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

Lycée Ahmed NourEddine Sousse	***4ème Sciences Informatiques 1+2***	***Durée : x H***
	Révision BAC 2023 n°01	
Avril 2023	***Algorithmique & Programmation***	

Exercice n°1 : (BAC 2022 SC)

Soient le tableau de définition des nouveaux types (TDNT) et le tableau de déclaration des objets (TDO) :

TDNT :

Type
Renseignements = Enregistrement Nom : Chaine Num : Octet Etat_Civil : Caractère Fin Renseignements Eleve = fichier de Renseignements Fent = Fichier d'entiers Tsec = Tableau de 50 Renseignements

TDO :

O	T/N
F	Texte
E	Renseignement
P	Booléen
B	Octet
M	Réel
Felv	Eleve
Fe	Fent
T	Tsec

Dans le tableau suivant , valider chacune des instructions en mettant V si valide et F sinon. Justifier la réponse si l'instruction est invalide.

Instruction	Valide / Invalide	Justification
B ← Fin_Fichier(F)		
Ecrire (F , E.Etat_Civil)		
P ← E.Num > 10		
Ecrire (Felv , E.Nom)		
Ecrire (Fe, M)		
T[2] ← E.Num		

Exercice n°2 : (Niven)

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 :*

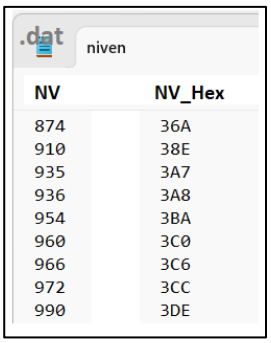
(102, 144, 264, 630, 915) ➔ (66, 90, 108, 276, 393)

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.

Partie théorique (Algorithme) :



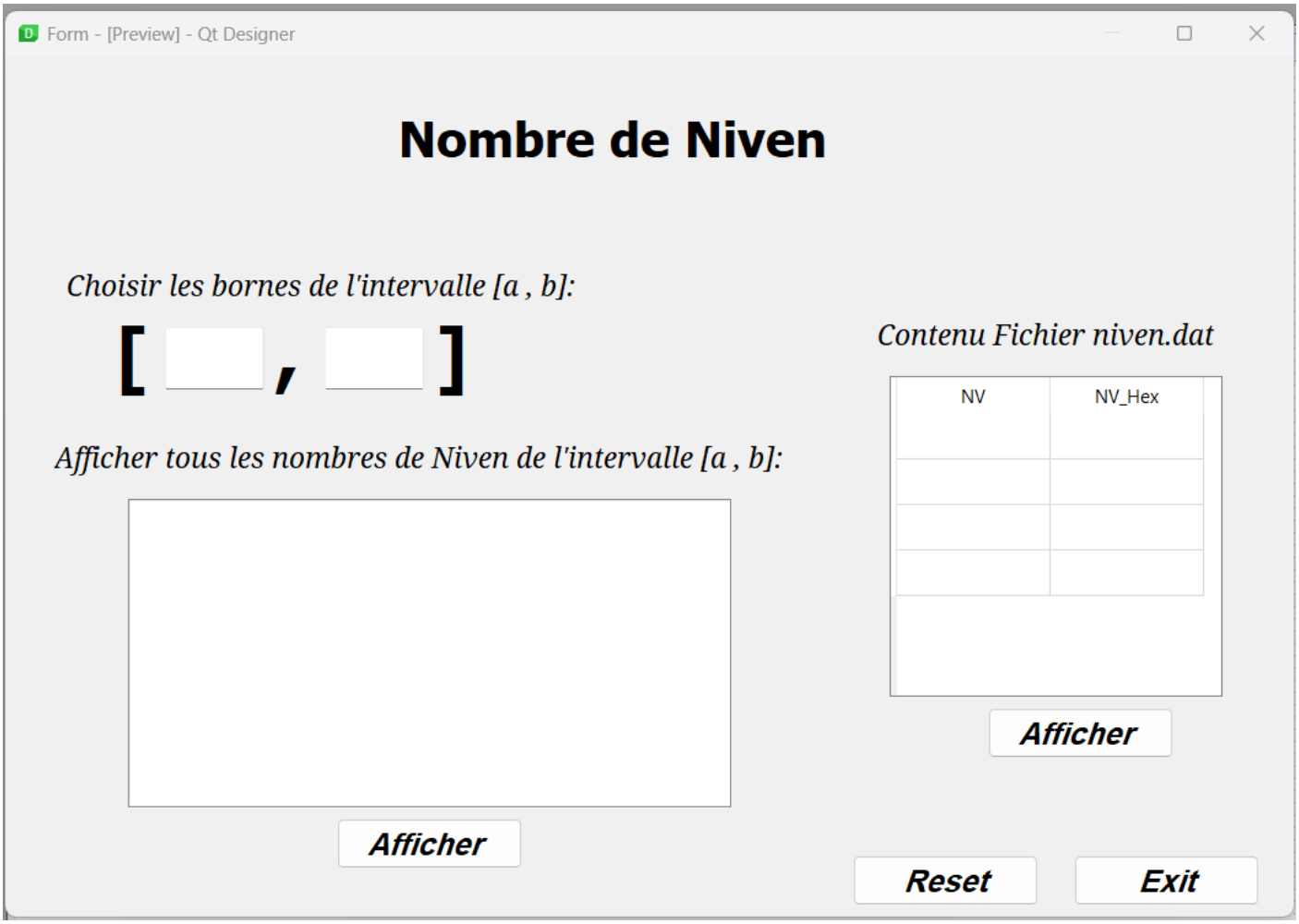
Ecrire un algorithme d'un module permettant de remplir le fichier ''niven.dat'' par tous les enregistrements dont le champs **NV** appartient à l'intervalle [100,1000] et que le champs **NV_Hex** contient au moins un caractère alphabétique.

