



**Concours Nationaux d'Entrée aux Cycles de Formation d'Ingénieurs  
Session 2009**

**Concours Toutes Options  
Epreuve d'Informatique**

**Date : Mardi 02 Juin 2009**

**Heure : 15 H**

**Durée : 2 H**

**Nbre pages : 5**

**Barème : EXERCICE 1 : 4 points**

**EXERCICE 2 : 6 points**

**PROBLEME : 10 points**

**DOCUMENTS NON AUTORISÉS  
L'USAGE DES CALCULATRICES EST INTERDIT**

**EXERCICE 1**

Soit la fonction  $f$  définie par :

$$f(x) = \begin{cases} \frac{x+1}{2} & \text{si } -1 \leq x \leq 1 \\ 0 & \text{sinon} \end{cases}$$

Donner les commandes MAPLE permettant de :

- 1) définir la fonction  $f$  ;
- 2) représenter graphiquement  $f$  entre  $-\pi$  et  $\pi$  ;
- 3) affecter à  $a_n$  et  $b_n$  respectivement  $a_n$  et  $b_n$ , les coefficients de Fourier de  $f$  définis par :

$$a_n = \frac{1}{\pi} \int_{-\infty}^{+\infty} f(x) \cos(nx) dx \quad \text{et} \quad b_n = \frac{1}{\pi} \int_{-\infty}^{+\infty} f(x) \sin(nx) dx$$

- 4) convertir  $a_n$  et  $b_n$  en deux suites  $a$  et  $b$ , fonctions de  $n$  ;
- 5) calculer la limite de  $a$  et celle de  $b$  quand  $n$  tend vers 0 ;
- 6) définir la liste  $L1$  contenant les points  $[i, a(i)]$  et la liste  $L2$  contenant les points  $[i, b(i)]$  pour tous les entiers  $i$  de l'intervalle  $[1, 20]$ .
- 7) représenter sur le même graphisme les deux listes  $L1$  et  $L2$ .

**Indication :** La commande `plot` permet de représenter une liste de points. Chaque point est défini par une liste constituée de son abscisse puis son ordonnée. Pour obtenir une représentation en points, utiliser l'option `style=point`.

8) définir la fonction  $SF$  à deux variables  $x$  et  $m$  représentant la série de Fourier à l'ordre  $m$ . On rappelle que la série de Fourier à l'ordre  $m$  est donnée par :

$$\frac{a_0}{2} + \sum_{k=1}^m (a_k \cos(kx) + b_k \sin(kx))$$

9) donner sur le même graphisme, la commande permettant la représentation graphique de  $f(x)$ ,  $SF(x, 2)$  et  $SF(x, 20)$  pour  $x \in [-\pi, \pi]$ .

## EXERCICE 2

1) Soient  $A$  une matrice carrée d'ordre  $n$  et  $tr(A)$  sa trace ( $tr(A) = \sum_{i=1}^n A_{i,i}$ ). On souhaite déterminer  $P$ , le polynôme caractéristique de  $A$ , donné par :

$P = x^n + C_1 x^{n-1} + C_2 x^{n-2} + \dots + C_{n-1} x + C_n$  où les  $C_i$  sont les coefficients de  $P$  selon les puissances décroissantes en  $x$ .

Pour cela, on calcule la suite des matrices  $A_i$  ainsi que leurs traces  $tr(A_i)$  définies par :

$$\begin{cases} A_1 = A, & C_1 = -tr(A_1) \\ A_i = (A_{i-1} + C_{i-1}I)A, & C_i = \frac{-tr(A_i)}{i} \text{ pour } 2 \leq i \leq n \end{cases}$$

$I$  étant la matrice identité d'ordre  $n$ .

Ecrire une procédure MAPLE, nommée `calcul_pol`, ayant comme paramètres une matrice carrée  $M$  et son ordre  $n$  et qui retourne le polynôme caractéristique de  $M$  en  $x$  en utilisant la méthode de calcul ci-dessus expliquée.

2) Soit  $M$  une matrice définie par :

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Ecrire les commandes MAPLE permettant de :

- 2.1) charger le package `linalg` ;
- 2.2) définir une fonction  $f$  qui sert à remplir la matrice  $M$  ;
- 2.3) définir  $M$  en utilisant  $f$  ;
- 2.4) affecter à la variable  $Id$  une matrice identité d'ordre  $n$  ( $n$  supposée définie) ;
- 2.5) affecter à la variable  $P1$  le polynôme caractéristique de  $M$  en  $x$  (sans utiliser `calcul_pol`) ;
- 2.6) affecter à la variable  $P2$  le polynôme caractéristique de  $M$  par appel à `calcul_pol` ;
- 2.7) vérifier si  $P1$  et  $P2$  sont identiques.

