



**Concours Mathématique et Physique, Physique et Chimie et Technologie
Alternative de correction de l'épreuve d'Informatique**

Barème sur 100

EXERCICE 1 (20 points)

1) 5 pt = 1,25 * 4

```
> ECS:=t->(1/2)*m*l^2*diff(theta(t),t)^2 ;  
EPS:=t->m*g*l*cos(theta(t)) ;  
EPR:=t->(1/2)*k*(theta(t))^2 ;  
EM:=ECS + EPS + EPR ;
```

2) 1,25 pt

```
> Eq:=D(EM) ;
```

3) 2,5 pt

```
> Eqd:=Eq(t)=0;
```

4) 1,25 pt = 0,5 + 0,75

```
> ci1:=theta(0)=a ; ci2:=D(theta)(0)=b ;
```

5) 2,5 pt

```
> dsolve({eqd , ci1 , ci2} , theta(t)) ;
```

6) 2,5 pt

```
> THETA := dsolve({Eqd , subs(a=Pi/6 , ci1) ,  
subs(b=0 , ci2) , g=9.81 , m=0.2 , k=0.3 , l=0.15} ,  
theta(t) , numeric) ;
```

7) 1,25 pt

```
> with(plots) ; odeplot(THETA , [t , theta(t)] ,  
t=0..5) ;
```

8) 1,25 pt

```
> EP:=EPS + EPR ;
```

9) 2,5 pt

```
> m , k , l := 0.015 , 0.002 , 0.15;  
plot({EP , D(EP) , (D@@2)(EP)} , 0..5);
```

EXERCICE 2 (30 points)

1) 5 pts

```
> EstProjecteur:=proc(M::matrix)  
equal(evalm(M^2),M) ;  
end proc;
```

2) 7,5 pts

```
> IndNilpotence:=proc(M::matrix)
local k;
for k from 1 to rowdim(M) do
if equal(evalm(M^k),Matrix(rowdim(M), rowdim(M)))
then RETURN(k) fi;
od;
RETURN(0);
end proc;
```

3) 7,5 pts

```
> EstCirculante:=proc(M::matrix)
local n,i,j;
n:=rowdim(M);
for i from 1 to n-1 do
if M[i,n] <> M[i+1,1] then RETURN(false)
else
for j from 2 to n do
if M[i+1,j] <> M[i,j-1] then RETURN(false) fi;od;
fi;od;
RETURN(true);
end proc;
```

4) 10 pts

```
> ProduitCirculant:=proc(A::matrix,B::matrix)
local n,C,i,j,k,s;
n:=rowdim(A);
C:=matrix(n,n);
for j to n do
s:=0;
for k to n do
s:=s+A[1,k]*B[k,j];od;
C[1,j]:=s;
od;
for i from 2 to n do
C[i,1]:=C[i-1,n];
for j from 2 to n do
C[i,j]:=C[i-1,j-1];od;od;
RETURN(evalm(C));
end proc;
```

PROBLEME (50 points)

Question 1 5 pts

Fonction estPermutation(E T :
PERMUTATION) : booleen

Variable i : entier

P : tableau [1..NMAX] de

booleen

DEBUT

Pour i de 1 à T[0] faire

P[i] ← FAUX

Fin Pour

Pour i de 1 à T[0] faire

Si T[i] < 1 OU T[i] > T[0] Alors

Retourner(FAUX)

Sinon Si P[T[i]] = Vrai Alors

Retourner(FAUX)

Sinon P[T[i]] ← Vrai Fin Si Fin Si

Fin Pour

Retourner(FAUX)

Fin

Question 2 5 pts

Fonction estIdentite(E T :
PERMUTATION) : booleen

Variable i : entier

DEBUT

Pour i de 1 à T[0] faire

Si T[i] <> i Alors

Retourner(FAUX) Fin Si

Fin Pour

Retourner(FAUX)

Fin

Question 3 5 pts

Fonction estTransposition(E T :
PERMUTATION) : booleen

Variable Cpt, i : entier

DEBUT

Cpt ← 0

Pour i de 1 à T[0] faire

Si T[i] <> i Alors Cpt ← Cpt + 1

Fin Si

Fin Pour

Question 4 5 pts

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

Variable i : entier

DEBUT

Repeter

Lire (n)

Jusqu'à n > 1 ET n <= NMAX

T[0] ← n

Repeter

Si Cpt = 2 Alors Retourner(VRAI)
Sinon Retourner(FAUX) Fin Si
Fin

Pour i de 1 à T[0] faire
Lire (T[i])
Fin Pour
Jusqu'à estPermutation(T)
Fin

Question 5 5 pts

Procédure composer(E T1,T2 :

PERMUTATION , S T3 :

PERMUTATION)

Variable i : entier

DEBUT

Pour i de 1 à T1[0] faire

T3[i] ← T1[T2[i]]

Fin Pour

Fin

Question 6 5 pts

Procédure reciproque(E T:

PERMUTATION , S T1 :

PERMUTATION)

Variable i : entier

DEBUT

Pour i de 1 à T [0] faire

T1[T [i]] ← i

Fin Pour

Fin

Question 7 5 pts

Fonction ordre(E T:

PERMUTATION) : entier

Variable k,i : entier

P : PERMUTATION

DEBUT

Pour i de 1 à T [0] faire

P[i] ← T[i]

Fin Pour

k ← 1

Tant Que NON(estIdentite(P))

Faire

composer(P , T , P)

k ← k + 1

Question 8 5 pts

Fonction periode(E T:

PERMUTATION , E i: entier) : entier

Variable k,i : entier

P : PERMUTATION

DEBUT

Pour i de 1 à T [0] faire

P[i] ← T[i]

Fin Pour

k ← 1

Tant Que P[i] <> i Faire

composer(P , T , P)

k ← k + 1

Fin Tant Que

Fin Tant Que
Retourner(k)
Fin

Retourner(k)
Fin

Question 9 5 pts

Fonction estDansOrbite(E T:
PERMUTATION, E i, j : entier) :
booléen
Variable p, p1, k : entier
DEBUT
p ← periode(T, i)
p1 ← T[i]
Pour k de 1 à p faire
Si p1 = j Alors Retourner(VRAI)
Fin Si
p1 ← T[p1]
Fin Pour
Retourner(FAUX)
Fin

Question 10 5 pts

Fonction estCycle(E T:
PERMUTATION) : booléen
Variable k, i : entier
B : booléen
DEBUT
k ← 0
B ← VRAI
Pour i de 1 à T[0] Faire
Si T[i] <> i ET B Alors
B ← FAUX
k ← i
Sinon Si T[i] <> i ET NON(B)
Alors
Si NON(estDansOrbite(T, k, i))
Alors Retourner(FAUX)
Fin Si Fin Si.
Fin Pour
Retourner(VRAI)
Fin