



Concours Mathématique et Physique, Physique et Chimie et Technologie Epreuve d'Informatique

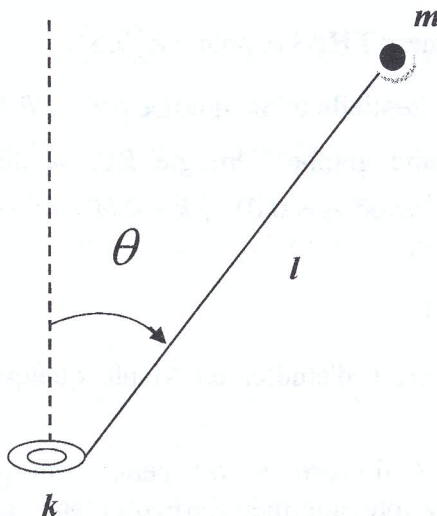
Date : Vendredi 27 Mai 2016 Heure : 14 H Durée : 2 H Nombre de pages : 5

Barème : EXERCICE 1 (MAPLE) : 4 points
EXERCICE 2 (MAPLE) : 6 points
PROBLEME (ALGORITHMIQUE) : 10 points

DOCUMENTS NON AUTORISES
L'USAGE DES CALCULATRICES EST INTERDIT

EXERCICE 1 (MAPLE)

Le schéma suivant représente un métronome :



où :

m : masse du point matériel de rayon négligeable

l : longueur de la tige de masse négligeable

k : constante de raideur du ressort spirale

θ : angle entre la tige et la verticale

L'énergie cinétique **ECS** et l'énergie potentielle **EPS** de la sphère sont données respectivement par les expressions suivantes :

$$ECS(t) = \frac{1}{2} ml^2 \left(\frac{\partial \theta}{\partial t} \right)^2$$

$$EPS(t) = mgl \cos(\theta(t))$$

où t représente le temps et g la constante universelle de gravitation.

L'énergie potentielle **EPR** du ressort est donnée par :

$$EPR(t) = \frac{1}{2} k \theta^2(t)$$

L'énergie du métronome **EM** est donnée par :

$$EM = ECS + EPS + EPR \quad (1)$$

Le comportement du métronome est défini par l'équation différentielle ci-dessous :

$$\frac{\partial(EM)}{\partial t} = 0 \quad (2) \quad \text{avec les conditions initiales : } \begin{cases} \theta(0) = a \\ \theta'(0) = b \end{cases} \quad (3)$$

Les équations (2) et (3) constituent le modèle du mouvement du métronome.

Donner les instructions MAPLE permettant de :

1. définir les fonctions **ECS**, **EPS**, **EPR** et **EM** dépendantes de t ;
2. définir **Eq**, la fonction dérivée de **EM** ;
3. définir **Eqd**, l'équation différentielle donnée par (2) ;
4. affecter dans **ci1** et **ci2**, respectivement les conditions initiales données par (3) ;
5. résoudre formellement l'équation **Eqd** avec les conditions **ci1** et **ci2** ;
6. récupérer dans **THETA** la résolution numérique de l'équation **Eqd** avec les conditions **ci1** et **ci2** pour les valeurs $a = \frac{\pi}{6}$ et $b = 0$ en prenant $g = 9.81$, $m = 0.2$, $k = 0.3$ et $l = 0.15$;
7. représenter graphiquement **THETA** pour $t \in [0, 5]$;
8. définir **EP**, l'énergie potentielle totale donnée par : $EP = EPS + EPR$;
9. représenter sur le même graphe l'énergie **EP**, sa dérivée première et sa dérivée seconde, pour $t \in [0, 5]$ avec $m = 0.015$, $k = 0.002$ et $l = 0.15$.

EXERCICE 2 (MAPLE)

Dans cet exercice, on se propose d'étudier en Maple quelques propriétés associées à des matrices carrées particulières.

1. Une matrice carrée M d'ordre n est celle d'un projecteur si on a $M^2 = M$. Ecrire une procédure Maple, nommée **EstProjecteur**, qui prend comme paramètre une matrice carrée **M** et retourne *true* si **M** est la matrice d'un projecteur et *false* sinon.

La procédure a pour entête : **EstProjecteur := proc(M :: matrix)**

2. On dit qu'une matrice carrée M d'ordre n est nilpotente s'il existe un entier naturel k , appelé indice de nilpotence, tel que M^k est égale à la matrice nulle d'ordre n avec $k \leq n$. L'indice de nilpotence est le plus petit $k \leq n$.

Ecrire une procédure Maple, nommée **IndNilpotence**, qui prend comme paramètre une matrice carrée **M**, supposée non nulle, et retourne l'indice de nilpotence s'il existe et 0 sinon.

