Série d'exercices n°4

On exige des solutions modulaires aux problèmes.
(Tri + Recherche + Algorithmes : récurrents, arithmétiques, d'approximation)

Exercice n°1 :

L'algorithme suivant est celui d'une fonction permettant de calculer la somme d'une partie d'éléments d'un tableau **T** de **n** entiers, délimitée par les indices **p1** et **p2.**

```
Fonction Somme (T : tab ; p1, p2 : entier) : entier
Début
        S ◄—— 0
        Pour i de p1 à p2 faire
                S ◄—— S + T[i]
        Retourner S
Fin
```

En exploitant la fonction dont l'algorithme est ci-dessus, on se propose de :

• Saisir un tableau **T** de **N** entiers (**5 ≤ n ≤ 20**).

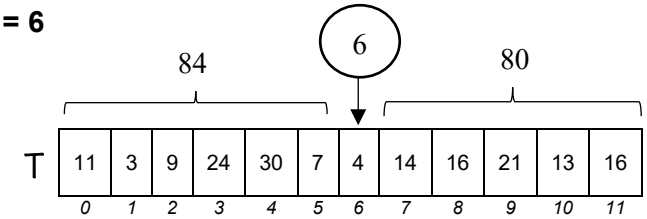
• Afficher l'indice (**ind**) de l'élément du tableau dont l'écart entre la somme (**S1**) des éléments qui le précèdent et celle des éléments qui le succèdent (**S2**) soit minimal.

• Afficher les sommes **S1** et **S2** correspondantes.

Exemple : Pour le tableau **T** suivant :

T	11	3	9	24	30	7	4	14	16	21	13	16
	0	1	2	3	4	5	6	7	8	9	10	11

Le programme affiche : **S1 = 84, S2 = 80 et ind = 6**



Questions

1) Faire l'algorithme du programme principal.

2) Faire les algorithmes des modules envisagés.

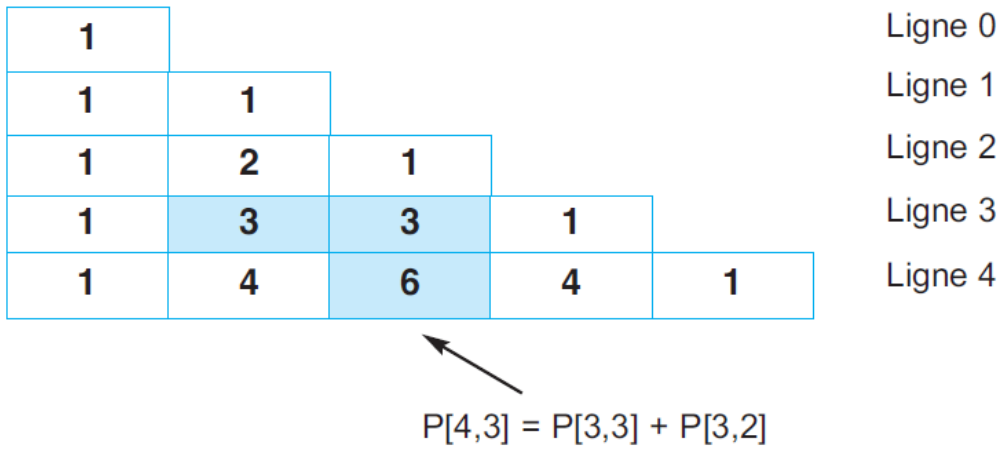
Exercice n°2 :

Ecrire un algorithme et son implémentation en **Python** d'une procédure **TermeN** qui permet de chercher et d'afficher les **n** premiers termes de la suite **U** définie par :

$$\begin{cases} U_0 = 5 \\ U_n = 2U_{n-1} + 1.5 \end{cases}$$

Exercice n°3 :

Ecrire un algorithme d'une procédure **PascalN** qui permet de remplir les **n** premières lignes du triangle de **Pascal** (**n** étant une donnée entière vérifiant **2 ≤ n ≤ 20**).



Exercice n°4 :

La **suite** de **Fibonacci** est une **suite récurrente** définie par :

$F_0 = 0$

$F_1 = 1$

$F_{n+2} = F_{n+1} + F_n$

Créer une **fonction** qui renvoie le **(n+1)ème** terme de la suite de **Fibonacci** pour un **entier n** donné, puis un algorithme qui affiche les **10 premiers** termes de la suite de **Fibonacci**.

Exemple :

F(6) = 0 1 1 2 3 5 8

Exercice n°5 :

Suite de Thue-Morse :

C'est une suite binaire définie par : $U_0 = 0$ et par la récurrence suivante : pour passer de U_n à U_{n+1} on remplace tous le **0** de U_n par **01** et tous les **1** par **10**.

Exemple :

$U_0 = 0$ **(ou $U_0 = 1$)**
$U_1 = 01$
$U_2 = 01\ 10$
$U_3 = 01\ 10\ 10\ 01$

Donner l'algorithme qui permet de saisir un chiffre binaire (qui peut être **0** ou 1) puis calcule et affiche les **N** premiers termes de la suite **Thue-Morse** (**avec N > 2**).

