



Concours Toutes Options

Alternative de correction de l'épreuve d'Informatique

PROBLEME 1 (11 points)

1) 3 pts = 0.75 par question

```
1.1) > equal(Kronecker(evalm(A+B) , evalm(E+F)) ,  
evalm(Kronecker(A , E) + Kronecker(A , F) +  
Kronecker(B , E) + Kronecker(B , F)));
```

```
1.2) > equal(transpose(Kronecker(A,B)) ,  
Kronecker(transpose(A) , transpose(B)));
```

```
1.3) > evalb(trace(Kronecker(A , B)) = trace(A) *  
trace(B));
```

```
1.4) > evalb(det(Kronecker(A , B)) = det(A) ^ rowdim(B)  
* det(B) ^ rowdim(A));
```

2) 3 pts

```
> Kronecker:=proc(A::matrix , B::matrix)  
local C,i,j,k,l;  
C:=matrix(rowdim(A)* rowdim(B) , coldim(A)*  
coldim(B));  
for i to rowdim(B) do for j to coldim(B) do  
for k to rowdim(A) do for l to coldim(A) do  
C[(i - 1) * m + k , (j - 1) * n + l]:=A[k , l] *  
B[i , j];  
od;od;od;od;  
RETURN(evalm(C));  
end proc;
```

3) 3 pts = 0.5 par instruction

```
> Sylvester1:=proc(A::matrix , B::matrix , C::matrix)  
local K1,K2,M,v,y,X ;  
K1:=Kronecker(diag(1 $ rowdim(B)) , A);  
K2:=Kronecker(transpose(B) , diag(1 $ rowdim(A)));  
M:=evalm(K1 + K2);  
v:=convert(C , vector);  
y :=linsolve(M , v) ;  
X :=matrix(n , m , y) ;  
end proc ;
```

4) 2 pts

```
> Sylvester2:=proc(A::matrix , B::matrix , C::matrix)
local E1,E2 ;
E1:={eigenvals(A)};E2:={eigenvals(-B)};
if evalb((E1 intersect E2) = {}) then Sylvester1(A
, B , C)
else ERROR("la matrice M est singulière") fi;
end proc ;
```

PROBLEME 2 (9 points)

1) 0.5 pts

```
Fonction Max(T : VECT , n :
entier) : entier
Variable M , i : entier
DEBUT
M ← T[1]
Pour i de 2 à n faire
Si T[i] > M Alors M ← T[i] Fin Si
Fin Pour
Retourner(M)
Fin
```

2) 0.5 pts

```
Fonction Pos_Max(T : VECT , n :
entier) : entier
Variable M , p , i : entier
DEBUT
M ← T[1]
p ← 1
Pour i de 2 à n faire
Si T[i] > M Alors M ← T[i]
p ← i Fin Si
Fin Pour
Retourner(p)
Fin
```

3) 1 pt

```
Procedure Min_Max(T : VECT ,
n : entier , variable M1,M2 : entier)
Variable p , i : entier
DEBUT
p ← Pos_Max(T , n)
M1 ← T[p]
M2 ← M1

Pour i de 1 à n faire
Si T[i] < M2 Alors M2 ← T[i] Fin Si
Fin Pour
Fin
```

4) 1 pt

```
Fonction Verif(T : VECT , n :
entier) : booleen
Variable i , j : entier
DEBUT
Pour i de 2 à n faire
Pour j de 1 à i-1 faire
Si T[j] = T[i] Alors
Retourner(faux) Fin Si
Fin Pour
Fin Pour
Retourner(vrai)
Fin
```


5) 2 pts

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

Variable M, i : entier

DEBUT

Si T[1] > T[2] alors M1 ← T[1]
M2 ← T[2]

Sinon M1 ← T[2]
M2 ← T[1] Fin Si

Pour i de 3 à n faire

Si T[i] > M1 alors M2 ← M1
M1 ← T[i]

Sinon Si T[i] > M2 alors
M2 ← T[i] Fin Si Fin Si

Fin Pour

Retourner(M2)

Fin

6) 2 pts

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

Variable T1, T2 : VECT

i, j, t : entier

DEBUT

t ← n

Pour i de 1 à t faire

T1[i] ← T[i] Fin Pour

Tant que t > 1 faire

j ← 1

Si t mod 2 = 0 alors

Pour i de 1 à t (pas = 2)
faire

Si T1[i] < T1[i + 1] alors

T2[j] ← T1[i + 1] sinon T2[j] ← T1[i]

Fin Si

j ← j + 1

Fin Pour

t ← t div 2

Sinon

Pour i de 1 à t - 1 (pas = 2)

faire

Si T1[i] < T1[i + 1] alors

T2[j] ← T1[i + 1] sinon T2[j] ← T1[i]

Fin Si

j ← j + 1

Fin Pour

T2[(t div 2) + 1] ← T1[t]

t ← (t div 2) + 1

Fin Si

Pour i de 1 à t faire

T1[i] ← T2[i] Fin Pour

Fin Tant que

M ← T1[i]

Fin

7) 1 pt

Fonction Max2_Bis(T : VECT, n : entier) : entier
Variable T1 : VECT
X, i, j : entier

DEBUT

Max_Bis(T, n, X)

j ← 1

Pour i de 1 à n faire

Si X <> T[i] alors T1[j] ← T[i]
j ← j + 1 Fin

Si

Fin Pour

Max_Bis(T1, j - 1, X)

Retourner(X)

Fin

8) 1 pt

Procédure Min_Max_Bis(T : VECT, n : entier, variable M1, M2 : entier)
Variable p, i : entier

DEBUT

Pour i de 1 à n faire

T1[i] ← T[i] Fin Pour

Pour i de 1 à n - 1 (pas = 2) faire

Si T1[i] < T1[i + 1] Alors
T1[i] ← T1[i + 1]

T1[i

+ 1] ← T1[i] Fin Si

Fin Pour

M1 ← T1[1]

M2 ← T1[2]

Pour i de 3 à n - 1 (pas = 2) faire

Si T1[i] > M1 Alors M1 ← T1[i]

Fin Si

Si T1[i + 1] < M2 Alors

M2 ← T1[i + 1] Fin Si

Fin Pour

Fin

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 2^{ème} 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 :

Procédure 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 $M2 = 4$