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

Partie Pratique (Python + Qt Designer) :

On vous demande de créer une interface graphique permettant de :

- Saisir les bornes de l'intervalle [a,b] tel que 100 < a< b < 1000.
- Afficher tous les nombres de Niven de l'intervalle [a,b] dans une List widget, en appuyant sur le bouton **Afficher.**
- Afficher le contenu du fichier typé ''niven.dat'' sous forme d'une Table Widget comme décrit dans l'exercice.



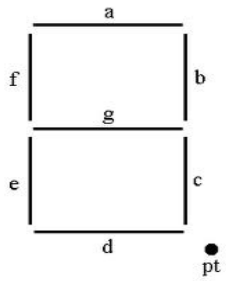
Exercice n°3 : (Calculette)

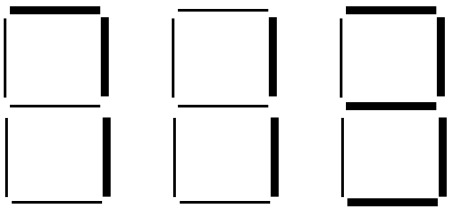
Une **calculatrice**, ou **calculette**, est une machine conçue pour simplifier, et fiabiliser, des opérations de calculs.
Dans les années 1970, elles se miniaturisent pour devenir portables grâce à l'affichage à **sept segments**

Les afficheurs 7 segments sont un type d'afficheur très présent sur les calculatrices et les montres à affichage numérique : les chiffres s'écrivent en allumant ou en éteignant des segments. Quand les 7 segments sont allumés, on obtient le chiffre **8**.

*Un segment allumé est représenté par le caractère **1***
*Un segment éteint est représenté par le caractère **0***

Voici quelques exemples représentés avec l'affichage à 7 segments :





1110000 0110000 1111001

- Pour que la calculatrice affiche la valeur 7 il faut que les segments **a,b,c** doivent être allumer et les segments **d,e,f,g** éteints (1110000).
- Pour que la calculatrice affiche la valeur 3 il faut que les segments **a,b,c,d,g** doivent être allumer et les segments **e , f** éteint.(1111001)

Etant donnée un tableau **T** de dix codes binaires représentant chacun un chiffre selon l'état des segments allumé ou éteint en commencent par le segment **a** puis **b** … jusqu'à **g (ordre alphabétique)** :

1111110	0110000	1101101	1111001	0110011	1011011	1011111	1110000	1111111	1110111
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

En Utilisant le tableau T, on se propose compléter le remplissage d'une matrice **M** en calculant les opérations d'additions des codes mémorisés dans les deux premières colonnes de **M.**
- La matrice **M** est formée de 3 colonnes dont la première colonne contient l'opérande **A** , la deuxième colonne contient l'opérande **B** et la troisième colonne contiendra le résultat à calculer de l'opération (**A+B**).
- chaque ligne de **M** contient une seule opération.
- chaque **chiffre** est une chaîne de **7** caractères (codes binaires du tableau T).