Exercice n°6 :

Compléter le tableau suivant :

n	Fib (n)	Fib (n+1) / Fib (n)	Valeur approchée de Fib (n+1) / Fib (n)
1	1	1	1
2	1	2	2
3	2	3/2	1,5
4	3	5/3	1,666
5	5	8/5	1,6
6	8	13/8	1,625
7	13	21/13	1,615
8	21		
9	34		
10	55		
11	89		
12	144		
13	233		

Donc la suite de **Fibonacci** et la valeur **1,618** sont fortement **liées**.

La valeur à laquelle converge **Vn** s'appelle **le nombre d'or** (phi) $\lambda = \frac{1+\sqrt{5}}{2}$

Soient deux suites **U** et **V** définies comme suit :

$$\begin{cases} U_1 = 1 \\ U_2 = 2 \\ U_n = U_{n-1} + U_{n-2} \text{ pour } n \geq 3 \text{ et} \\ V_n = U_n/U_{n-1} \text{ pour tout } n \geq 2. \end{cases}$$

La suite $\mathbf{V_n}$ tend vers une limite appelée **nombre d'or**.

On suppose que le $\mathbf{n^{ième}}$ terme de la suite **V** est égal à $\mathbf{V_n}$.

Vn est la valeur approchée du **nombre d'or** avec une précision **e** dès que $\mathbf{|\ V_n - V_{n-1}\ | < e.}$

Travail Demandé

Donner l'algorithme d'un programme nommé **Nombre_or**, qui cherche **Vn** à 10^{-4} prés et son rang.

Exercice n°7 :

On se propose de construire à partir d'un chiffre **E** impair donné une pyramide composée de **L** lignes. Chaque ligne est calculée en fonction de la ligne qui la précède en insérant à son début et à sa fin un chiffre **C** tel que :

C = (La somme des chiffres de la ligne précédente **+** nombre de chiffres de la ligne précédente) **MOD 10**.

La dernière ligne de la pyramide correspond au premier nombre divisible par **7**.

Exemple : Pour **E** = **1**

1
212
82128
6821286
068212860
20682128602
8206821286028
682068212860286 ← *{C'est le premier nombre divisible par 7}*

La ligne **6** est calculée en insérant le chiffre **C** au début et à la fin du nombre de la ligne ***5***.
Avec **C = ((0+6+8+2+1+2+8+6+0)+9) MOD 10 = 42 MOD 10 = 2**
NB : le chiffre **0** à gauche est pris en compte dans le calcul du nombre de chiffres de la ligne précédente

Pour déterminer si un nombre **N** est divisible par **7**., Il suffit de le décomposer en des tranches de trois chiffres en commençant par la droite et d'insérer alternativement des **+** et des **-** devant les tranches en commençant par l'opérateur **+**. On effectue l'opération ainsi écrite, si le résultat est divisible par **7** alors : **N** est divisible par **7**.

Pour : **N = 682068212860286** et en appliquant la règle de divisibilité par **7** ci-dessus, on obtient **+ 286 – 860 + 212 – 068 + 682 = 252** qui est divisible par **7** donc **N** est divisible par **7**.

Travail Demandé

Ecrire un programme qui permet de saisir un entier **E** impair (**1 ≤ E ≤ 9**), d'afficher les entiers correspondants à **E** selon le principe décrit précédemment à raison d'un entier par ligne.

NB : *Le candidat n'est pas appelé à afficher les entiers sous la forme d'une pyramide*

Exercice n°8 :

Soit le tableau suivant :

ALPHA	A	C	R	Y	T	G	I	J	O	N	P	F	E
	0	1	2	3	4	5	6	7	8	9	10	11	12

On se propose de chercher l'existence dans le tableau **ALPHA** d'une lettre **I** saisi par l'utilisateur. On demande d'appliquer les méthodes suivantes :

a) **La méthode de recherche séquentielle**
b) **La méthode de recherche dichotomique** : utiliser la méthode de tri à bulles pour trier le tableau **ALPHA.**

Exercice n°9 :

N est un nombre de **SIERPINSKI** s'il vérifie la propriété suivante : $N = K * 2^i + 1$ avec **i** un entier strictement positif.

Exemple :

Pour **K = 3** et pour tout **i** variant de **1 à 10**. Le programme affiche tous les nombres de **SIERPINSKI**, ci-dessous :

K	i	N	
3	1	7	Premier
3	2	13	Premier
3	3	25	
3	4	49	
3	5	91	Premier
3	6	193	Premier
3	7	385	
3	8	769	Premier
3	9	1537	
3	10	3073	

N.B :

Un nombre est dit **premier** s'il n'est divisible que par **1** et par **lui-même**. Par définition, **1** n'est pas premier.

Travail demandé :

Ecrire un programme qui permet de choisir aléatoirement un entier **K** de l'intervalle **[1,5]** de chercher et d'afficher tous les nombres de **SIERPINSKI** en variant **i** de **1** à **10**, tout en mentionnant le terme "**Premier**" devant les nombres de **SIERPINSKI** qui sont premiers comme indiqué dans l'exemple ci-dessus.

