Les Fonctions et Graphisme

Exercice 1.

Exemple (à taper dans SciNotes) :

```
function stom=envers(mots)
  n=length(mots)   //n : nombre de caractères dans mots
  stom=""   ;   // stom : chaîne de caractères vide
  for i=n:-1:1
      stom=stom+part(mots,i)
      //on parcourt les lettres de mots à l'envers
      //et on les enregistre dans stom
   end
endfunction
```

Charger la fonction (ctrl+L), puis essayer dans la console : --

-->envers("Cette fonction est vraiment trés utile !")

Posez-vous les questions suivantes :
- Quel est le nom de la fonction ?
- Quel est le nom de la variable d'entrée ?
- Quel est le nom de la variable de sortie ?
- Que fait cette fonction ?

Exercice 2.

En informatique, une fonction récursive est une fonction qui s'appelle elle-même. A contrario, une fonction qui ne s'appelle pas elle même sera dite itérative (ou encore impérative).

1) Ecrire la fonction factorielle itérative **FactoIt** qui calcule la factorielle d'un nombre entier, la tester et comprendre son procéssus

2) Ecrire la fonction factorielle récursive **FactoRe** qui calcule la factorielle d'un nombre entier, la tester et comprendre son procéssus

Exercice 3. Minimum et maximum

On interdit l'utilisation des fonctions min et max de Scilab dans cet exercice.

1. Créer une fonction maximum qui prend en argument un vecteur v et qui renvoie le plus grand de ses coefficients.
2. Faire de même avec une fonction minimum qui renvoie le plus petit de ses coefficients. (On pourra utiliser la fonction maximum).
3. Créer une fonction MatMax qui prend en argument une matrice A de taille n × p et qui retourne le plus grand de ses coefficients. Faire de mˆeme avec une fonction MatMin retournant le plus petit de ses coefficients.
4. Créer une fonction MinMaxLi qui prend en argument une matrice A de taille n×p et qui renvoie le plus petit des maximums de chaque ligne.
5. Faire de même avec une fonction MinMaxCo pour les colonnes.
6. On renverse le problème : créer deux fonctions MaxMinLi et MaxMinCo qui retournent respectivement le plus grand des minimums de chaque ligne/colonne.

Exercice 4.

Définir une fonction Scilab qui traduit la fonction mathématique suivant :($x \to x\sin(2x)$)

Exécuter à partir de la console la fonction y=f(x) sur l'intervalle $[-7, 7]$ et en prenant 1000 valeurs

A l'aide de l'instruction plot2d, tracer le graphe de y=f(x)

Exercice 5.

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 script 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).

Exercice 6.

Les polynômes de Tchebytchev sont déterminés par les relations suivantes :

$$\begin{cases} T_0 = 1 \\ T_1 = X \\ T_{n+2} = 2XT_{n+1}(X) - T_n(X) \end{cases}$$

1. Écrire une fonction récursive scilab `tche` qui prend deux arguments n et x et qui renvoie la valeur de $T_n(x)$.
2. Écrire une fonction scilab itérative, basée sur une boucle for, `tche2` qui prend deux arguments n et x et qui renvoie la valeur de $T_n(x)$
3. Calculer $T_8(0.7)$.
4. Tracer T_6 entre -1 et 1.

Exercice 7.

Créer une fonction du nom de votre choix qui prendra en argument trois paramètres : a, b et h et qui tracera ensuite le graphe de la fonction

$$f : x \longmapsto \frac{(4x^3 - 6x^2 + 1)\sqrt{\sin(x^2) + 1}}{1 + x^2}$$

sur $[a, b]$ et avec un pas de h.

Exercice 8.

Soit la fonction $f(x) = xe^x$, Ecrire un fonction Scilab qui permet de tracer f sur l'intervalle [a,b] passé en paramètres d'entrées;]en rouge.

Exercice 9.

1. Créer une fonction `syracuse` qui prendra en argument d'entrée un entier u_n et qui renverra $\frac{u_n}{2}$ si u_n est pair, $3u_n + 1$ si u_n est impair.
2. Créer un programme qui demandera à l'utilisateur un nombre u et qui affichera ensuite les termes de la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par $u_1 = u$ et pour $n > 0$:

$$u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{, si } u_n \text{ est pair} \\ 3u_n + 1 & \text{, si } u_n \text{ est impair} \end{cases}$$

   tant que $u_n \neq 1$ (on utilisera, bien évidemment, la fonction syracuse précédente).
3. Modifier ce dernier programme pour qu'il affiche aussi l'entier n du rang d'arrêt.

Exercice 10.

Suite de Fibonacci

1) Créer une fonction `fibonacci` qui prendra en argument un entier n et qui renverra les n premiers termes de la suite de Fibonacci.
Pour rappel la suite de Fibonacci est une suite récurrente linéaire d'ordre 2, définie par $u_1 = u_2 = 1$ et pour tout entier $n \in \mathbb{N}^*$

$$u_{n+2} = u_{n+1} + u_n$$

2) À partir de quel rang n a-t-on $u_n > 10^6$?

Exercice 11.

1. Créer dans SciNotes une fonction `factorielle` qui prendra en argument d'entrée un entier naturel k et qui renverra $k!$.
2. Ajouter dans le fichier déjà crée, une fonction `puissance` qui prendra en arguments d'entrée deux paramètres, le premier un nombre x et le second un entier naturel k et qui renverra le nombre x^k sans utiliser l'op´eration $\widehat{}$.

3. Ajouter à la suite des deux fonctions précédentes une fonction `expo` qui prendra en arguments d'entrée deux paramètres, un nombre x et un entier naturel n et qui renverra la somme

$$\sum_{k=0}^{n} x^k/k!$$

(on utilisera les fonctions `factorielle` et `puissance`). Comparer avec la valeur de e^x. Que remarquez-vous ?

Exercice 12.

Tracer $y = x^{\frac{1}{3}} + x^{\frac{1}{2}}$ pour x variant entre 10 et 20. Puis tracer $\ln y$ en fonction de $\ln x$ pour x variant entre 10 et 20.