REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement
Supérieur, de La Recherche Scientifique et
de La Technologie
Concours nationaux d'entrée aux cycles
de formation d'ingénieurs
Session 2008

الجمهورية التونسية
وزارة التعليم العالي
والبحث العلمي والتكنولوجيا

المناظرات الوطنية للدخول
إلى مراحل تكوين المهندسين
دورة 2008

Concours Mathématiques et Physique, Physique et Chimie, Biologie et Géologie &Technologie
Epreuve d'Informatique

Date : Mardi 03 Juin 2008 **Heure : 15 H** **Durée : 2 H** **Nbre pages : 6**

Barème : EXERCICE 1 : 4 points, EXERCICE 2 : 5 points, PROBLEME : 11 points

DOCUMENTS NON AUTORISES
L'USAGE DES CALCULATRICES EST INTERDIT

EXERCICE 1

On considère l'expression algébrique suivante :

$$F = \frac{(x-y-1)(x^2-y-1)}{2(xy-1)}$$

Donner les commandes en **MAPLE** permettant de :

1. définir F ;
2. évaluer F pour $x = 1$ et $y = 3$;
3. récupérer dans NF la forme développée du numérateur de F et dans DF le dénominateur de F ;
4. donner le nombre des coefficients de NF ;
5. donner le quotient et le reste de la division euclidienne de NF par DF par rapport à x ;
6. définir la fonction f dont l'expression est F et la fonction $f5$ tel que $f5(x) = f(x, 5)$;
7. donner les points singuliers de f (les points où f n'est pas définie);
8. chercher les racines de $f5$;
9. représenter graphiquement $f5$ sur $[-2\pi, 2\pi]$;
10. représenter graphiquement f pour x dans $[-10,10]$ et y dans $[-10,10]$.

EXERCICE 2

On se propose d'approcher la valeur de π en utilisant différentes méthodes.

Présentation des méthodes :

1. Méthode *Cues*

   Cette méthode se base sur les deux suites a et b définies par :

   $$\begin{cases} a_1 = 0 \\ a_{n+1} = \dfrac{a_n + b_n}{2} \end{cases} \quad et \quad \begin{cases} b_1 = \dfrac{1}{4} \\ b_{n+1} = \sqrt{a_{n+1}\, b_n} \end{cases}$$

   Les termes $\dfrac{1}{2a_n}$ et $\dfrac{1}{2b_n}$ convergent vers π lorsque n tend vers ∞.

2. Méthode *Brounker*

   La fraction continue suivante permet de calculer la valeur de π lorsque n tend vers ∞ :

   $$\pi = 4 \cfrac{1}{1 + \cfrac{1^2}{2 + \cfrac{3^2}{2 + \cfrac{5^2}{2 + \cfrac{\cdots}{2 + \ddots \cfrac{((2n-1)-2)^2}{2+(2n-1)^2}}}}}}$$

3. Méthode de *Newton* :

   π est calculé par :

   $$\pi = 6 \sum_{n=0}^{\infty} \frac{(2n)!}{2^{4n+1} (n!)^2 (2n+1)}$$

4. Méthode d'*Euler* :

   La valeur de π est donnée par :

   $$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$$

Travail demandé :

1. Donner la commande en Maple donnant une valeur approchée de π avec 20 chiffres significatifs.
2. Ecrire une procédure en Maple, nommée ***Cues***, calculant la valeur de π avec une précision ε donnée. La valeur approchée de π est atteinte lorsque la différence en valeur absolue entre deux termes successifs est inférieure à ε.
3. Ecrire une procédure en Maple, nommée ***Brounker***, retournant une valeur approchée de π. Cette procédure prend comme paramètre en entrée la valeur de ***n***.
4. Ecrire la commande en Maple qui permet d'approcher la valeur de π par la méthode de ***Newton***.
5. Sans utiliser la commande prédéfinie de Maple (utilisée en 4), écrire une procédure en Maple, nommée ***Euler***, retournant une valeur approchée de π avec une précision ε donnée. La valeur approchée de π est atteinte lorsque la différence en valeur absolue entre les valeurs obtenues suite à l'exécution de deux itérations successives est inférieure à ε.

PROBLEME

La bio-informatique est une discipline récente utilisant des notions de mathématiques, d'informatique et de biologie. L'une des occupations de ce domaine est la recherche de similarités entre des séquences d'ADN. Du point de vue d'un bio-informaticien, une séquence ADN est une collection ordonnée d'un nombre variable de caractères choisis dans l'alphabet **(A, C, G, T)**.
AATCGACTTACGGAT est un exemple de séquence d'ADN.
Le problème a pour but de mesurer la similarité entre deux séquences. La **similarité** est définie comme étant la mesure de la ressemblance entre deux séquences. Plusieurs algorithmes sont apparus pour déterminer la similarité pour deux séquences de longueurs différentes.
On définit un **alignement** comme étant la mise en correspondance des caractères des deux séquences en insérant des espaces dans la séquence la moins longue pour que les longueurs deviennent identiques.
Le travail demandé consiste à aligner deux séquences de longueurs différentes et à calculer la similarité entre elles. En partant de deux séquences *S1* et *S2* où *S2* est la séquence la moins longue, la méthode qui sera utilisée consiste à compléter la séquence *S2* par des espaces représentés par le caractère tiret « - » de façon que le nombre de caractères dans *S1* et dans *S2* soit le même. Les tirets peuvent être ajoutés à la séquence *S2* de différentes façons et dans différentes positions (voir exemple 1).