Exercice n°10 :

On se propose d'écrire un programme qui permet de saisir un entier **N** de trois chiffres non nuls, de déterminer et d'afficher tous les nombres qui peuvent être formes par les chiffres de **N**, ainsi que le plus **petit** et le plus **grand** de ces nombres.

Exemple :

Pour **N= 427** :

• Les nombres formes par les chiffres de N sont : **427, 472, 724, 742, 247, 274**

• Le plus petit nombre est **247**

• Le plus grand nombre est **742**

Exercice n°11 :

On se propose d'écrire un programme qui permet d'effectuer sur un tableau **T** de **N** éléments de type entier (**5<N<20**) les opérations suivantes :

1) Saisir deux entiers positifs **Ind_i** et **Ind_j** avec **0 < Ind_i < Ind_j ≤ N**
2) Déterminer et afficher la valeur minimale (**Min**) d'une partie du tableau **T** comprise entre les indices données **Ind_i** et **Ind_j** .
3) Déterminer le nombre de multiples de la valeur **Min** dans le tableau **T**.

Exercice n°12 :

Sachant que :

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \ldots$$ tel que **x** ∈ [0, 2π].

Ecrire un programme qui permet d'afficher **sin(x)** en utilisant la formule ci-dessus. Le calcul s'arrête quand la différence entre deux termes consécutifs devient ≤ à 10^{-4}. La dernière somme calculée est une valeur approchée de **sin(x)**.

Exercice n°13 :

Ecrire un algorithme d'un programme permettant de :

- Remplir aléatoirement un tableau **T** par **N (avec 4<n<=20)** nombres premiers de l'intervalle **[2,99].**
- Afficher toutes les séquences de valeurs consécutives croissantes, ainsi que leur nombre.

Exemple :

Pour le tableau **T** suivant :

T	5	29	13	17	53	23	2	5	19
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*

Le programme affichera :

5 29

13 17 53

23

2 5 19

*Le nombre de séquences est **4***

Exercice n°14 :

Ecrire un programme qui permet de saisir **n (10<n<40)** entiers à mettre dans un tableau **T** et deux entiers non nuls **p** et **s** (**p** et **s** doivent être deux éléments de **T**). On demande d'afficher tous les blocs d'éléments de **T** placés entre **p** et **s** dans l'ordre (**p** et **s** peuvent figurer dans cet ordre plusieurs fois dans **T**).

Exemple :

Si **p = 5** et **s = 3** et si on donne le tableau **T** suivant :

T	7	**5**	0	**3**	9	1	**5**	6	8	**3**
	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

*Alors le résultat de l'affichage sera : **0 6 8***

Exercice n°15 :

On désire faire un programme nommé « **Somme_pro** » qui permet de faire les tâches suivantes :

• Saisir un entier **N** (**avec 4 ≤ N ≤ 15**) et un réel **x > 0**

• Calculer et afficher la somme **S** suivante :

$$S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^N}{N!}$$

Travail à Faire :

- Faire l'algorithme du programme principal.
- Faire les algorithmes des modules envisagés.

Exercice n°16 :

Soit un tableau **T** de **N** entiers (**on suppose que N >= 2**), on veut déterminer et afficher le **K^{ième}** plus petit élément (**1 ≤ k ≤ N**) et l'indice de sa première apparition dans le tableau **T**.

Exemple :

Soit le tableau **T** suivant :

T	5	2	7	2	1	4	9	4	1	1
	0	1	2	3	4	5	6	7	8	9

Pour k = 3

*Le **3ème** élément minimal est **4** et l'indice de sa première apparition est **5**.*

Travail à Faire :

- Faire l'algorithme du programme principal.
- Faire les algorithmes des modules envisagés.

Exercice n°17 :

Ecrire un algorithme d'un programme intitule **Tri** permettant de trier un tableau **T** de **N** entiers distincts (**5 < N < 20**) selon le principe suivant :

Pour chaque élément du tableau **T** :

- Déterminer le nombre d'éléments qui lui sont inférieurs.
- En déduire sa position au sein d'un autre tableau résultat appelé **R**.

Exemple : Pour un tableau **T** de **10** éléments :

1) Quatre valeurs sont inférieures au premier élément du tableau **T**. Cet élément sera donc placé à la position **4** du tableau **R.**

T	15	10	18	20	23	5	8	25	4	19
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

15 > 10, 15 > 5, 15 > 8 et 15 > 4

R					15					
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

2) Trois valeurs sont inférieures au deuxième élément du tableau **T**. Cet élément sera donc placé à la position **3** du tableau **R.**

T	15	10	18	20	23	5	8	25	4	19
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

10 > 5, 10 > 8 et 10 > 4

R				10	15					
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

3) Cinq valeurs sont inférieures au Troisième élément du tableau **T**. Cet élément sera donc placé à la position **5** du tableau **R.**

18 > 15, 18 > 10 , 18 > 5, 18 > 8 et 18 > 4

R				10	15	18				
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

……

A la fin **R** est :

R	4	5	8	10	15	18	19	20	23	25
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**