REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique

Concours Nationaux d'Entrée
aux Cycles de Formation d'Ingénieurs
Session 2015

الجمهورية التونسية
وزارة التعليم العالي والبحث العلمي

المناظرات الوطنية للدخول
إلى مراحل تكوين المهندسين
دورة 2015

Concours Toutes Options
Epreuve d'Informatique

Date : Mardi 09 Juin 2015 **Heure : 14 H** **Durée : 2 H** **Nombre de pages : 5**

Barème : PROBLEME 1 (MAPLE) : 11 points
PROBLEME 2 (ALGORITHMIQUE) : 9 points

DOCUMENTS NON AUTORISES
L'USAGE DES CALCULATRICES EST INTERDIT

PROBLEME 1 (MAPLE)

Partie 1 : Produit de Kronecker

Dans la littérature, outre le produit matriciel standard, différentes manières existent pour calculer un produit matriciel selon le besoin des applications (Hadamard, Kronecker, etc.). Dans cette partie nous abordons le produit matriciel de Kronecker qui sera utilisé dans la partie 2.

Soient A une matrice de taille $m \times n$ et B une matrice de taille $p \times q$. Leur produit de Kronecker, noté par le symbole $\otimes$, est la matrice $C = A \otimes B$ de taille $mp \times nq$ définie par blocs successifs de taille $p \times q$, où chaque bloc vaut $a_{ij} B$ avec $1 \le i \le m$ et $1 \le j \le n$.

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}$$

Exemple : Le produit de Kronecker de A de taille 2×2 par B de taille 3×2 est la matrice C de taille 6×4.

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix}$$

$$C = A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{11}b_{31} & a_{11}b_{32} & a_{12}b_{31} & a_{12}b_{32} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \\ a_{21}b_{31} & a_{21}b_{32} & a_{22}b_{31} & a_{22}b_{32} \end{pmatrix}$$

Le produit de Kronecker présente certaines propriétés. On propose de vérifier les propriétés suivantes :

- La distributivité par rapport à l'addition

  Pour tout quadruplet de matrices (A, B, E, F) on a :

  $$(A+B)\otimes(E+F)=(A\otimes E)+(A\otimes F)+(B\otimes E)+(B\otimes F) \quad (1)$$

  où A et B de même dimensions ainsi que E et F.

- La transposition

  Pour toutes matrices A et B on :

  $$\text{transposé}(A\otimes B)=\text{transposé}(A)\otimes\text{transposé}(B) \quad (2)$$

- La trace

  $$\text{trace}(A\otimes B)=\text{trace}(A)\times\text{trace}(B) \quad (3)$$

- Le déterminant

  $$\text{déterminant}(A\otimes B)=(\text{déterminant}(A))^{m}\times(\text{déterminant}(B))^{n} \quad (4)$$

  où A et B deux matrices carrées non singulières de dimensions respectives $n\times n$ et $m\times m$.

N.B :

- Les commandes Maple suivantes sont prédéfinies :
  - **coldim(A)** retourne le nombre de lignes de la matrice A.
  - **rowdim(A)** retourne le nombre de colonnes de la matrice A.
  - **equal(A , B)** retourne *true* si les matrices A et B sont égales et *false* sinon.

- La vérification de la compatibilité des tailles des matrices n'est pas demandée.

Travail demandé

1. En supposant que la procédure **Kronecker** est déjà définie et dont l'appel **Kronecker(A,B);** retourne $A\otimes B$. Donner les instructions Maple permettant de vérifier les propriétés du produit de Kronecker suivantes :

   1.1. la distributivité par rapport à l'addition (1)
   1.2. la transposition (2)
   1.3. la trace (3)
   1.4. le déterminant (4)

2. Ecrire la procédure **Kronecker** qui calcule le produit matriciel $A\otimes B$.

Partie 2 : Equation de sylvester

3. On se propose dans cette partie de résoudre l'équation matricielle de sylvester donnée par :

   $$AX+XB=C \quad \textbf{(eq1)}$$

avec :

- A une matrice de dimensions $n\times n$ de coefficients a_{ij} $(1\le i\le n$ et $1\le j\le n)$.
- X la matrice inconnue de dimensions $n\times m$ de coefficients x_{ij} $(1\le i\le n$ et $1\le j\le m)$.
- C la matrice de dimensions $n\times m$ de coefficients c_{ij} $(1\le i\le n$ et $1\le j\le m)$.
- B une matrice de dimensions $m\times m$ de coefficients b_{ij} $(1\le i\le m$ et $1\le j\le m)$.

Pour résoudre **(eq1)** on la transforme en un système linéaire classique de la forme :

$My = v$ **(eq2)**

avec :

- $M = (I_m \otimes A) + (B^T \otimes I_n)$.
- $y = [x_{11}, \ldots, x_{1m}, x_{21}, \ldots, x_{2m}, \ldots\ldots, x_{n1}, \ldots, x_{nm}]$.
- $v = [c_{11}, \ldots, c_{1m}, c_{21}, \ldots, c_{2m}, \ldots\ldots, c_{n1}, \ldots, c_{nm}]$.
- B^T est la transposée de B.
- I_n et I_m sont les matrices identités d'ordres respectifs n et m.

