Proposé par : ***Hajer Msolly & Fridhi Zied***

Lycée Ahmed NourEddine Sousse	***4ème Sciences Informatiques 1+2***	***Durée : 3 H***
	Devoir De Synthèse n°2	
14 Mars 2023	***Algorithmique & Programmation***	

Exercice n°1 : (2.75 points)

Soit la suite U suivante définie par :

$$\begin{cases} \mathbf{U_0 = 0} \\ \mathbf{U_1 = 3} \\ \mathbf{U_n = -U_{n-1} + 2 * U_{n-2}} \text{ , } \forall \boldsymbol{n} \geq \mathbf{0} \end{cases}$$

Questions :

1) Quel est l'ordre de récurrence de ce traitement, justifier ?
2) Ecrire un algorithme d'un module permettant de calculer le $n^{ème}$ terme de la suite U.

Exercice n°2 : (2.75 points)

Soit La formule de ***James Gregory*** suivante :

$$\mathbf{arctan\,(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \frac{x^{11}}{11} + \cdots}$$

1) Sachant que **si x = 1 , arctan(x) =** $\frac{\pi}{4}$, Déduire la formule permettant de calculer une valeur approchée de π à partir de **arctan(x)**.
2) Ecrire un algorithme d'un module permettant de calculer la valeur approchée de π à Epsilon= 10^{-5}.

Exercice n°3 : (5.5 points)

Mathématiquement, Un nombre de **Niven** est un nombre **divisible** par la **somme** de ses **chiffres**.

On se propose de créer un fichier "**niven.dat**" d'enregistrements représenté chacun par les deux champs suivants :

NV : un entier représentant un nombre de Niven de l'intervalle [100,1000]

NV_Hex : l'équivalent hexadécimal de NV qui contient au **moins un caractère** alphabétique.

Exemples : *Quelques nombres de Niven et leurs équivalents hexadécimaux :*

(102, 144, 264, 630, 915) ➔ (66, 90, 108, 276, 393)

Ces nombres de Niven ne seront pas sauvegarder dans le fichier car ils ne contiennent aucun caractère alphabétique.

(171, 195, 500, 700, 972) ➔ (**AB**, **C**3, 1**F**4, 2**BC**, 3**CC**)

Ces nombres de Niven seront sauvegarder dans le fichier car ils contiennent au moins un caractère alphabétique.

Travail demandé :

Ecrire un algorithme d'un module permettant de remplir le fichier ''niven.dat'' par tous les enregistrements dont le champs **NV** appartient à l'intervalle [100,1000] et que le champs **NV_Hex** contient au moins un caractère alphabétique.

NB : l'élève **doit** écrire tous les algorithmes des modules utilisés.

.dat niven

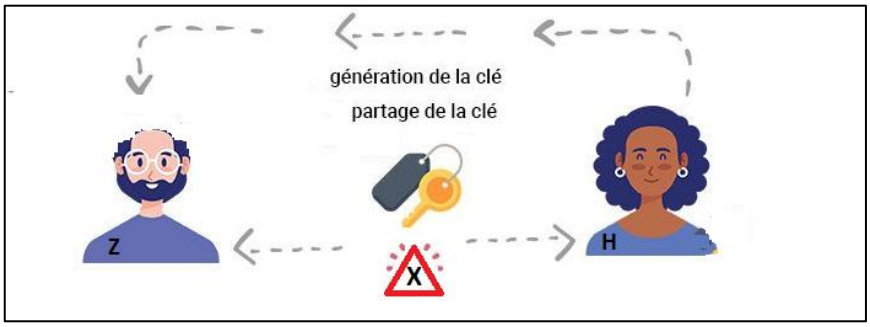
NV	NV_Hex
874	36A
910	38E
935	3A7
936	3A8
954	3BA
960	3C0
966	3C6
972	3CC
990	3DE

Exercice n°4 : (9 points)

La méthode de cryptage "**OU exclusif" ("XOR")** est une méthode de cryptage à **clé** qui utilise l'opération logique "OU exclusif" pour **combiner** un **message** avec une **clé secrète**.

Le chiffrement XOR fonctionne en effectuant l'opération **XOR** sur **chaque bit du message avec le bit correspondant de la clé secrète**. Le **résultat est un nouveau nombre binaire**, qui représente le message crypté.

Le chiffrement XOR est considéré comme un **chiffrement symétrique**, car la **même clé secrète** est utilisée pour **chiffrer et déchiffrer** le message. Cela signifie que la clé doit être **partagée** entre l'expéditeur et le destinataire en toute sécurité avant que le message ne soit chiffré.



Voici la table de vérité de l'opérateur logique XOR ("OU exclusif") :

- ✓ *Le résultat de (A **XOR** B) est VRAI si un et un seul des opérandes A et B est VRAI.*
- ✓ *Le résultat de (A **XOR** B) est FAUX si les deux opérandes A et B ont les mêmes valeurs logiques.*

A	B	(A XOR B)
FAUX	**FAUX**	**FAUX**
FAUX	VRAI	VRAI
VRAI	FAUX	VRAI
VRAI	**VRAI**	**FAUX**

Mécanisme de la méthode XOR pour crypter un message donné :

Etape 1 : Choisir une clé composée de caractères alphabétiques majuscules **aléatoires** tel que :

Long (Clé) = Long (Message) avec Message = une ligne du fichier texte.

Choisir **aléatoirement** un entier **X** tel que : **65≤ X ≤90** avec ORD(**''A''**) = 65 et ORD**(''Z'')** = 90

Etape 2 : Chaque **caractère** du message initial à coder est représenté par son **code ASCII**. Ce code est lui-même converti en son équivalent **binaire (une chaine binaire de 8 bits)**.

