Problème : Matrice – Enregistrement – Décomposition en modules

On se propose de chercher dans une matrice **M**, les plus longues séquences d'éléments **juxtaposés** dont la somme est égale à **zéro**, en procédant comme suit :

- remplir une matrice **M** de dimension **LxC** (avec $2 < L \leq 24$ et $2 < C \leq 24$) par des entiers non nuls.
- chercher, pour chaque ligne, toutes les séquences d'éléments **juxtaposés** dont la somme est égale à 0 et les stocker dans un tableau d'enregistrements **T**. Chaque enregistrement contient trois champs : **le numéro de la ligne** (NL), **l'indice de la colonne représentant le début de la séquence** (ICD) et **l'indice de la colonne représentant la fin de la séquence** (ICF).
- **afficher à l'écran :**
  - ✓ **Le nombre d'éléments de la plus longue séquence**
  - ✓ toutes les plus longues séquences d'éléments. Chacune sera représentée, dans une ligne à part, par les valeurs de NL, ICD et ICF, séparés par le caractère "**#**".

Exemple :
Pour la matrice **M** de dimension **5x6** suivante :

	1	2	3	4	5	6
1	**6**	**-2**	**-2**	**-1**	**-1**	**4**
2	**-1**	**2**	**2**	**-3**	**-2**	**9**
3	**-5**	**6**	**-2**	**-2**	**-1**	**-1**
4	**-2**	**3**	**2**	**-1**	**4**	**10**
5	**-6**	**1**	**2**	**2**	**1**	**-7**

- ✓ Dans la 1ère ligne, il y a 2 séquences d'éléments juxtaposés dont la somme est égale à 0 :
  - La séquence ayant comme ICD la valeur **1** et comme ICF la valeur **5**.
  - La séquence ayant comme ICD la valeur **3** et comme ICF la valeur **6**.
- ✓ Dans la 2ème ligne, il y a une seule séquence d'éléments juxtaposés dont la somme est égale à 0 :
  - La séquence ayant comme ICD la valeur **1** et comme ICF la valeur **4**.
- ✓ Dans la 3ème ligne, il y a une seule séquence d'éléments juxtaposés dont la somme est égale à 0 :
  - La séquence ayant comme ICD la valeur **2** et comme ICF la valeur **6**.
- ✓ Dans la 4ème ligne, il n'y a aucune séquence d'éléments juxtaposés dont la somme est égale à 0.
- ✓ Dans la 5ème ligne, il y a une seule séquence d'éléments juxtaposés dont la somme est égale à 0 :
  - La séquence ayant comme ICD la valeur **1** et comme ICF la valeur **5**.

Le contenu du tableau **T** sera :

T

	1	2	3	4	5	
1	1	2	**3**	**5**	← *Numéro de la ligne*
1	3	1	**2**	**1**	← *Indice du début*
5	6	4	**6**	**5**	← *Indice de fin*

Le programme affiche :

```
Le nombre d'éléments de la plus longue séquence = 5
1#1#5
3#2#6
5#1#5
```

Correction :

M : est une matrice de 24 Lignes x 24 Colonnes (Matrice d'entiers)

L : Nombre de Lignes avec (2 ≤ L ≤ 24)

C : Nombre de Colonnes avec (2 ≤ C ≤ 24)

Travail demandé :

1. *Remplir la matrice M par des entiers non nuls*
2. *Pour chaque ligne :*
    a. *chercher les séquences d'éléments juxtaposées dont la somme est nul*
    b. *les informations relatives à une séquence nulle sont :*
        i. ***NL** : Numéro de la ligne de la séquence dans M*
        ii. ***ICD** : Indice Colonne de Début de la séquence*
        iii. ***ICF** : Indice Colonne de Fin de la séquence*

⇨ *Ces trois informations seront enregistrées dans un tableau d'enregistrement où chaque enregistrement comporte trois champs (**NL** , **ICD** , **ICF**)*

3. *à la fin de la tâche 2 ➔ Le tableau T contient les informations de toutes les séquences juxtaposées nul existantes dans la matrice M*
4. *en se référant au Tableau T : Déterminer le Nombre d'élément de la Plus Longue Séquence (NPLS)*
5. *afficher les éléments du tableau T dont la longueur de séquence = NPLS sous la forme NL#ICD#ICF*

Algorithme du PP

Algorithme Sequnece_Matrice

Début

Saisie(L, C)

Remplissage(M, L , C)

Sequences (M, L, C, T, Nt)

NPLS ← Plus_Longue_Séq(T, Nt)

Ecrire("Le nombre d'éléments de la plus longue séquence=", NPLS)

Affichage(T, Nt, NPLS)

Fin

Tableau de Déclaration des Nouveaux Types	
Mat	**Tableau de 24 Lignes x 24 Colonnes**
Séquence = Enregistrement **NL : Entier** **ICD : Entier** **ICF : Entier** **Fin**	
T ab= Tableau de 100 Séquences	