La procédure a pour entête : **IndNilpotence := proc(M :: matrix)**

3. On dit qu'une matrice carrée M d'ordre n est circulante si chaque ligne d'indice i (avec $2 \leq i \leq n$) est la permutation circulaire vers la droite de la ligne d'indice $i-1$.

Exemple : $M = \begin{pmatrix} 1 & a & b & c \\ c & 1 & a & b \\ b & c & 1 & a \\ a & b & c & 1 \end{pmatrix}$ est circulante.

Ecrire une procédure Maple, nommée **EstCirculante**, qui prend comme paramètre une matrice carrée M et retourne *true* si M est circulante et *false* sinon.

La procédure a pour entête : **EstCirculante:=proc(M :: matrix)**

4. Sachant que le produit de deux matrices circulantes est une matrice circulante. Ecrire une procédure Maple, nommée **ProduitCirculant**, qui prend comme paramètre deux matrices carrées A et B de même ordre, supposées circulantes, et retourne leur produit matriciel C en exploitant la propriété des matrices circulantes.

La procédure a pour entête : **ProduitCirculant:=proc(A :: matrix , B :: matrix)**

PROBLEME (ALGORITHMIQUE)

La notion de permutation formalise la notion intuitive de réarrangement d'objets.

Soient n un entier naturel non nul et E_n la suite d'entiers distincts $x_1, x_2, \dots, x_n$ où chaque x_i appartient à l'intervalle $[1, n]$.

On définit une permutation comme un réarrangement des éléments de E_n dans E_n en appliquant, à chacun des éléments de E_n , une fonction f bijective.

Une permutation est donc une fonction bijective de l'intervalle $[1, n]$ dans l'intervalle $[1, n]$.

Chaque permutation sera représentée par un tableau d'entiers T à une dimension dont la case d'indice 0 stocke n et les différentes cases $T[i]$ (c'est-à-dire les $f(i)$) stockent les x_i , avec $1 \leq x_i \leq n$ et $1 \leq i \leq n$.

Exemple : T est une permutation pour $n = 8$

f	T	8	3	5	1	8	4	7	6	2	...	
E_n		0	1	2	3	4	5	6	7	8		NMAX

Travail demandé

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

CONSTANTE NMAX = 1000

TYPE PERMUTATION = tableau [0 .. NMAX] de entier

On désigne par E les paramètres passés par valeur et par S les paramètres passés par variable dans les entêtes des procédures et des fonctions.

Question 1

Ecrire une fonction algorithmique, nommée **estPermutation**, qui prend en entrée un tableau **T** et qui retourne *vrai* si **T** représente une permutation et *faux* sinon. La fonction a pour entête :

Fonction estPermutation(E T : PERMUTATION) : boolean

Question 2

Ecrire une fonction algorithmique, nommée **estIdentite**, qui prend en entrée une permutation **T** et qui retourne *vrai* si **T** est une permutation identité et *faux* sinon.

Exemple : **T** est une permutation identité pour $n=8$

T	8	1	2	3	4	5	6	7	8	...	
	0	1	2	3	4	5	6	7	8		NMAX

La fonction a pour entête :

Fonction estIdentite(E T : PERMUTATION) : boolean

Question 3

Une transposition est une permutation qui, à partir d'une permutation identité, échange uniquement deux éléments et laisse les autres inchangés.

Ecrire une fonction algorithmique, nommée **estTransposition**, qui prend en entrée une permutation représentée par un tableau **T** et retourne *vrai* si **T** est une transposition et *faux* sinon.

Exemple : **T** est une transposition associée à une permutation pour $n=8$

T	8	1	2	7	4	5	6	3	8	...	
	0	1	2	3	4	5	6	7	8		NMAX

La fonction a pour entête :

Fonction estTransposition(E T : PERMUTATION) : boolean

Question 4

Ecrire une procédure algorithmique, nommée **saisiePermutation**, qui saisit la taille **n** de la permutation, avec $1 < n \leq NMAX$, puis saisit dans un tableau **T** une permutation quelconque de taille **n** (qui n'est pas forcément une identité). La procédure a pour entête :

Procedure saisiePermutation(S n : entier, S T : PERMUTATION)

Question 5

On rappelle que la composée $f \circ g$ de deux fonctions est donnée par $f(g(x))$.

Ecrire une procédure algorithmique, nommée **composer**, qui prend en entrée deux permutations **T1** et **T2**, supposées de même taille, et renvoie dans un tableau **T3** la permutation composée de **T1** par **T2**. La procédure a pour entête :

Procedure composer(E T1, T2 : PERMUTATION, S T3 : PERMUTATION)

Question 6

On rappelle que la fonction réciproque f^{-1} d'une bijection f est définie comme associant x à $f(x)$.