3) Écrire une procédure MAPLE, nommée **comparaison**, ayant comme paramètres deux polynômes  $P$  et  $Q$  en  $x$  et qui permet, par comparaisons successives des coefficients des termes de même degré, de retourner *vrai* si les polynômes sont identiques et *faux* sinon.

## PROBLEME

On désire manipuler en base 2 des nombres réels appartenant à l'intervalle  $[0,1[$  avec une grande précision.

Soit  $x$  un réel appartenant à l'intervalle  $[0,1[$ . Ce réel s'écrit sur  $N$  chiffres en base 2 sous la forme :

$$x = \sum_{i=1}^N x_i 2^{-i} = 0.x_1 x_2 x_3 \dots x_N \text{ tels que les entiers } x_i \in \{0, 1\}.$$

Le réel  $x$  sera représenté par un tableau  $T$  de  $N$  entiers tel que  $T[i] = x_i$ . La méthode de calcul des  $x_i$  du tableau  $T$  est la suivante :

$$x_1 = \text{partie entière de } 2x \text{ (} x \in [0,1[), \text{ donc } x_1 = \begin{cases} 0 & \text{si } 2x < 1 \\ 1 & \text{sinon} \end{cases}$$

Notons  $R_1 = 2x - x_1$ .

Si  $R_1 = 0$  alors les entiers  $x_2, x_3, \dots, x_N$  valent tous zéro.

Si  $R_1 \neq 0$ , on calcule à chaque fois  $x_i$  et  $R_i$  tels que :

$$\begin{cases} x_i = \text{partie entière de } (2R_{i-1}) \\ R_i = 2R_{i-1} - x_i \end{cases} \text{ pour } 2 \leq i \leq N$$

On arrête les calculs lorsque l'une des deux conditions suivantes est satisfaite :

- $R_i = 0$ , dans ce cas, les entiers  $x_{i+1}, x_{i+2}, \dots, x_N$  valent tous zéro ;
- $i = N$ .

Exemple : pour un réel  $x = 0.625$

$$x_1 = \text{partie entière de } 2x, \text{ donc } x_1 = 1$$

$$R_1 = 2x - x_1 = 0.625 * 2 - 1 = 0.25$$

$$R_1 \neq 0, \text{ on calcule donc } \begin{cases} x_2 = \text{partie entière de } (2R_1) = 0 \\ R_2 = 2R_1 - x_2 = 0.5 \end{cases}$$

$$R_2 \neq 0, \text{ on calcule donc } \begin{cases} x_3 = \text{partie entière de } (2R_2) = 1 \\ R_3 = 2R_2 - x_3 = 0 \end{cases}$$

$$R_3 = 0, \text{ donc } x_4 = 0, x_5 = 0, \dots, x_N = 0.$$

Dans la suite, on suppose avoir effectué les définitions suivantes :

**Constante**    **N=1000**

**Type**         **TABGR = tableau [1 .. N] d'entier**

On suppose également que les variables réelles  $x$ ,  $y$  et  $z$  appartenant à l'intervalle  $[0,1[$  se représentent respectivement par les tableaux  $Tx$ ,  $Ty$  et  $Tz$  de type TABGR.

**N.B :** Le mot **variable** figurant dans les entêtes des procédures et fonctions signifie le mode de passage S ou ES.

1) Ecrire une procédure algorithmique **saisie**, permettant de saisir un réel  $w$  ( $w \in [0,1[$ ), et dont l'entête est :

**Procédure saisie(variable w : réel)**

2) Ecrire une procédure algorithmique **convert**, qui convertit un réel  $w$  ( $w \in [0,1[$ ), en sa forme binaire représentée par un tableau  $Tw$  de type TABGR en utilisant la méthode proposée ci dessus. Cette procédure a comme entête :

**Procédure convert (w : réel , variable Tw :TABGR)**

3) Ecrire une fonction algorithmique **plusgrand** telle que :

$$\text{plusgrand}(Tx, Ty) = \begin{cases} \text{vrai} & \text{si } x \geq y \\ \text{faux} & \text{si } x < y \end{cases}$$

Cette fonction a comme entête :

**Fonction plusgrand(Tx, Ty :TABGR) : booléen**

4) La multiplication par 2 d'un nombre binaire revient à décaler tous ses chiffres d'une position vers la gauche.

