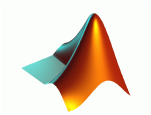

EXAMEN DE GÉNIE INFORMATIQUE

Durée 2h00.
Tout document autorisé.
Communication avec l'extérieur (internet, téléphone ou autre) interdit.
Collaboration avec d'autres étudiants interdite.

1 • INTRODUCTION.

1.1 • But de l'examen.

Hormis le fait de tester vos connaissances sur l'utilisation du logiciel MATLAB et la programmation en général cet examen propose de réaliser un programme de tri d'un vecteur d'une liste de mots (comme c'est le cas quand on veut faire un index alphabétique). Parmi les nombreuses techniques de tri existant, nous vous proposons de réaliser celle qui se nomme "tri à bulle" (bubble sort) qui se prête bien à une programmation modulaire.
Le programme à réaliser est très simple. Si vous êtes astucieux, il ne prendra pas beaucoup de lignes de code. Pour vous noter, nous lancerons simplement votre programme. Si tout fonctionne correctement et que votre programme est bien commenté, vous aurez la note maximale. Dans le cas contraire, les points vous seront attribués en fonction de la qualité de votre travail de programmateur. Pensez à commenter votre programme !!!
Si vous voulez réussir cette épreuve, il vous faut **très bien vous organiser**, ne mettre aucune valeur en dur et écrire tout bien proprement, sans quoi vous aurez toutes les peines du monde à enchaîner les différentes parties de cette épreuve.

1.2 • Restitution de votre travail.

Votre travail doit être restitué en fin d'épreuve sous la forme d'un (ou plusieurs) fichier(s). Vous devez envoyer vos fichiers avec vos nom et prénom à mon adresse (olivier.strauss@lirmm.fr) au maximum 5mn après la fin de l'épreuve! Toute minute de retard est retirée sur la note (1mn = 1/2 pt en moins) - la date d'envoi faisant foi.
Vous devez déclarer clairement vos variables en début de fichier.

2 • TRI A BULLE.

2.1 • Principe.

Le principe du tri à bulle est de comparer les couples d'élément successifs d'un tableau et de les échanger lorsqu'il ne respectent pas l'ordre voulu. L'algorithme est poursuivi récursivement jusqu'à ce que le tableau soit totalement trié (c'est à dire lorsqu'on n'échange plus aucun éléments lors d'une passe de l'algorithme).

Exemple : soit à trier par ordre croissant le vecteur $x = \begin{bmatrix} 5 & 8 & 3 & 5 & 9 & 2 \end{bmatrix}$.

1- On compare 5 et 8. 5 est inférieur à 8, on ne change rien. On compare 8 et 3. 8 est supérieur à 3, on échange 8 et 3 ($x = \begin{bmatrix}5 & 3 & 8 & 5 & 9 & 2\end{bmatrix}$). On compare 8 et 5. 8 est supérieur à 5, on échange 8 et 5 ($x = \begin{bmatrix}5 & 3 & 5 & 8 & 9 & 2\end{bmatrix}$). On compare 8 et 9. 8 est inférieur à 9, on ne change rien. On compare 9 et 2. 9 est supérieur à 2, on échange 9 et 2 ($x = \begin{bmatrix}5 & 3 & 5 & 8 & 2 & 9\end{bmatrix}$).

2- On compare 5 et 3. 5 est supérieur à 3, on échange 5 et 3 ($x = \begin{bmatrix}3 & 5 & 5 & 8 & 2 & 9\end{bmatrix}$). On compare 5 et 5. On ne fait rien. On compare 5 et 8. On ne fait rien. On compare 8 et 2. 8 est supérieur à 2, on échange 8 et 2 ($x = \begin{bmatrix}3 & 5 & 5 & 2 & 8 & 9\end{bmatrix}$). On compare 8 et 9. On ne fait rien.
Comme vous le voyez, en deux passes de l'algorithme le vecteur x est quasiment trié (il faudra encore deux passes pour le trier complètement).

