

Section : ..... N° d'inscription : ..... Série : .....

Signature des  
surveillants

Nom et Prénom : .....

Date et lieu de naissance : .....



Algorithmique et Programmation - Section : Sciences de l'informatique – Session de contrôle 2025

Le sujet comporte 5 pages numérotées de 1/5 à 5/5.

Les réponses aux questions de l'exercice 1 doivent être rédigées sur les pages 1/5 et 2/5 qui doivent être remises à la fin de l'épreuve.

### Exercice 1 (3 points)

...../3,00

Soit l'algorithme suivant de la fonction **Inconnue** :

```
Fonction Inconnue (N, B : Entier) : Chaîne de caractères
DEBUT
  Si N=0 Alors
    Retourner "0"
  Sinon
    CH ← ""
    Répéter
      K ← N Mod B + 48 + ((N Mod B) Div 10) * 7
      CH ← Chr(K) + CH
      N ← N Div B
    Jusqu'à N = 0
    Retourner CH
FinSi
FIN
```

**NB** : **Chr(65)** retourne le caractère "A", **Chr(97)** retourne le caractère "a" et **Chr(48)** retourne le caractère "0".

#### Travail demandé

1) Valider chacune des propositions suivantes en mettant dans la case correspondante la lettre « V » si elle est correcte ou la lettre « F » dans le cas contraire.

a) Le résultat retourné par l'appel **Inconnue(5, 2)** est :

100

31

101

b) Le résultat retourné par l'appel **Inconnue(31, 16)** est :

1F

31

F1



N° d'inscription

--	--	--	--	--	--

**Exercice 2 (3,5 points)**

Soit **K** un entier de **d** chiffres (avec  $d \geq 2$ ) ayant la forme  $C_1C_2 \dots C_d$ . A partir de ce nombre on définit une suite **U** telle que les **d** premiers termes sont les **d** chiffres du nombre **K**. Ensuite chaque terme suivant est calculé en additionnant les **d** termes précédents.

$$U \begin{cases} U_1 = C_1 \\ U_2 = C_2 \\ \dots \dots \\ U_d = C_d \\ U_n = U_{n-1} + U_{n-2} + \dots + U_{n-d} \text{ pour } n > d \end{cases}$$

Le nombre **K** est appelé **nombre de Keith**, si à un moment donné, un terme de la suite **U** est égal au nombre **K**.

**Exemples :**

- Pour **K = 197**, le nombre de chiffres qui le forment est égal à **3**, donc **d = 3** et la suite **U** est définie par :

$$U \begin{cases} U_1 = 1 \longleftarrow \text{Le 1}^{\text{er}} \text{ chiffre de K} \\ U_2 = 9 \longleftarrow \text{Le 2}^{\text{ème}} \text{ chiffre de K} \\ U_3 = 7 \longleftarrow \text{Le 3}^{\text{ème}} \text{ chiffre de K} \\ U_n = U_{n-1} + U_{n-2} + U_{n-3} \text{ pour } n > 3 \end{cases}$$

Les termes de la suite sont :  $U_1=1, U_2=9, U_3=7, U_4=17, U_5=33, U_6=57, U_7=107, U_8=197, \dots$   
 $U_8 = 197 = K$ , **K** est un terme de la suite donc **K est un nombre de Keith**.

- Pour **k = 24**, le nombre de chiffres qui le forment est égal à **2**, donc **d = 2** et la suite **U** est définie par :

$$U \begin{cases} U_1 = 2 \longleftarrow \text{Le 1}^{\text{er}} \text{ chiffre de K} \\ U_2 = 4 \longleftarrow \text{Le 2}^{\text{ème}} \text{ chiffre de K} \\ U_n = U_{n-1} + U_{n-2} \text{ pour } n > 2 \end{cases}$$

Les termes de la suite sont :  $U_1=2, U_2=4, U_3=6, U_4=10, U_5=16, U_6=26, \dots$ . Comme  $U_6=26 > K$  donc **K n'est pas un terme de la suite et par conséquent K n'est pas un nombre de Keith**.

**Travail demandé :**

1) Ecrire un algorithme d'une fonction **Verif(K)** qui permet de retourner :

- La valeur de **n** telle que  $U_n=K$ , si **K** est un **nombre de Keith**
- La valeur **-1** dans le cas contraire.

**Exemples :** **Verif(197)** retourne **8** et **Verif(24)** retourne **-1**

2) En utilisant la fonction **Verif**, écrire un algorithme d'un programme qui permet de remplir un tableau d'enregistrements **NK** par les nombres de **Keith** qui sont inférieurs à **100000**, sachant que chaque enregistrement du tableau **NK** contient les champs suivants :

- **Nbre** : le nombre de **Keith**.
- **Num** : le numéro du terme de la suite **U** qui est égal au nombre de **Keith** contenu dans le champ **Nbre**.

### Exercice 3 (3,5 points)

Une matrice carrée **M1** de taille  $d \times d$  est dite **Matrice multiple d'un premier** si le plus grand commun diviseur (PGCD) de ses éléments est un nombre premier.