Exemple :

Pour M de 3 lignes et 3 colonnes, si le contenu de la matrice M avant calcul des opérations d'additions est :

A	*B*	*A+B*
1101101	**01100111110000**	
1111001	**1011111**	
11111111111110	**11100000110000**	

Après le calcul des opérations d'addition (A+B) et le codage la matrice contient :

A	*B*	*A+B*
1101101	01100111110000	**01100111110111**
1111001	1011111	**1110111**
11111111111110	11100000110000	**01100001011011010000**

En effet : **1101101+01100111110000 = 01100111110111** ➔ **2 + 47 = 49**
2 + 4 7 4 9

8 0 1 5 1

11111111111110 + 11100000110000 = 01100001011011000 0 → 80 + 71 = 151

7 1

NB : *l'élève n'est pas appelé à remplir le tableau T et on suppose que les deux colonnes de la matrice sont remplies d'avance.*

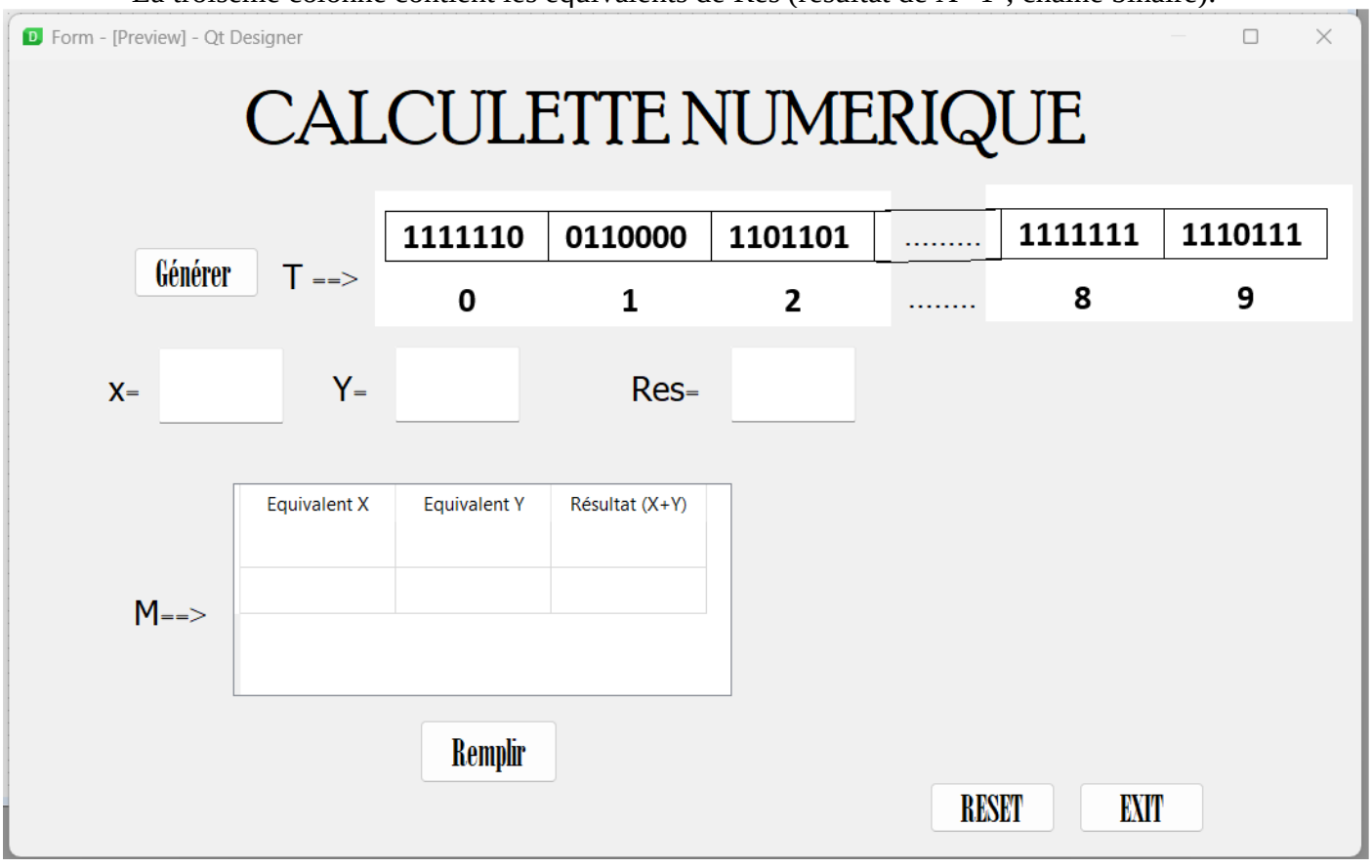
Partie théorique (Algorithme) :

1) Ecrire l'algorithme du programme principal permettant de saisir le nombre de lignes et de compléter la troisième colonne de la matrice M, enfin l'afficher.
2) Ecrire les algorithmes des modules envisagés.