Donc, un algorithme de tri à bulle à besoin de trois fonctionnalités : 1-comparaison, 2-échange, 3-boucle élémentaire.
Nous vous proposons de le réaliser pour une liste de mots. Vous trouverez la liste de mots en question à l'adresse suivante :
http://www.lirmm.fr/~strauss/ListeDeMots.txt
Comme vous pouvez le voir, ces mots comportent tous 6 lettres, ce qui évite la question de comparer deux chaînes de caractère ayant des tailles différentes.
Vous devez récupérer cette liste de mot et la restituer avec vos programmes.

2.2 • Lecture.

Votre fichier matlab doit s'appeler **TriDeMots.m**. Il doit débuter par la lecture du fichier ListeDeMots.txt.
Pour lire votre fichier et le stocker dans un tableau de caractères, vous devez débuter votre fichier par le code suivant :

```
fid = fopen('ListeDeMots.txt') ;
n=1 ; A=1 ;
while A>0 A = fgetl(fid) ;
 if A>0
  if n==1 TableauDeMots = A ;
  else TableauDeMots = [TableauDeMots ; A] ;
  end ;
  n=n+1 ;
 end ;
end ;
fclose(fid) ;
```

Q1 : Commentez ce code.
Attention !!! le premier mot de la liste `TableauDeMots` est `TableauDeMots(1,:)`.

2.3 • Fonction de comparaison.

Vous devez réaliser une fonction que vous appellerez **CompareMots**.
Cette fonction doit avoir comme prototype :

```
resultat = CompareMots( Mot1, Mot2 ) ;
```

Si `Mot1` est supérieur à `Mot2` la variable `resultat` doit être égale à 1, elle doit être égale à -1 si `Mot2` est supérieur à `Mot1` et 0 sinon.
Un mot est dit supérieur à un autre si sa première lettre est plus loin dans l'ordre alphabétique (par exemple P est supérieur à G), si les deux premières lettres sont identiques, on compare les deuxièmes lettres, et ainsi de suite (on appelle cela un ordre lexicographique). Cette fonction doit être placée dans un fichier appelé `CompareMots.m` qui fera partie des fichier restitués.
Il vous est conseillé de bien commenter cette fonction et de la tester. N'oubliez pas que les lettres sont des codes ASCII (ce qui simplifie la comparaison).

2.4 • Fonction d'échange.

Vous devez réaliser une fonction que vous appellerez **EchangeMots**.
Cette fonction doit avoir comme prototype :

```
[MotA, MotB] = EchangeMots( Mot1, Mot2 ) ;
```

Après appel de cette fonction, la chaîne de caractère se trouvant dans `Mot1` doit se retrouver dans `MotB` et celle qui se trouvait dans `Mot2` doit se retrouver dans `MotA`.

2.5 • Boucle unitaire.

Vous devez réaliser une fonction que vous appellerez **UneIterationDeTri** qui doit réaliser une itération du tri à bulle comme indiqué dans le paragraphe 2.1. On suppose qu'on trie dans l'ordre croissant de l'alphabet. Elle utilise les deux fonctions précédentes.
La fonction doit parcourir le `TableauDeMots` et échanger les mots qui ne sont pas dans le bon ordre. Le prototype de la fonction que vous devez créer est :

```
[TableauDeMots_modifie, modifie] = UneIterationDeTri( TableauDeMots ) ;
```

La variable `modifie` prend la valeur 1 s'il y a eu au moins un échange et 0 s'il n'y a eu aucun échange. C'est cette variable qui vous permettra d'arrêter l'algorithme de tri.

2.6 • Tri de la liste de mots.

Vous devez réaliser une fonction que vous appellerez **TriAlphabetique** qui doit réaliser le tri à bulle de la liste de mots. Cette fonction utilise la fonction `UneIterationDeTri` de façon récursive. Elle a comme prototype :

```
[TableauDeMotsTrie] = TriAlphabetique( TableauDeMots ) ;
```

Après appel de cette fonction, les mots se trouvant dans `TableauDeMotsTrie` doivent être rangés dans l'ordre lexicographique.
Dans votre programme principal (TriDeMots.m), vous devez faire suivre la lecture du fichier de mots de l'appel de la fonction. Après appel de votre programme, on doit afficher la liste initiale et la liste triée.