

SECTION : SCIENCES DE L'INFORMATIQUE

ÉPREUVE : ALGORITHMIQUE ET PROGRAMMATION

DURÉE : 1H 30

NOMBRE DE PAGES : 3

COEF : 1

Exercice 1 : (3 pts)

Dans un contexte informatique et pour chacune des propositions données ci-dessous, mettre dans chaque case, la lettre **V** si la proposition est correcte, ou la lettre **F** dans le cas contraire.

a. L'opération de décalage est utilisée dans :

Le tri rapide Le tri par insertion Le tri Shell

b. Le tri par insertion est un cas particulier du :

Tri par sélection Tri à bulle Tri Shell

c. Le pas du tri Shell noté P est déterminé en utilisant la suite :

$\begin{cases} P_0 = 0 \\ P_n = 3 + P_{n-1} \end{cases}$ $\begin{cases} P_0 = 1 \\ P_n = 2 * P_{n-1} \end{cases}$ $\begin{cases} P_1 = 1 \\ P_n = 3 * P_{n-1} + 1 \end{cases}$

d. La fonction **Verif** permet de vérifier si les N entiers d'un tableau **T** sont triés en ordre croissant :

Fonction Verif (T : Tab , N : entier) : Booléen
Début

```

Si N = 1 Alors
|   Retourner Vrai
Sinon
|   Si T[N-1] < T[N - 2] Alors
|   |   Retourner Faux
|   Sinon
|   |   Retourner Verif ( T , N - 1)
|   Fin Si
Fin Si

```

Fin

Fonction Verif (T : Tab , N : entier) : Booléen
Début

```

Si N = 1 Alors
|   Retourner Vrai
Sinon
|   Retourner (T[N-1] ≥ T[N - 2]) ET Verif ( T , N - 1)
Fin Si

```

Fin

Fonction Verif (T : Tab , N : entier) : Booléen
Début

```

Si N = 1 Alors
|   Retourner Vrai
Sinon
|   Retourner Verif ( T , N - 1) ET (T[N-1] ≥ T[N - 2])
Fin Si

```

Fin

Exercice 2 : (3 pts)

En mathématiques, et plus particulièrement en combinatoire, les **nombre de Catalan** forment une suite d'entiers naturels utilisée dans divers problèmes de dénombrement. Cette suite est définie comme suit :

$$C \begin{cases} C_0 = 1 \\ C_{n+1} = \frac{(4 * n + 2)}{(n + 2)} * C_n \quad \forall n + 1 > 0 \end{cases}$$

Questions :

- 1- Quel est l'ordre de récurrence de cette suite ? justifier votre réponse.
- 2- Ecrire un algorithme d'une fonction récursive qui retourne le n^{ème} terme de la suite.

EXERCICE 3 : (6 points)

Pour **A** et **B** deux entiers strictement positifs, on définit la relation suivante :

Si $(A! * B! \text{ MOD } (A+B) = A)$ ou $(A! * B! \text{ MOD } (A+B) = B)$ Alors **A+B** est un nombre premier

Exemples :

A	B	A+B	A!	B!	A! * B!	A! * B! mod (A+B)	A+B
2	3	5	2	6	12	$12 \text{ mod } 5 = 2 = A$	5 est premier
7	4	11	5040	24	120960	$120960 \text{ mod } 11 = 4 = B$	11 est premier

Soit un fichier texte nommé « **source.txt** » contenant dans chaque ligne un couple de deux valeurs séparées par un espace représentant respectivement les valeurs de deux entiers **A** et **B**.

A partir du fichier « **source.txt** », on se propose de générer un nouveau fichier d'enregistrements « **resultat.dat** » où chaque enregistrement contient les valeurs du couple **A** et **B**, vérifiant la relation définie précédemment.

NB. : Le candidat n'est pas appelé à remplir le fichier "**source.txt**".

Travail demandé :

- 1- Tracer le tableau de déclaration des nouveaux types où vous déclarez un type qui simule l'enregistrement utilisé dans le fichier "**resultat.dat**" et un type pour ce fichier.
- 2- Ecrire un algorithme d'un module permettant de générer le fichier "**resultat.dat**" à partir du fichier "**source.txt**" comme expliqué ci-dessus.

Exercice 4 : (8 points)

Le principe du tri à peigne est le même que le tri à bulles mais cette fois-ci on permute des éléments plus lointains puis on raccourcit progressivement par un facteur de réduction jusqu'à un pas égal à 1.

Initialement le pas est égal à la partie entière de la division de la taille du tableau par le facteur de réduction ensuite, on raccourcit progressivement par à la partie entière de la division de son ancienne valeur par le facteur de réduction (le pas est égal à 1 si la partie entière de la division est égale à zéro)

Le tri s'arrête lorsque le pas est égal à 1 et **aucune permutation**.