Partie Pratique (Python + Qt Designer) :

Concevoir une interface phraphique à travers laquelle l'utilisateur pourra effecteur des opérations d'addition et afficher les résultats dans une matrice de N lignes et 3 colonnes.

- D'abord, vous devrez afficher les équivalents binaires (segments allumés et éttients) des chiffres de 0 à 9, dans un tableau (Table Widget) . en appuyant sur le bouton **''Générer''**.
- Saisir deux nombres X et Y d'au moins un chiffre puis calculer leur somme (Res)
- Remplir la matrice M de façon :
  - La première colonne contient les équivalents de X (chaine binaire).
  - La deuxième colonne contient les équivalents de Y (chaine binaire).
  - La troisème colonne contient les équivalents de Res (résultat de X+Y , chaine binaire).

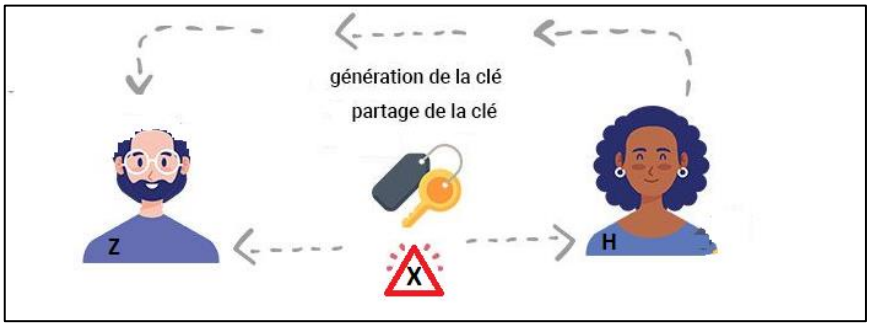


Exercice n°4 : (Cryptage)

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)**

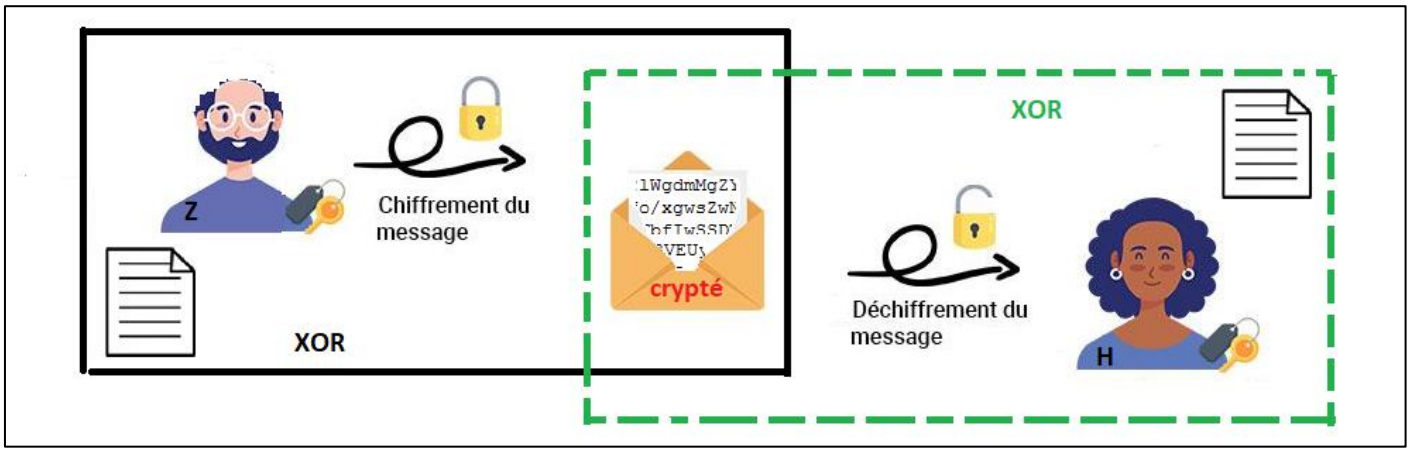
Etape 6 : Les nombres décimaux obtenus correspondent aux **codes ASCII** des caractères du **message crypté**.

