



Concours Toutes Options
Alternative de correction de l'épreuve d'Informatique

Barème sur 100

PROBLEME 1 (50 points)

1) 2.5 pts

> n:=nops(X);

2) 2.5 pts

> f:=unapply(E, (op(X)));

3) 5 pts

> V:=seq(diff(f(op(X)), op(i,X)), i=1..n); ou bien

> V:=seq(diff(f(op(X)), X[i]), i=1..n); ou bien

> V:=seq((D[i])(f)(op(X)), i=1..n); ou bien

> V:=[];

for i from 1 to n do

V:=[op(V), diff(f(op(X)), X[i])]; od;

4) 5 pts

> P:={solve({V}, {op(X)})};

5) 5 pts (2.5 + 2.5)

> m:=nops(P);

if m > 0 then 0.25 pt

6) 5 pts

> g:=(i,j)->diff(f(op(X)), op(i,X), op(j,X)); ou bien

> g:=(i,j)->diff(f(op(X)), X[i], X[j]); ou bien

> g:=(i,j)->(D[i,j])(f)(op(X));

On accepte également la définition avec la commande unapply

7) 2.5 pts

> H:=matrix(n,n,g);

8) 2.5 pts

> VP:=[eigenvals(H)]; ou bien

> VP:=[solve(charpoly(H,lambda), lambda)];

9) 5 pts

> LVP:=[seq(subs(op(i,P), VP), i=1..m)]; ou bien

> LVP:=[];

for i from 1 to m do

LVP:=[op(LVP), subs(op(i,P), VP)]; od;

On accepte également la commande eval

10) 5 pts

```
> signe:=x->piecewise(x>0,1,x<0,-1,0); ou bien
> signe:=x->if x>0 then 1 elif x<0 then -1 else 0
fi ;
```

11) 5 pt

```
> LS:=[seq(map(signe,LVP[i]),i=1..m)]; ou bien
> LS:=[seq([seq(signe(LVP[i][j]),j=1..n)],i=1..m)]; ou
bien
> LS:=[];
  for i from 1 to m do
    L:=[];
    for j from 1 to n do
      L:=[op(L),signe(LVP[i][j])]; od;
    LS:=[op(LS),L]; od;
```

12) 5 pts (4+1)

```
L:=[];
for i from 1 to m do
a:=LS[i][1];
test1:=true;
test2:=true;
for j from 1 to n do
test1:=test1 and evalb(a=LS[i][j]);
test2:=test2 and evalb(LS[i][j]<>0);od;
if not(test2) then L:=[op(L),[op(i,P),`point
dégénéré`]];
elif a=1 and test1 then L:=[op(L),[op(i,P),`minimum
local`]];
elif a=-1 and test1 then L:=[op(L),[op(i,P),`maximum
local`]];
else L:=[op(L),[op(i,P),`point col`]];fi;od;

SPC:=m,L;
```

PROBLEME 2 (50 points = 6.25 par question)

1)

```
FONCTION Taille(): entier
VARIABLE E: entier
DEBUT
REPETER
Lire(E)
JUSQU'A E >= 1 ET E <= NMAX
RETOURNER (E) ou {Taille←E}
FIN
```

2)

```
PROCEDURE SaisieTxt(E n:
entier, S tab: TEXTE)
VARIABLE i: entier
DEBUT
tab[0]←n
POUR i de 1 à n FAIRE
  REPETER
    Lire(tab[i])
  JUSQU'A tab[i] >= 1 ET
  tab[i] <= 26
FN POUR
FIN
```

3)
FONCTION VerifMot (E tab ,
 mot : TEXTE , E k : entier) :
 booléen
VARIABLE flag : booléen
 i : entier

DEBUT

flag ← VRAI
 i ← 1
TANT QUE flag **ET** i ≤ mot[0]
FAIRE
 SI mot[i] < > tab[i + k - 1]
ALORS flag ← FAUX **SI NON**
 i ← i + 1 **FIN SI**
FIN TANT QUE
RETOURNER(flag)

FIN

ou bien

Pour i de 1 à mot[0] **FAIRE**
 SI mot[i] < > tab[i + k - 1]
ALORS **RETOURNER**(faux) **FIN SI**
FIN POUR
RETOURNER(vrai)

4)
FONCTION RechercheMot (E
 tab, mot : TEXTE) : entier
VARIABLE flag : booléen
 i : entier

DEBUT

flag ← FAUX
 i ← 1
TANT QUE NON(flag) **ET** i ≤
 tab[0] - mot[0] **FAIRE**
 flag ← VerifMot (tab , mot , i)
 i ← i + 1
FIN TANT QUE
 SI flag **ALORS** **RETOURNER** (i - 1
) **SINON** **RETOURNER** (-1) **FIN SI**

FIN

ou bien

Pour i de 1 à tab[0] - mot[0]
FAIRE
 SI VerifMot (tab , mot , i) **ALORS**
RETOURNER(i) **FIN SI**
FIN POUR
RETOURNER(-1)

5)
PROCEDURE TabOccur(E tab ,
 mot : TEXTE , S occ : TEXTE)
VARIABLE i , j : entier

DEBUT

j ← 0
POUR i de 1 à tab[0] -
 mot[0] + 1 **FAIRE**
 SI VerifMot (tab , mot , i) **ALORS**
 j ← j + 1
 occ[j] ← i **FIN SI**
FIN POUR
 occ[0] ← j

FIN

6)
FONCTION MotRecouv(E mot ,
 occ : TEXTE) : booléen
VARIABLE i : entier

DEBUT

POUR i de 2 à occ[0]
FAIRE
 SI occ[i] - occ[i - 1] < mot[0]
ALORS **RETOURNER**(VRAI)
FIN SI
FIN POUR
RETOURNER (B)

FIN

7)
 PROCEDURE
 FrequenceLettres(E tab : TEXTE ,
S Freq : FREQUENCE)
 VARIABLE i , j : entier
 mot , occ : TEXTE

DEBUT

POUR i de 1 à 26 FAIRE
 mot[0] ← 1
 mot[1] ← i
 TabOccur(tab , mot , occ)
 POUR j de 0 à occ[0]
 FAIRE
 Freq[i , j] ← occ[j]
 FIN POUR
 FIN POUR

FIN

ou bien

POUR i de 1 à 26 FAIRE
 Freq[i , 0] ← 0
 FIN POUR
 POUR i de 1 à tab[0]
 FAIRE
 mot[0] ← 1
 mot[1] ← i
 TabOccur(tab , mot , occ)
 POUR j de 0 à occ[0]
 FAIRE
 Freq[i , j] ← occ[j]
 FIN POUR
 FIN POUR

8)
 PROCEDURE LettresPlusFreq(E
 tab : TEXTE , E Freq :
 FREQUENCE , S tabF :
 FREQMAX)
 VARIABLE i , m : entier

DEBUT

m ← 0
 POUR i de 1 à 26 FAIRE
 SI Freq[i , 0] > m ALORS
 m ← Freq[i , 0] FIN SI
 FIN POUR
 tabF[0] ← m
 j ← 0
 POUR i de 1 à 26 FAIRE
 SI Freq[i , 0] = m ALORS
 j ← j + 1
 tabF[j] ← i
 FIN SI
 FIN POUR
FIN