Description de la méthode

Les séquences *S1* et *S2* seront représentées respectivement par deux tableaux *A* et *B*. Le tableau *A* contient les *n* caractères de *S1* et le tableau *B* contient les *m* caractères de *S2* avec $n>m$. Pour aligner *A* et *B*, on utilisera une matrice de caractères *M* à *q* lignes et *n* colonnes où *q* est le nombre de combinaisons entre *n* et *p* avec $p=(n-m)$. Selon les règles de l'analyse combinatoire il y a, $\frac{n!}{p!(n-p)!}$ possibilités de placer des tirets entre les caractères du tableau *B*. Chaque ligne de la matrice *M* contient ainsi une combinaison différente des caractères de *B* avec les tirets.

Exemple 1 :
Soient les séquences *S1* et *S2* représentées respectivement par les tableaux *A* et *B* de tailles respectives $n=5$ et $m=2$; dans ce cas $p=3$ et $q=\frac{n!}{p!(n-p)!}$ qui vaut 10.

S1 = A A T C G

A	A	A	T	C	G

S2 = A C

B	A	C

La matrice *M* à *q* lignes et *n* colonnes correspondant à *B* est la suivante :

$$M=\begin{bmatrix} - & - & - & A & C \\ - & - & A & - & C \\ - & - & A & C & - \\ - & A & - & - & C \\ - & A & - & C & - \\ - & A & C & - & - \\ A & - & - & - & C \\ A & - & - & C & - \\ A & - & C & - & - \\ A & C & - & - & - \end{bmatrix}$$

Afin de créer la matrice *M* où chaque ligne contient les caractères du tableau *B* avec un arrangement différent des tirets, on fait usage d'un tableau monodimensionnel *S* d'entiers contenant pour chaque ligne de *M* les positions des tirets à placer sur la ligne en question.

Le tableau initial servant à la création de la première ligne de M est le tableau S contenant p entiers successifs : S = [*1, 2,..., p*], avec $p=(n-m)$. A partir d'un tableau précédent S, la génération d'un nouveau tableau $S = [n_1, n_2,..., n_p]$ à p entiers, avec $n_p \leq n$ et $n_i < n_{i+1}$ pour $1 \leq i \leq p-1$, se fait selon le principe suivant.:

- Si $n_p + 1 <= n$, le tableau S à p éléments devient alors : $S = [n_1, n_2,..., n_p+1]$ sinon passer à l'élément n_{p-1}.
- Pour l'élément n_{p-1}, si $n_{p-1} + 2 <= n$, le tableau S contenant p éléments devient alors : $S = [n_1, n_2,..., n_{p-1}+1, n_{p-1}+2]$ sinon passer à l'élément n_{p-2} et ainsi de suite.

.

- Pour l'élément n_i, avec $1 \leq i \leq p$, si $n_i + (p-i+1) <= n$, le nouveau tableau S contenant p éléments sera : $S = [n_1, n_2,..., n_i+1, n_i+2, n_i+3,..., n_i+(p-i+1)]$.

Appliquons ce principe à l'exemple 1 pour $n = 5$ et $p = 3$,

- Si S =[1, 2, 3] (tableau initial), le nouveau tableau S sera S = [1, 2, 4] car $3 + 1 <= n$.
- Si S =[1, 2, 4], le nouveau tableau S sera S = [1, 2, 5] car $4 + 1 <= n$.
- Si S = [2, 4, 5], on a :
  - $5+1 > n$, on passe alors à l'élément précédent qui est 4 ;
  - $4+2 > n$, on passe à celui qui le précède qui est 2 ;
  - $2+3 <= n$, dans ce cas on incrémente l'élément 2 de 1, on obtient 3. Les éléments suivants de S seront alors les successeurs de 3. Le nouveau tableau S obtenu est : S = [3, 4, 5].

Ainsi, chaque ligne de la matrice M est créée à partir d'une occurrence du tableau S et des éléments de B comme le montre l'exemple. Chaque occurrence de S est générée selon le principe ci-dessus décrit.