Ecrire une procédure algorithmique, nommée **reciproque**, qui prend en entrée une permutation **T** et renvoie dans un tableau **T1** sa permutation réciproque.

Exemple : T1 est la permutation réciproque de la permutation T pour n=8

T	8	3	5	1	8	4	7	6	2	...	
	0	1	2	3	4	5	6	7	8		NMAX

T1	8	3	8	1	5	2	7	6	4	...	
	0	1	2	3	4	5	6	7	8		NMAX

La procédure a pour entête :

Procédure reciproque(E T : PERMUTATION, S T1 : PERMUTATION)

Question 7

On définit l'ordre d'une permutation T comme étant le plus petit entier naturel k non nul telle que la composée de T avec lui-même k fois est la permutation identité.

En utilisant impérativement la fonction **estIdentite** et la procédure **composer**, écrire une fonction algorithmique, nommée **ordre**, qui prend en entrée une permutation T et retourne son ordre k. La fonction a pour entête :

Fonction ordre(E T : PERMUTATION) : entier

Question 8

La période d'un indice i pour une permutation T est définie comme le plus petit entier naturel k non nul telle que la composée, k fois, de T[i] donne i.

Ecrire une fonction algorithmique, nommée **periode**, qui prend en entrée une permutation T et un indice i et retourne la période de i pour T. La fonction a pour entête :

Fonction periode(E T : PERMUTATION , E i : entier) : entier

Question 9

L'orbite d'un indice i pour une permutation T est l'ensemble des indices j tel qu'il existe k (k ne dépasse jamais la période de i), avec la composée, k fois, de T[i] donne j.

Ecrire une fonction algorithmique, nommée **estDansOrbite**, qui prend en entrée une permutation T ainsi que deux indices i et j et retourne *vrai* si l'indice j est dans l'orbite de l'indice i et *faux* sinon. La fonction a pour entête :

Fonction estDansOrbite(E T : PERMUTATION, E i, j : entier) : boolean

Question 10

Un cycle est une permutation dont exactement une des orbites est de taille strictement supérieure à un.

Ecrire une fonction algorithmique, nommée **estCycle**, qui prend en entrée une permutation T et retourne *vrai* si T est un cycle et *faux* sinon. La fonction a pour entête :

Fonction estCycle(E T : PERMUTATION) : boolean

Exemple : T1 est la permutation réciproque de la permutation T pour n=8

T	8	3	5	1	8	4	7	6	2	...	
	0	1	2	3	4	5	6	7	8		NMAX
T1	8	3	8	1	5	2	7	6	4	...	
	0	1	2	3	4	5	6	7	8		NMAX

La procédure a pour entête :

Procédure reciproque(E T : PERMUTATION, S T1 : PERMUTATION)

Question 7

On définit l'ordre d'une permutation T comme étant le plus petit entier naturel k non nul telle que la composée de T avec lui-même k fois est la permutation identité.

En utilisant impérativement la fonction **estIdentite** et la procédure **composer**, écrire une fonction algorithmique, nommée **ordre**, qui prend en entrée une permutation T et retourne son ordre k. La fonction a pour entête :

Fonction ordre(E T : PERMUTATION) : entier

Question 8

La période d'un indice i pour une permutation T est définie comme le plus petit entier naturel k non nul telle que la composée, k fois, de T[i] donne i.

Ecrire une fonction algorithmique, nommée **periode**, qui prend en entrée une permutation T et un indice i et retourne la période de i pour T. La fonction a pour entête :

Fonction periode(E T : PERMUTATION , E i : entier) : entier

Question 9

L'orbite d'un indice i pour une permutation T est l'ensemble des indices j tel qu'il existe k (k ne dépasse jamais la période de i), avec la composée, k fois, de T[i] donne j.

Ecrire une fonction algorithmique, nommée **estDansOrbite**, qui prend en entrée une permutation T ainsi que deux indices i et j et retourne *vrai* si l'indice j est dans l'orbite de l'indice i et *faux* sinon. La fonction a pour entête :

Fonction estDansOrbite(E T : PERMUTATION, E i, j : entier) : boolean

Question 10

Un cycle est une permutation dont exactement une des orbites est de taille strictement supérieure à un.

Ecrire une fonction algorithmique, nommée **estCycle**, qui prend en entrée une permutation T et retourne *vrai* si T est un cycle et *faux* sinon. La fonction a pour entête :

Fonction estCycle(E T : PERMUTATION) : boolean