Exemple :  $(0.0101011)_2 * 2 = (0.1010110)_2$

Ecrire une procédure algorithmique **foisdeux**, qui multiplie par 2 un réel  $x$  représenté par un tableau  $Tx$  donnant si possible un tableau  $Ty$  représentant un réel  $y$  et une variable booléenne **tropgrand** telle que :

$$\text{tropgrand} = \begin{cases} \text{vrai} & \text{si } 2x \geq 1 \\ \text{faux} & \text{si } 2x < 1, \text{ dans ce cas } y = 2x \end{cases}$$

Cette procédure a comme entête :

**Procédure foisdeux(Tx :TABGR , variable Ty : TABGR, variable tropgrand : booléen)**

5) Soient deux variables  $x$  et  $y$  de type réel telles que  $0 \leq x < y < \frac{1}{2}$ . On veut effectuer la division de  $x$  par  $y$ . Pour cela, on définit la suite  $u$  de réels et la suite  $q$  de chiffres binaires (0 ou 1) par :

$$\begin{cases} u_0 = x \\ u_{i+1} = 2u_i - q_{i+1}y \end{cases} \quad \text{avec} \quad q_{i+1} = \begin{cases} 0 & \text{si } 2u_i < y \\ 1 & \text{si } 2u_i \geq y \end{cases}$$

A partir de cette suite, on déduit que :

$$u_i = 2^i x - y \sum_{j=1}^i q_j 2^{i-j} \quad \text{avec} \quad 0 \leq u_i < y \quad \text{pour } i \geq 0$$

$$\text{d'où} \quad \frac{x}{y} = \sum_{j=1}^i q_j 2^{-j} + 2^{-i} \frac{u_i}{y} = \sum_{j=1}^i q_j 2^{-j} \quad \text{quand } i \rightarrow \infty$$

Autrement dit :  $\frac{x}{y}$  s'écrit :  $0, q_1 q_2 q_3 \dots q_i q_{i+1} \dots q_N$  en base 2

5.1) On suppose que l'on dispose d'une procédure algorithmique :

**Procédure difference**(Tx, Ty : TABGR , variable Tz : TABGR)

tel que :  $z = x - y$  avec  $x \geq y$ . (On rappelle que les réels  $x$ ,  $y$  et  $z$  sont représentés respectivement par Tx, Ty et Tz de type TABGR).

Ecrire une procédure algorithmique *itère* dont l'entête est :

**Procédure itère**(Tx, Ty : TABGR , variable Tz : TABGR , variable tropetit : booléen, variable erreur : booléen)

Cette procédure permet de calculer Tz, tropetit et erreur tels que :

$$\text{erreur} = \begin{cases} \text{faux} & \text{si } x < \frac{1}{2}, \text{ dans ce cas} \\ \text{vrai} & \text{si } x \geq \frac{1}{2} \end{cases} \begin{cases} \text{si } 2x - y < 0 \text{ alors} \\ \text{si } 2x - y \geq 0 \text{ alors} \end{cases} \begin{cases} \text{tropetit} = \text{vrai} \\ z = 2x \end{cases} \text{ et} \begin{cases} \text{tropetit} = \text{faux} \\ z = 2x - y \end{cases} \text{ et}$$

**N.B** : Cette procédure permet de calculer un terme  $u_i$  de la suite  $u$  (représenté par Tz) et déduire, selon la valeur de la valeur de tropetit, le terme  $q_{i+1}$  de la suite  $q$  lors de la division de  $x$  par  $y$ .

5.2) Ecrire une procédure algorithmique *divise* qui, à partir de deux réels  $x$  et  $y$  tels que  $x < y < \frac{1}{2}$ , calcule un réel  $z$  formé des  $N$  premiers chiffres binaires (0 ou 1) de  $\frac{x}{y}$ . Cette procédure fait appel à la procédure *itère* et a comme entête :

**Procédure divise** (N : entier, Tx, Ty : TABGR , variable Tz : TABGR, variable correct : booléen)

$$\text{avec } \text{correct} = \begin{cases} \text{vrai} & \text{si } x < y < \frac{1}{2} \\ \text{faux} & \text{sinon} \end{cases}$$

6) Si la contrainte  $x < y < \frac{1}{2}$  n'est pas satisfaite, indiquer sommairement (on ne demande pas l'écriture d'une procédure) comment on peut calculer  $\frac{x}{y}$ .