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,

le code binaire de 83= ORD("**S**")

le code binaire de 71= ORD("**G**")

XOR

	0	1	0	1	0	0	1	1
	0	1	0	0	0	1	1	1
=	0	0	0	1	0	1	0	0

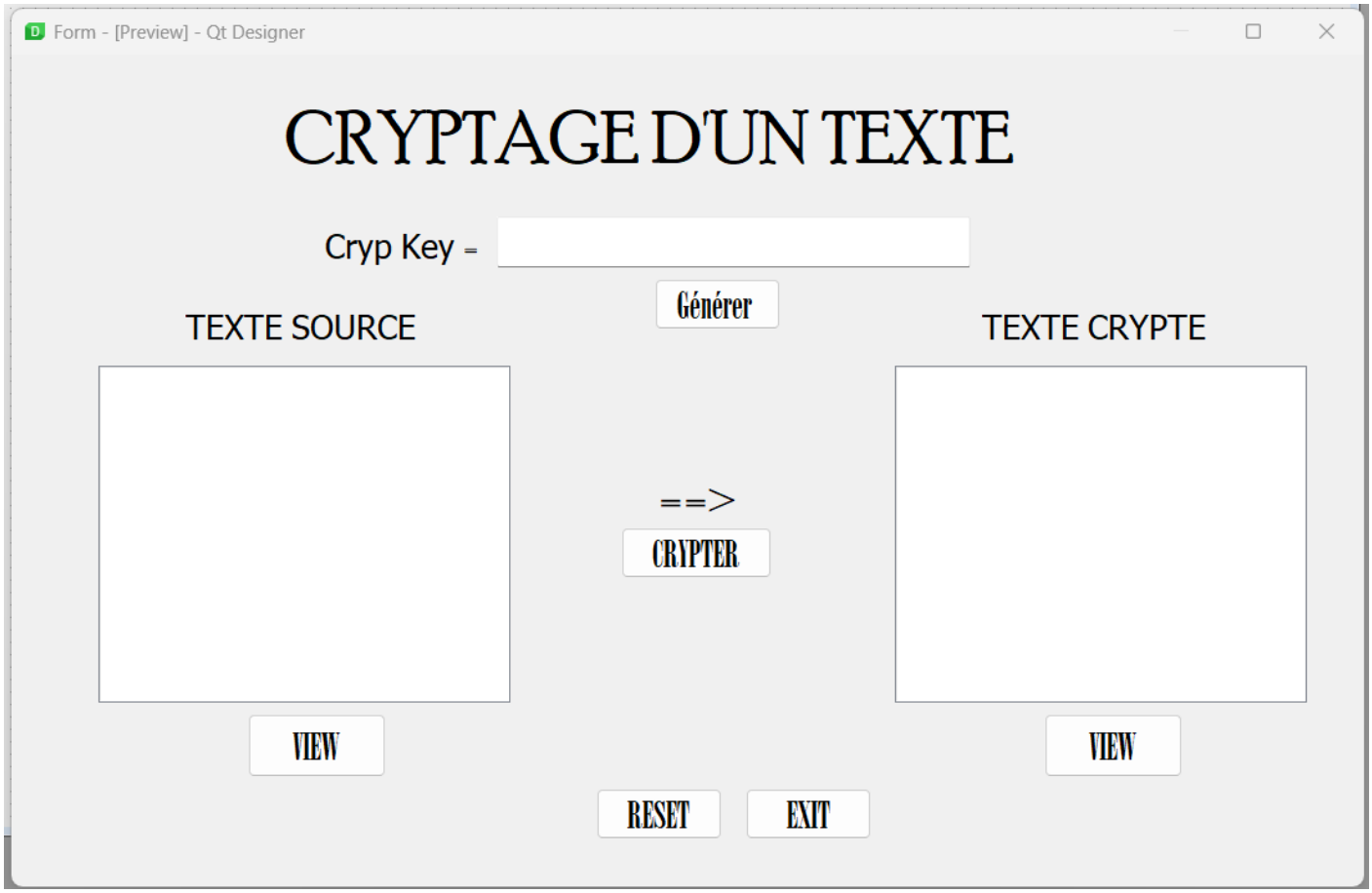
Partie théorique (Algorithme) :

1) Ecrire un algorithme modulaire du programme principal
2) Ecrire les algorithmes des modules envisagés.

Partie Pratique (Python + Qt Designer) :

Concevoir une interface graphique permettant de répondre à l'objectif de l'exercice (crypter un texte).
L'interface contient essentiellement :

- Un bouton ''Générer'' permettant de générer la clé de cryptage dans la zone de saisie convenable.
- Un bouton ''View'' permettant d'afficher le contenu du fichier source **"F_init.txt"** dans la List Widget 1
- Un bouton ''CRYPTER'' permettant d'appeler la fonction de cryptage comme décrit précédemment.
- Un deuxième bouton ''View'' permettant d'afficher le contenu du fichier résultat **"F_res.txt"** dans la List Widget 2



NB : Pour toutes les intefaces graphiques ajouter deux boutons **''EXIT''** et **''RESET''** .