En général, on prend un facteur de réduction compris entre 1.25 et 1.33. **On prendra ici 1.3**

Exemple : Soit le tableau **T** suivant :

T	14	21	8	15	35	59	63	9	42	69
	0	1	2	3	4	5	6	7	8	9

Ici $N = 10$, \rightarrow Pas = ent $(\frac{10}{1.3}) = 7$

Donc on compare T[0] avec T[7], T[1] avec T[8] et T[2] avec T[9] d'où le tableau T devient

T	9	21	8	15	35	59	63	14	42	69
	0	1	2	3	4	5	6	7	8	9

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{7}{1.3} \right) = 5$$

Donc on compare T[0] avec T[5], T[1] avec T[6], T[2] avec T[7], T[3] avec T[8] et T[4] avec T[9] d'où le tableau T devient

T	9	21	8	15	35	59	63	14	42	69
	0	1	2	3	4	5	6	7	8	9

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{5}{1.3} \right) = 3$$

Donc on compare T[0] avec T[3], T[1] avec T[4], T[2] avec T[5], T[3] avec T[6], T[4] avec T[7], T[5] avec T[8] et T[6] avec T[9] d'où le tableau T devient

T	9	21	8	15	14	42	63	35	59	69
	0	1	2	3	4	5	6	7	8	9

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{3}{1.3} \right) = 2$$

Donc on compare T[0] avec T[2], T[1] avec T[3], T[2] avec T[4], T[3] avec T[5], T[4] avec T[6], T[5] avec T[7], T[6] avec T[8] et T[7] avec T[9] d'où le tableau T devient

T	8	15	9	21	14	35	59	42	63	69
	0	1	2	3	4	5	6	7	8	9

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{2}{1.3} \right) = 1$$

Donc on compare T[i] avec T[i+1] avec i de 0 à 8 d'où le tableau T devient

T	8	9	15	14	21	35	42	59	63	69
	0	1	2	3	4	5	6	7	8	9

On contenu car il y'a encore des permutations (Permut = VRAI)

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{1}{1.3} \right) = 0 \text{ donc Pas} = 1$$

Donc on compare T[i] avec T[i+1] avec i de 0 à 8 d'où le tableau T devient

T	8	9	14	15	21	35	42	59	63	69
	0	1	2	3	4	5	6	7	8	9

On contenu car il y'a encore des permutations (Permut = VRAI)

$$\text{Pas} = \text{ent} \left(\frac{\text{Pas}}{1.3} \right) = \text{ent} \left(\frac{1}{1.3} \right) = 0 \text{ donc Pas} = 1$$

Donc on compare T[i] avec T[i+1] avec i de 0 à 8 d'où le tableau T devient

T	8	9	14	15	21	35	42	59	63	69
	0	1	2	3	4	5	6	7	8	9

Le Pas ≤ 1 et aucune permutation donc le tableau est trié

Travail demandé :

- Soit **F** un fichier d'au maximum 100 entiers dont son nom physique est « **nombre.dat** ». Ecrire un algorithme d'une fonction nommée **taille(F)** permettant de retourner le nombre d'éléments dans ce fichier.
- En se basant sur le principe de tri peigne décrit précédemment et en utilisant la fonction **taille**, écrire l'algorithme d'un module nommé **TRI_peigne** qui permet de trier dans un ordre croissant les éléments du fichier **F**.

Solutions Python

Exercice 2 :

$$C_{n+1} = \frac{(4*n+2)}{(n+2)} * C_n \rightarrow C_n = \frac{(4*(n-1)+2)}{(n-1+2)} * C_{n-1} = C_n = \frac{(4*n-2)}{(n+1)} * C_{n-1}$$

```
def catalan(n):
    if n==0:
        return 1
    else:
        return int((4*n-2)/(n+1)*catalan(n-1))
```

Exercice 3 :

```
from pickle import load,dump
def remplir():
    F=open("Nombre.dat","wb")
    for i in range(10):
        k=int(input("K="))
        dump(k,F)
    F.close()
```

Exercice 4 :

```
from numpy import array
from pickle import load,dump
def taille(F):
    nb=0
    Fin=False
    while not Fin:
        try:
            k=load(F)
            nb=nb+1
        except:Fin=True
    return nb
def tri(T,N):
    P=N
    permut=True
    while permut or P>1:
        permut=False
        P=int(P/1.3)
        if P<1:
            P=1
        for i in range(N-P):
            if T[i]>T[i+P]:
                T[i],T[i+P]=T[i+P],T[i]
            permut=True
def TRI_peigne(F):
    n=taille(F)
    F=open("Nombre.dat","rb")
    T=array([int]*n)
    for i in range(n):
        T[i]=load(F)
    tri(T,n)
    F=open("Nombre.dat","wb")
    for i in range(n):
        dump(T[i],F)
    F.close()
```