| HORIZON de Sousse | |
|---|---|
| **Révision Examen (1/3)** | |
| ***Classe* : 1LGL**<br>***Matière* : Atelier Prog. 1**<br>***Enseignant* : Sakka Rouis Taoufik**<br>***Documents* Autorisés : Non** | **Session Dec. 2025**<br>**A.U. : 2025-2026**<br>**Durée : 1H30**<br>**Nombre Total de Pages : 1** |

## Exercice 1 : (20 points)

On se propose de :

✓ Remplir un tableau T par n entiers.

✓ Chercher et afficher tous les entiers dont le **miroir est un nombre appartenant à la suite de Fibonacci**.

- Le miroir d'un entier X est obtenu en inversant ses chiffres.
- La suite de Fibonacci est définie par :
  F(0)=0, F(1)=1 et F(n)=F(n-1)+F(n-2).

Exemples :

- Le miroir de 731 est 137.
- Les nombres de Fibonacci inférieurs à 500 sont : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377.

**Question :** Implémenter les méthodes suivantes :

- void saisie(int *n) permettant de lire la taille (au max 20) du tableau T.
- void remplir(int T[], int n) qui permet de remplir T par n entiers strictement supérieurs à 10.
- void triInsertion (int T[], int n) qui trie le tableau T dans l'ordre croissant en utilisant l'algorithme du tri par insertion.
- void afficher(int T[], int n) permettant d'afficher les n éléments du tableau
- int miroir(int X) qui retourne le miroir de X.
- int estFibo (int X) qui retourne 1 si X appartient à la suite de Fibonacci, sinon 0.
- void afficherRes(int T[], int n) qui affiche les entiers de T dont le miroir appartient à la suite de Fibonacci.
- La fonction main() pour tester.

| HORIZON de Sousse | |
|---|---|
| **Révision Examen (2/3)** | |
| ***Classe* : 1LGL**<br>***Matière* : Atelier Prog. 1**<br>***Enseignant* : Sakka Rouis Taoufik**<br>***Documents* Autorisés : Non** | **Session Dec. 2025**<br>**A.U. : 2025-2026**<br>**Durée : 1H30**<br>**Nombre Total de Pages : 1** |

## Exercice 1 : (20 points)

On se propose de :

✓ Remplir un tableau T par n entiers.

✓ Vérifier si le **miroir de la somme des chiffres** d'un entier est un **nombre triangulaire**.

- Le miroir d'un entier X est constitué par ses chiffres écrits à l'envers.
- La somme des chiffres d'un entier est la somme de ses digits.
- Un entier k est triangulaire s'il existe un entier t tel que :
  $k = t \times (t + 1) / 2$.

Exemples :

- Somme des chiffres de 842 : 8 + 4 + 2 = 14
- Miroir de 14 = 41
- 41 n'est pas triangulaire (car $41 \neq t(t+1)/2$)

Question : Implémenter les méthodes suivantes :

- void saisie(int *n) permettant de lire la taille (au max 20) du tableau T.
- void remplir(int T[], int n) qui remplit le tableau T par des entiers supérieurs à 10.
- void triSeq (int T[], int n) qui trie le tableau T dans l'ordre croissant en utilisant l'algorithme du tri par sélection.
- void afficher(int T[], int n) qui affiche les éléments du tableau.
- int somme_chiffres(int X) qui retourne la somme des chiffres de X.
- int miroir(int X) qui retourne le miroir de X.
- int triangulaire(int X) qui retourne 1 si X est triangulaire, sinon 0.
- void afficherRes(int T[], int n)qui affiche tous les entiers du tableau T dont le miroir de la somme des chiffres est un nombre triangulaire.
- La fonction main() pour tester.

| HORIZON de Sousse | |
|---|---|
| **Révision Examen (3/3)** | |
| ***Classe* : 1LGL**<br>***Matière* : Atelier Prog. 1**<br>***Enseignant* : Sakka Rouis Taoufik**<br>***Documents* Autorisés : Non** | **Session Dec. 2025**<br>**A.U. : 2025-2026**<br>**Durée : 1H30**<br>**Nombre Total de Pages : 1** |

Dans un système de transmission numérique, une trame de données est souvent représentée sous la forme d'un tableau d'entiers. Lors de certaines transmissions, cette trame peut contenir de longues suites de valeurs identiques, ce qui augmente inutilement la taille des données à envoyer. Pour optimiser la transmission, on souhaite compresser la trame avant son envoi en appliquant un algorithme de codage par comptage de répétitions (Run-Length Encoding).

Le principe de cet algorithme consiste à remplacer chaque suite consécutive d'une même valeur **x** répétée **n** fois par deux valeurs successives : la valeur **x** et le nombre de répétitions **n**. Ainsi, la trame compressée contient une alternance de valeurs et de compteurs.

**Travail demandé :**

**1)** Écrire une fonction en langage C qui, à partir d'une trame **T1** de taille **N1**, construit la trame compressée **T2** et de taille **N2**.

**Exemple :** Pour la trame T1 : [1 1 1 5 5 5 5 1 1 1 1 1 1 1 1] de longueur N1=15, le compresseur génère la trame T2 : [ 1 3 5 4 1 8] de longueur N2=6.

**2)** Proposer une seconde fonction qui permet d'afficher la trame initiale à partir de la trame compressée **T2**. La fonction doit avoir **exactement** le prototype suivant :

**void decompresser (int T2[], unsigned N2) ;**

*Rappel* : Pour les tableaux, le passage se fait toujours par adresse et T2 + i désigne l'adresse de la case numéro i.

**3)** Pour renforcer la sécurité lors de la transmission et afin d'éviter une attaque par analyse statistique de la trame compressée, on vous demande d'implémenter une méthode de **chiffrement** (cryptage) et une seconde de **déchiffrement**.

**a)** Écrire un module en langage C qui permet le chiffrement de la trame compressée T2 en appliquant le principe suivant :

T2[i] = ( (T2[i]×(i+1) ) div K) ×10 + ( (T2[i]×(i+1) ) mod K) où **K** (appelée *clé de chiffrement*) est un entier choisi par l'émetteur de la trame.

**b)** Écrire un module en langage C qui permet le **déchiffrement** de la trame cryptée T2, sachant que pour garantir le décryptage, on suppose que **K** est un entier **premier** inférieur à 10.

**N. B.** : Les prototypes de ces deux modules sont :

**void crypter (int T2[], unsigned N2, unsigned K) ;**
**void decrypter (int T2[], unsigned N2, unsigned K) ;**

| **Barème :** | **Question** | **1** | **2** | **3. a** | **3. b** |
|---|---|---|---|---|---|
| | **Note** | **6** | **5** | **4** | **5** |