T.D.O. GLOBAUX	
M	**Mat**
T	**Tab**
L, C, Nt, NPLS	**Entier**
Saisie	**Procédure**
Remplissage	**Procédure**
Séquences	**Procédure**
Plus_Longue_Séq	**Fonction**
Affichage	**Procédure**

Algorithme de la procédure Saisie :

Procédure Saisie (@ L, C : Entier)

```
Début
   Répéter
     Ecrire("Introduire nombre de lignes:")
     Lire( L )
   Jusqu'à (L dans [2..24])
   Répéter
     Ecrire("Introduire nombre de colonnes:")
     Lire( C )
   Jusqu'à (L dans [2..24])
Fin
```

Algorithme de la procédure Remplissage :

Procédure Remplissage(@ M:Mat ; L, C : Entier)

```
Début
   Pour i de 0 à (L-1) Faire
     Pour j de 0 à (C-1) Faire
       Répéter
         Ecrire("M[ " , i , "," , j , "]=")
         Lire(M[i,j])
       Jusqu'à (M[i,j]≠0)
     FinPour
   FinPour
Fin
```

T.D.O. GLOBAUX	
Objet	**Nature / Type**
i , j	**Entier**

Algorithme de la procédure Séquences

Procédure Séquences (M : Mat ; L, C : Entier ; @ T: Tab ; @ Nt : Entier)

```
Début
  Nt ← 0
  Pour i de 0 à (L-1) Faire
     Pour j de 0 à ( C-2) Faire
       somme ← M[i , j]
       k ← j+1
```

```
        Tantque (somme ≠0 ) Et ( k< C) Faire
          somme ← somme + M[i, k]
          k ← k+1
        Fin TantQue
        Si (somme = 0 ) Alors
          T[Nt].NL ← i
          T[Nt].ICD ← j
          T[Nt].ICF ← k
          Nt ← Nt + 1
        FinSi
      FinPour
    FinPour
Fin
```

T.D.O. GLOBAUX	
Objet	**Nature / Type**
i , j	**Entier**
k, somme	**Entier**

Algorithme de la fonction Plus_Longue_Seq

```
Fonction Plus_Longue_Seq(T : Tab ; Nt : Entier): Entier
Début
  L ← T[0].ICF – T[0].ICD
  Pour i de 1 à (Nt-1) Faire
    Si (T[i].ICF – T[i].ICD > L ) Alors
      L ← T[i].ICF – T[i].ICD
    FinSi
  FinPour
  Retourner L
Fin
```

T.D.O. GLOBAUX	
Objet	**Nature / Type**
i	**Entier**
L	**Entier**

Algorithme de la procédure Affichage :

```
Procédure Affichage( T : Tab ; Nt, L :Entier)
Début
  Pour i de 0 à (Nt-1) Faire
    Si (T[i].ICF – T[i].ICD = L) Alors
      Ecrire(T[i].NL , "#" , T[i].ICD , "#" , T[i].ICF)
    FinSi
  FinPour
Fin
```

T.D.O. GLOBAUX	
Objet	**Nature / Type**
i	**Entier**

```python
from numpy import *
M=array([[int()]*24]*24)
T=array([{}]*100)
def Saisie():
    global L
    global C
    valid=False
    while (valid==False):
        L=int(input(" donner 2<L<=24:"))
        valid = (2<L<=24)
    valid=False
    while (valid==False):
        C=int(input(" donner 2<C<=24:"))
        valid = (2<C<=24)
def Remplissage(M,L,C):
     for i in range(L):
         for j in range(C):
             valid=False
             while(valid==False):
                 M[i,j]=int(input("M[ "+str(i)+" , "+str(j)+" ] !=0 : "))
                 valid=M[i,j] !=0
def Sequences(M,L,C,T):
    global Nt
    Nt=0
    for i in range (L) :
        for j in range(C-1):
            somme=M[i,j]
            k=j+1
            while(k<C)&(somme!=0):
                somme = somme + M[i , k]
                k= k + 1
            if somme == 0 :
                T[Nt]={}
                T[Nt]["NL"] = i
                T[Nt]["ICD"] =j
                T[Nt]["ICF"] =k
                Nt = Nt+1
def Plus_Longue_Séq(T, Nt):
    L = T[0]["ICF"] - T[0]["ICD"]
    for i in range(1,Nt):
        if T[i]["ICF"] - T[i]["ICD"]> L :
            L = T[i]["ICF"] - T[i]["ICD"]
    return L
def Affichage(T,Nt, NPLS):
    for i in range(Nt):
        if T[i]["ICF"] - T[i]["ICD"] == L :
            print(T[i]["NL"] , "#" , T[i]["ICD"] , "#" , T[i]["ICF"])
# Programme Principal
Saisie()
Remplissage(M, L,C)
Sequences(M,L,C,T)
NPLS = Plus_Longue_Séq(T, Nt)
print("Le nombre d'éléments de la plus longue séquence=", NPLS)
Affichage(T, Nt, NPLS)

```