⇨ **Aussi, la clé doit subir le même traitement que le message initial.**

Etape 3 : Le message et la clé étant **converti en binaire**, on effectue **un XOR, bit par bit**, sachant que : **VRAI = 1** et **FAUX = 0**.

Etape 4 : Le **résultat en binaire** doit être **reconverti en décimale**.

Etape 5 : Ajouter à chaque nombre obtenu la valeur **X (choisi à l'étape 1)**

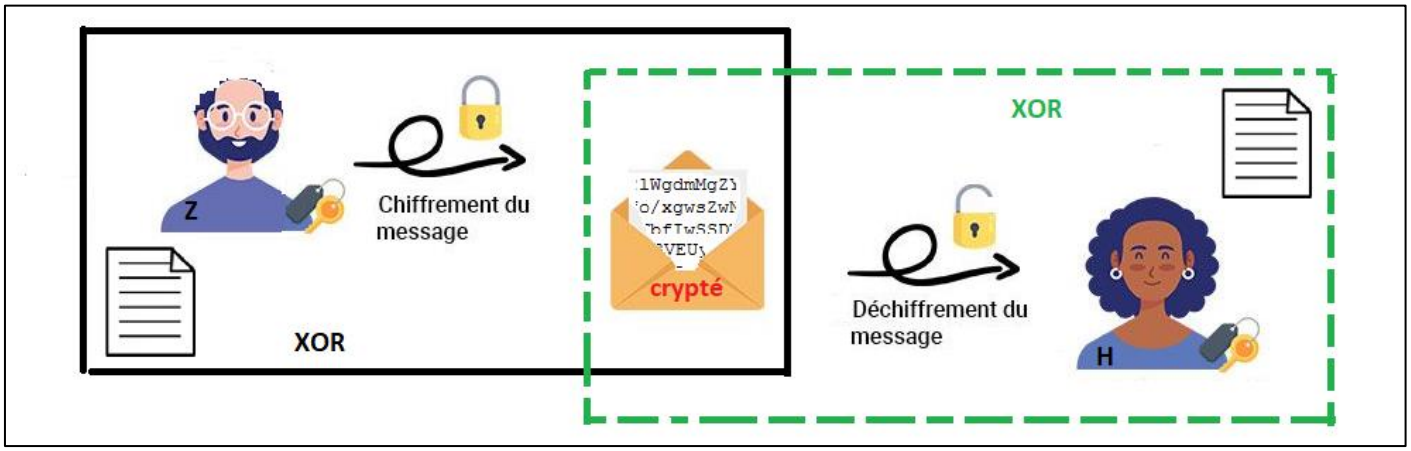
Etape 6 : Les nombres décimaux obtenus correspondent aux **codes ASCII** des caractères du **message crypté**.

On se propose d'écrire un programme permettant :

- Crypter le contenu d'un fichier texte **"F_init.txt"** en appliquant le principe décrit précédemment et le sauvegarder dans un deuxième fichier **"F_res.txt"**. Sachant que les lignes du fichier initial sont de **même longueur** et égal à **long(clé)**.
- Afficher le contenu du fichier crypté.

NB : l'élève n'est pas appelé à remplir le fichier **"F_init.txt".**

Illustration du principe de la méthode avec un exemple :



On suppose que la première ligne du fichier texte contient le mot : **"MESSAGE"** et **"AXGIFSD"** représente la clé de cryptage saisie aléatoirement et la valeur **X = 65.**

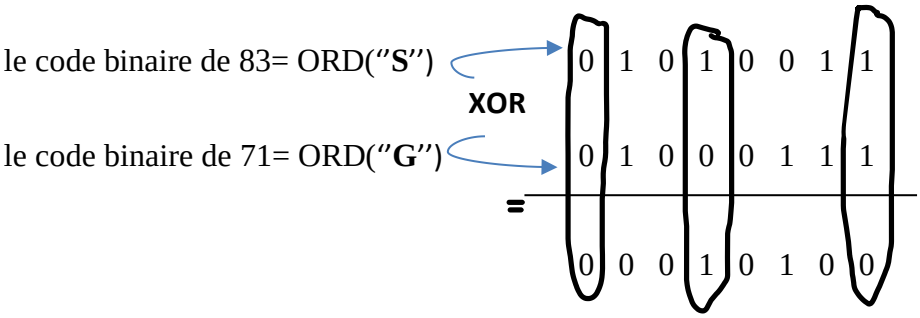
	"M"	**"E"**	**"S"**	**"S"**	**"A"**	**"G"**	**"E"**
Codes ASCII	77	69	83	83	65	71	69
Binaire (8bits)	01001101	01000101	01010011	01010011	01000001	01000111	01000101

Clé de cryptage	**"A"**	**"X"**	**"G"**	**"I"**	**"F"**	**"S"**	**"D"**
Codes ASCII	65	88	71	73	70	83	68
Clé en binaire	01000001	01011000	01000111	01001001	01000110	01010011	01000100

Msg **XOR** Clé	00001100	00011101	00010100	00011010	00000111	00010100	00000001

Code ASCII	12 + 65 = 77	29 + 65 = 94	20 + 65 = 85	26 + 65 = 91	7 + 65 = 72	20 + 65 = 85	1 + 65 = 66
Message crypté	**"M"**	**"^"**	**"U"**	**"["**	**"H"**	**"U"**	**"B"**

⇨ Le caractère ‘‘**S**’’ est remplacé par ‘‘**U**’’. En effet,



Travail demandé :

1) Ecrire un algorithme modulaire du programme principal

2) Ecrire les algorithmes des modules envisagés.