Pour calculer le PGCD des éléments de la matrice **M1**, on suit les étapes suivantes :

- **Etape 1** : Stocker tous les éléments de la matrice **M1** dans la première ligne d'une matrice **M2** de taille  $d^2 \times d^2$ .
- **Etape 2** : Remplir les autres lignes de la matrice **M2** comme suit :  
$$M2[L,C] = \text{PGCD}(M2[L-1,C], M2[L-1,C+1])$$

Dans ce cas, le PGCD des éléments de **M1** est le contenu de la case  $M2[d^2 - 1,0]$

**Exemple** : Pour  $d = 2$  et la matrice **M1** suivante :

	0	1
0	20	10
1	4	8

On génère la matrice **M2** suivante :

	0	1	2	3
0	20	10	4	8
1	10	2	4	
2	2	2		
3	2			

En effet : - On remplit la ligne d'indice 0 par les éléments de M1.  
- On remplit les autres lignes comme suit :

**La ligne d'indice 1 :**

$$M2[1,0] = \text{PGCD}(20,10) = 10$$

$$M2[1,1] = \text{PGCD}(10,4) = 2$$

$$M2[1,2] = \text{PGCD}(4,8) = 4$$

**La ligne d'indice 2 :**

$$M2[2,0] = \text{PGCD}(10,2) = 2$$

$$M2[2,1] = \text{PGCD}(2,4) = 2$$

**La ligne d'indice 3 :**

$$M2[3,0] = \text{PGCD}(2,2) = 2$$

Le PGCD des éléments de **M1** est égal à  $M2[3,0]$  qui est égal à 2. Comme 2 est un nombre premier, donc **M1** est dite **Matrice multiple d'un premier**.

#### Travail demandé

- 1) Ecrire un algorithme d'une fonction **PGCD(a, b)** qui permet de calculer le PGCD de deux entiers **a** et **b**.
- 2) En appliquant les étapes définies précédemment et en utilisant la fonction **PGCD** de la question 1), écrire un algorithme d'une fonction **Verif(M1, d)** qui permet de vérifier si la matrice **M1** de taille  $d \times d$  est dite **Matrice multiple d'un premier**.

**NB :**

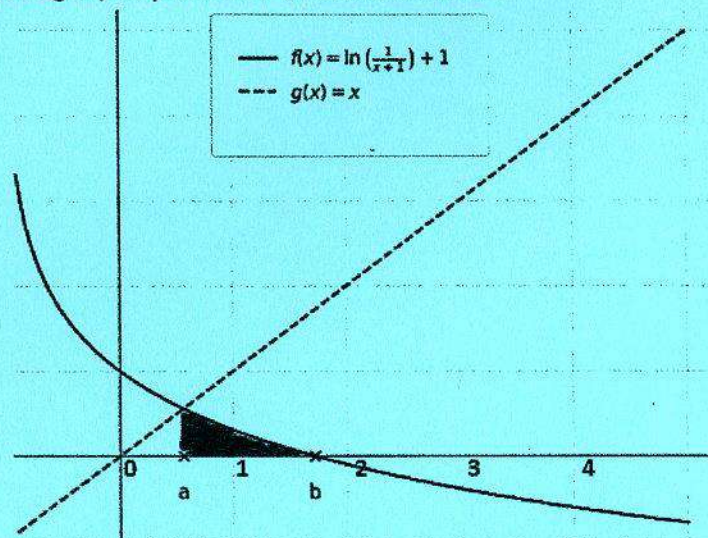
- **M1** est de type **Mat1**.
- **M2** est de type **Mat2**.
- Le candidat n'est pas appelé à dresser le tableau de déclaration pour définir les types **Mat1** et **Mat2**.

## Exercice 4 (10 points)

Soient les deux fonctions  $f$  et  $g$  suivantes :

- $f(x) = \text{Ln}\left(\frac{1}{x+1}\right) + 1$ , définie et continue sur l'intervalle  $]-1, +\infty[$  ( $\text{Ln}$  désigne le logarithme népérien).
- $g(x) = x$ , définie et continue sur  $\mathbb{R}$ .

Ci-après les représentations graphiques des deux fonctions :



Avec  $a$  et  $b$  sont :

- $a$  : Le point fixe de la fonction  $f$ ,  $f(a) = a$ .
- $b$  : Le point zéro de la fonction  $f$ ,  $f(b) = 0$  dans l'intervalle  $[1, 2]$ .

Soit  $A = \int_a^b f(x) \cdot dx$  l'aire de la partie grisée. Afin d'étudier la convergence des approximations de cette aire par les méthodes "**rectangles à gauche**" et "**rectangles à droite**" vers celle des "**rectangles du point milieu**", on se propose d'écrire un programme permettant de réaliser les tâches suivantes :

- La saisie d'un réel **epsilon** positif et inférieur ou égal à  $10^{-4}$ .
- La recherche du point fixe de  $f$  (le point  $a$ ) à **epsilon** près.
- La recherche du point zéro de  $f$  (le point  $b$ ) à **epsilon** près.
- La génération d'un fichier d'enregistrements "**Calcul.dat**" sur la racine du disque **C** où chaque enregistrement est formé par les champs suivants :
  - **n** : le nombre des subdivisions dans l'intervalle  $[a, b]$ .
  - **rd** : l'aire  $A$  calculée par la méthode des rectangles à droite avec  $n$  subdivisions.
  - **rg** : l'aire  $A$  calculée par la méthode des rectangles à gauche avec  $n$  subdivisions.
  - **rm** : l'aire  $A$  calculée par la méthode des rectangles du point milieu avec  $n$  subdivisions.

**NB :**

- Le traitement s'arrête lorsque la moyenne des aires calculées par les deux méthodes rectangles à droite et rectangles à gauche converge vers l'aire calculée par la méthode des rectangles du point milieu à epsilon près, c'est à dire :
$$\left| \frac{rd+rg}{2} - rm \right| < \text{epsilon}$$
- **n** est initialisé à **1** et s'incrémente jusqu'à la condition d'arrêt.
- Le candidat peut utiliser la fonction  $\text{Ln}(x)$ , sans la développer, pour calculer le logarithme népérien de  $x$ .

**Travail demandé :**

- 1) Ecrire un algorithme du programme principal en le décomposant en modules permettant de réaliser les tâches décrites précédemment.
- 2) Ecrire un algorithme pour chaque module envisagé.