Tableau initial S =

1	2	3

S =

1	2	4

S =

1	2	5

S =

1	3	4

S =

1	3	5

S =

1	4	5

S =

2	3	4

S =

2	3	5

S =

2	4	5

S =

3	4	5

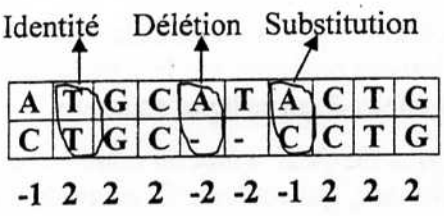
$$M = \begin{bmatrix} - & - & - & A & C \\ - & - & A & - & C \\ - & - & A & C & - \\ - & A & - & - & C \\ - & A & - & C & - \\ - & A & C & - & - \\ A & - & - & - & C \\ A & - & - & C & - \\ A & - & C & - & - \\ A & C & - & - & - \end{bmatrix}$$

Pour le calcul des scores de similarité et la recherche du score optimal, on détermine la similarité entre la séquence représentée par le tableau A à n éléments et chacune des séquences figurant sur les q lignes de la matrice M représentant différents alignements des m éléments du tableau B. Pour chaque ligne de M, le principe de calcul se base sur la comparaison de chaque caractère de A avec celui de la ligne de M se trouvant à la même position. Trois situations sont possibles pour une position donnée de l'alignement et à chacune des situations on attribue un score :

- Identité : ayant pour score 2 si les caractères sont les mêmes,
- Délétion : ayant pour score -2 si l'une des positions est un tiret.
- Substitution : ayant pour score -1 si les caractères sont différents,

Le score d'une comparaison entre deux séquences est la somme des scores de comparaison caractère par caractère. Le score optimal est le plus grand score parmi les scores obtenus en alignant A avec chacune des lignes de la matrice M.

Exemple 2:



Le score de cette comparaison est égal à 6.

Travail demandé

On suppose avoir effectué les déclarations suivantes :

CONSTANTE *NMAX* = 10

QMAX = 252

TYPE *TABC* = TABLEAU [1..*NMAX*] DE CARACTERE

TABE = TABLEAU [1..*NMAX*] D'ENTIER

MAT = TABLEAU [1..*QMAX*, 1..*NMAX*] DE CARACTERE

1. Ecrire une fonction algorithmique *SAISIE_NB* permettant de retourner un entier saisi au clavier compris entre 1 et *NMAX*.
2. Ecrire une procédure algorithmique *SAISIE_SEQ* permettant de remplir un tableau *T* avec *n* caractères. Chaque élément saisi devra appartenir à l'alphabet (A, C, G, T).
3. Ecrire une fonction algorithmique *FCT* qui calcule le factoriel d'un entier *n* passé en paramètre.
4. Ecrire une fonction *COMB* qui à partir des deux paramètres entiers *n* et *m*, retourne la valeur de l'expression $\frac{n!}{m!(n-m)!}$.
5. Ecrire une procédure algorithmique *INIT_TPOS* qui, à partir d'un paramètre entier *p*, remplit un tableau *S* avec des entiers successifs de 1 à *p* dans l'ordre.
6. Ecrire une procédure algorithmique *ECRIRE_TIRETS* qui écrit, à partir d'un tableau *S* d'entiers, d'un numéro de ligne *L* et d'une matrice *M* de caractères, des tirets « - » à la ligne *L* de *M* sur les colonnes d'indices les valeurs du tableau *S*.
7. Ecrire une procédure algorithmique *ECRIRE_SEQ* qui écrit, dans les cases ne contenant pas des tirets « - » de la ligne *L* d'une matrice *M*, les *m* caractères d'un tableau *B* dans l'ordre.
8. Ecrire une procédure algorithmique *INCREM_SEQ* qui détermine, à partir d'un tableau *S* à *p* entiers et d'un entier *n*, le nouveau contenu du tableau *S* (voir paragraphe description de la méthode).
9. En utilisant les procédures et fonctions adéquates, écrire la procédure algorithmique CREE_MAT qui crée, à partir d'un tableau *B* de *m* caractères, la matrice *M* de type *MAT* à *q* lignes (à déterminer) et *n* colonnes comme suit:
   - calculer le nombre de lignes *q* de *M* à partir de *n* et *m*,
   - initialiser tous les éléments de *M* au caractère « x »,
   - créer un tableau initial *S* avec *p* valeurs entières successives,
   - placer les tirets « - » aux endroits correspondants de la première ligne de *M*,
   - remplir les cases restantes avec les *m* éléments de *B* dans l'ordre,

- itérer les deux étapes précédentes pour les lignes suivantes de *M*.
  A chaque itération :
  - ➢ générer un nouveau tableau *S* à partir du tableau précédent ;
  - ➢ écrire les tirets à leurs bonnes positions sur la ligne courante de la matrice *M* ;
  - ➢ écrire les éléments du tableau *B* sur la ligne courante.

10. Ecrire une fonction algorithmique *SCORE*, ayant en entrée une matrice *M*, un numéro de ligne *L* et un tableau *A* à *n* colonnes, qui retourne le score de similarité correspondant en attribuant le score –2 à une délétion, 2 à une identité et -1 à une substitution.

11. Ecrire une fonction algorithmique *SCORE_OPT* qui retourne le meilleur score entre le tableau *A* et chacune des lignes de la matrice *M*.