Sachant que l'équation **(eq1)** possède une solution, écrire une procédure Maple **Sylvester1** d'entête :

Sylvester1 :=proc(A ::matrix, B :: matrix, C :: matrix)

qui détermine et retourne la matrice X solution de l'équation **(eq1)** en appliquant la démarche suivante :

- calculer $K1 = (I_m \otimes A)$;
- calculer $K2 = (B^T \otimes I_n)$;
- calculer $M = K1 + K2$;
- déterminer le vecteur v à partir de la matrice C ;
- déterminer le vecteur y par résolution du système linéaire classique $My = v$;
- déterminer la matrice X à partir du vecteur y ;
- retourner la matrice X.

4. Dans le cas où la matrice M de l'équation **(eq2)** est non singulière, la solution de l'équation **(eq1)** de **Sylvester** existe et est unique. La matrice M est non singulière si l'intersection de l'ensemble des valeurs propres de la matrice A et celui de la matrice $(-B)$ donne l'ensemble vide.

   Ecrire une procédure Maple **Sylvester2** d'entête :

   Sylvester2 :=proc(A ::matrix, B :: matrix, C :: matrix)

   qui retourne la solution obtenue par Sylvester1 si la solution existe. Prévoir un message d'erreur dans le cas contraire.

PROBLEME 2 (ALGORITHMIQUE)

Le but de ce problème est de développer quelques algorithmes de recherche appliqués sur les tableaux.

Dans ce qui suit, on suppose avoir déjà effectué les déclarations suivantes :

CONSTANTE NMAX = 1000

TYPE VECT = tableau [1..NMAX] de entier

On suppose également manipuler un tableau T de type VECT à n éléments ($2 \leq n \leq NMAX$).

N.B :

- L'entête de chaque procédure ou fonction demandée est donnée à la suite de chaque question.
- Le mot ***variable*** signifie le mode de passage **S** ou **E/S**.

Travail demandé

Partie 1 : Algorithmes séquentiels

1) Ecrire la fonction **Max** qui permet de retourner la valeur maximale contenue dans un tableau.

   Fonction Max (T : VECT , n : entier) : entier

2) Ecrire la fonction **Pos_Max** qui permet de retourner la position de la valeur maximale contenue dans un tableau.

   Fonction Pos_Max (T : VECT , n : entier) : entier

3) En utilisant impérativement **Pos_Max**, écrire la procédure Min_Max qui détermine dans M1 la valeur du maximum et dans M2 celle du minimum contenue dans un tableau.

   Procedure Min_Max (T : VECT , n : entier , variable M1,M2 : entier)

4) Ecrire la fonction **Verif** qui permet de retourner *vrai* si tous les éléments du tableau sont distincts et *faux* sinon. Prévoir une sortie au premier test qui donne *faux*.

   Fonction Verif (T : VECT , n : entier) : booleen

5) En supposant que les éléments du tableau sont tous distincts, écrire une fonction **Max2** qui permet de retourner le $2^{ème}$ plus grand élément du tableau.

   Fonction Max2 (T : VECT , n : entier) : entier

Partie 2 : Algorithmes particuliers

6) Ecrire la procédure **Max_Bis** d'entête :

   Procedure Max_Bis (T : VECT , n : entier , variable M : entier)

   qui détermine dans **M** la valeur maximale contenue dans **T** en utilisant le principe suivant :

   - Dans chaque étape du processus, les valeurs sont comparées par paire. Les plus grands éléments de chacune des paires sont placés dans la première partie du tableau. Dans le cas où le nombre d'éléments du tableau est impair, le dernier élément est stocké à la suite des éléments déjà placés. Le nombre d'éléments significatifs est mis à jour par le nombre d'éléments placés.
   - On répète ce processus jusqu'à ce qu'il n'y ait plus qu'un seul élément significatif.

Exemple : Soit le tableau **A** :

A	**51**	**8**	**5**	**6**	**40**	**4**	**7**	**510**	**17**	**700**
Etape 1	**51**	**6**	**40**	**510**	**700**	4	7	510	17	700
Etape 2	**51**	**510**	**700**	510	700	4	7	510	17	700
Etape 3	**510**	**700**	700	510	700	4	7	510	17	700
Etape 4	**700**	700	700	510	700	4	7	510	17	700

Dans ce cas **M = 700**

N.B : les cases grisées sont non significatives.

7) Ecrire une fonction **Max2_Bis** qui détermine le 2ème plus grand élément du tableau comme suit :
   - On recherche tout d'abord le plus grand élément par appel à la procédure précédente (**Max_Bis**).
   - Le deuxième plus grand élément est obtenu en recherchant l'élément maximal parmi les éléments restants.

   Fonction Max2_Bis (T : VECT , n : entier) : entier

8) En supposant que la taille **n** du tableau est paire, écrire une procédure **Min_Max_Bis** d'entête :

   Procedure Min_Max_Bis(T : VECT , n : entier , variable M1,M2 : entier)

   qui détermine dans **M1** la valeur du maximum et dans **M2** celle du minimum contenues dans un tableau **T** de type **VECT** en appliquant le principe suivant:
   - Les éléments du tableau sont comparés par paire; le maximum de chaque paire est stocké à gauche (indice impair) et le minimum est stocké à droite (indice pair).
   - On recherche le maximum du tableau parmi tous les plus grands éléments de chacune des paires.
   - On recherche le minimum du tableau parmi tous les plus petits éléments de chacune des paires.

Exemple :

Tableau initial	51	8	5	6	40	4	7	510	17	700
Après permutation	51	8	**6**	**5**	40	4	**510**	7	**700**	**17**

Dans ce cas **M1 = 700 et M2 = 4**