avec la nouvelle paire jusqu'à ce qu'un des deux nombres soit nul. Dans ce cas, l'autre nombre est le pgcd.
4) Ecrire la fonction **[t]=reduire(f)** permettant de simplifier la fraction f grâce à la fonction précédente.
5) Ecrire la fonction **[f3]=addition(f1,f2)** permettant d'obtenir la fraction f3 comme somme des fractions f1 et f2 (avec cette somme simplifiée).
6) Ecrire la procédure **[f3]=soustraction(f1,f2)** permettant d'effectuer, de la même manière la soustraction de f1 par f2.
7) Ecrire la procédure **[f3]=multiplication(f1,f2)** afin d'obtenir la fraction simplifiée f3 comme produit de f1 par f2.
8) Terminer la construction des outils de calculs sur les fractions par la procédure **[f3]=division(f1,f2)** permettant d'obtenir la division de f1 par f2.

Partie B :

Connaissant la fraction $\frac{a}{b}$, la division permet d'obtenir l'écriture décimale. Réciproquement, peut-on retrouver l'écriture fractionnaire ou la meilleure approximation fractionnaire d'un nombre exprimé en écriture décimale ?
Examinons le cas d'un réel positif.
Si ce réel contient "peu" de chiffres après la virgule, une méthode consiste à exprimer ce réel comme la division d'un entier par une puissance de dix (définition d'un nombre décimal) puis de simplifier cette fraction.
Dans le cas où ce réel est obtenu suite à un calcul et s'il contient "beaucoup" de chiffres après la virgule, une autre méthode consiste à déterminer la fraction continue associée à ce nombre puis de simplifier cette fraction continue.

Exemple : $\pi = 3{,}14159265359\ldots = \mathbf{3} + 0{,}14159265359\ldots = \mathbf{3} + \dfrac{1}{7{,}062513306\ldots}$

$$= \mathbf{3} + \frac{1}{\mathbf{7} + 0{,}062513306\ldots} = \mathbf{3} + \frac{1}{\mathbf{7} + \dfrac{1}{15{,}9965944068\ldots}}$$

$$= \mathbf{3} + \frac{1}{\mathbf{7} + \dfrac{1}{15{,}9965944068\ldots}} = \mathbf{3} + \frac{1}{\mathbf{7} + \dfrac{1}{\mathbf{15} + 0{,}9965944068\ldots}}$$

Les suite des nombres entiers [3 ; 7 , 15 , …] est la fraction continue associée au réel initial, dont des approximations de plus en plus fines sont les fractions successives :

$$\mathbf{3} \text{ puis } 3 + \frac{1}{7} = \frac{\mathbf{22}}{\mathbf{7}} \text{ puis } 3 + \frac{1}{7 + \dfrac{1}{15}} = \frac{\mathbf{333}}{\mathbf{106}} \text{ puis } \ldots$$

Nous nous contenterons, dans un premier temps, de la détermination des cinq premiers éléments des fractions continues qui seront donc représentées par un tableau de taille 5.
1) Ecrire la fonction **[f]=frac_cont(x)** permettant d'obtenir la fraction continue f du réel x.
2) Ecrire la fonction **reconstitution(f)** permettant la détermination puis l'affichage de la fraction simplifiée de l'approximation obtenue par la fraction continue représentée par le tableau f.
Exemple, si vous aviez la fraction continue [3 ; 7 , 15 , …] (mais elle est ici de taille 3), vous devrez déterminer puis afficher 333/106.

BCPST	Informatique : D.S. n°3	Samedi 8 juin

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

Vous pourrez, tout au long du ds, expliquer votre démarche par un texte en français.

Exercice 1 (4,5 points)

Soit la suite (x_n) définie par $x_0 = \frac{1+a}{2}$

$\forall n \in \mathbb{N}, x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$ où a désigne un réel positif ou nul.

Cette suite converge vers $\sqrt{a}$.
Ecrire un programme dans lequel a est lu au clavier, calculant et affichant les 10 premières valeurs de la suite (x_n).

Exercice 2 (4,5 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 $X_n = (l_n, c_n, d_n)$, $n \geqslant 0$, 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 3 Un générateur de Charabia latin (6,5 points)

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 fonctions suivantes :
Rappel : le $i^{\text{ième}}$ caractère de la chaîne c est obtenu par l'instruction part(c,i).
1) Ecrire la fonction **[mot_charabia]=Transf_Franc_Charabia(mot_francais)** permettant de transformer le mot français de la variable mot_francais en son équivalent en charabia latin dans la variable mot_charabia.

2) Ecrire la fonction **[s]=Recherche_mot(texte,n)** permettant de déterminer le $n^{ième}$ mot de la chaîne de caractères texte. Le résultat sera la chaîne s. Vous pourrez supposer que cette recherche est toujours possible, c'est-à-dire que l'entier *n* est toujours inférieur au nombre de mots de la chaîne texte.

3) Ecrire la fonction **[n]=Nombre_mots(texte)** permettant de déterminer le nombre de mots de la phrase donnée dans la chaîne de caractères texte en utilisant le nombre d'espaces (on supposera que deux mots sont séparés par un seul espace).

4) Ecrire le programme complet, utilisant les fonctions précédentes, permettant de demander une phrase au clavier, de la transformer en son équivalent en charabia latin et de l'afficher.

Exercice 3 Nombres de Kaprekar (6,5 points)
Lorsque l'on élève au carré un nombre de Kaprekar à *n* chiffres et qu'on ajoute le nombre formé par les *n* chiffres de droite à celui formés des *n* ou *n* – 1 chiffres de gauche, on retrouve le nombre d'origine.

Exemples : $9^2 = 81$ et $8 + 1 = 9$
$297^2 = 88\,209$ et $88 + 209 = 297$

Il s'agit de déterminer tous les nombres de Kaprekar inférieurs ou égaux à 1 000.

1) Construire la fonction **[t]=decomposer_nombre(n)** permettant de décomposer un entier *n* en ses chiffres que vous rangerez par la droite dans le tableau *t*, le tableau étant de taille 9.
Les premiers 0 dans le tableau doivent indiquer l'absence de chiffre en les positions correspondantes.
exemple : 325 a comme décomposition
dans le tableau ci-contre :

t

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	3	2	5

2) Ecrire la fonction **[n]=Reconstituer(t)** permettant de reconstituer l'entier *n* à partir de son écriture dans le tableau *t*.

3) Ecrire la fonction **[s]=somme(t)** permettant d'effectuer la somme des deux parties du nombre.
exemple : Si on doit déterminer la valeur de somme(*t*) du tableau de l'exemple précédent, vous devez affecter à la variable s le résultat 3 + 25 = 28.

4) Ecrire le programme Scilab permettant de déterminer tous les nombres de Kaprekar inférieurs ou égaux à 1 000.