RÉPUBLIQUE TUNISIENNE **MINISTÈRE DE L'ÉDUCATION** **EXAMEN DU BACCALAURÉAT** SESSION **2019**	**Session de contrôle**	
	Épreuve : **Algorithmique et Programmation**	Section : **Sciences de l'informatique**
	Durée : **3h**	Coefficient de l'épreuve : **2.25**

Le sujet comporte 4 pages numérotées de 1/4 à 4/4.

Important :

Chaque solution développée par le candidat, sous forme d'une analyse ou d'un algorithme doit être accompagnée d'un tableau de déclarations des objets ayant la forme suivante :

Objet	*Type / Nature*	*Rôle*

Exercice 1 (2 points)

Reproduire le tableau suivant, puis pour chacune des propositions ci-après, remplir la case correspondante par la lettre de la réponse correcte.

Proposition	**1**	**2**	**3**	**4**
Réponse				

1. Dans un algorithme récursif, la condition d'arrêt permet :
   - **a)** de changer la valeur d'un paramètre.
   - **b)** de stopper les appels récursifs.
   - **c)** d'appeler le module autant de fois, dans son corps.
2. Le processus de partitionnement dans la méthode du tri rapide permet de :
   - **a)** diviser un tableau en trois parties égales.
   - **b)** fusionner deux tableaux triés.
   - **c)** mettre tous les éléments inférieurs au pivot à sa gauche et les autres à sa droite.
3. L'algorithme qui permet de calculer la factorielle d'un nombre est :

a)

```
0) DEF FN Fact (n:octet) : entier
1) Si n = 0 Alors Fact ← n
   Sinon Fact ← n * FN Fact (n-1)
   Fin Si
2) Fin Fact
```

b)

```
0) DEF FN Fact (n:octet) : entier
1) Si n > 1
   Alors Fact ← n * FN Fact (n-1)
   Fin Si
2) Fin Fact
```

c)

```
0) DEF FN Fact (n:octet) : entier
1) F ← 1
   Pour i de 2 à n faire
      F ← F * i
   Fin Pour
2) Fact ← F
3) Fin Fact
```

4. Dans le problème des tours de Hanoï, le nombre minimal de déplacements de n disques nécessaire pour arriver à l'état final est :
   - **a)** $2^n - 1$
   - **b)** 2^{n-1}
   - **c)** $2^n + 1$

Exercice 2 (3 points)

Soit l'algorithme suivant de la fonction intitulée **Quoi** :

```
0) DEF FN Quoi (T : Tab ; a, deb, fin : entier) : ……
1) m ← (deb + fin) Div 2
2) Si (deb > fin) Alors Quoi ← faux
     Sinon Si (T[m] = a) Alors Quoi ← vrai
       Sinon Si (T[m] > a) Alors Quoï ← FN Quoi (T, a, deb, m-1)
         Sinon Quoi ← FN Quoi (T, a, m+1, fin)
   Fin Si
3) Fin Quoi
```

NB : T est un tableau d'entiers trié dans l'ordre croissant.

Travail demandé :

1- Donner le type de retour de la fonction **Quoi**.

2- Donner la valeur retournée par la fonction **Quoi** (T, 6, 1, 5) pour les deux cas suivants :

Cas 1 :

T	-2	3	6	10	12

Cas 2 :

T	-2	3	4	10	12

3- Déduire le rôle de la fonction **Quoi**.

Exercice 3 (4 points)

Le jeu **des tours de Hanoï** est un jeu de réflexion qui consiste à déplacer **N** disques de diamètres différents d'une tour de départ (**D**) à une tour d'arrivée (**A**) en passant par une tour intermédiaire (**Inter**), et ceci en un minimum de coups, tout en respectant les règles suivantes :

- On ne peut pas déplacer plus d'un disque à la fois.
- On ne peut placer un disque que sur un autre disque plus grand ou sur un emplacement vide.

Soit l'algorithme récursif suivant de la procédure qui simule le jeu des tours de Hanoï :

```
0) DEF PROC Hanoi (N : entier ; D, A, Inter : caractère)
1) Si (N > 0) Alors
   PROC Hanoi (N-1, D, Inter, A)
   Ecrire ("Déplacer un disque de ", D, " à ", A)
   PROC Hanoi (N-1, Inter, A, D)
   Fin Si
2) Fin Hanoi
```

Travail demandé :

1) Pour sauvegarder les différents déplacements des **N** disques, on se propose d'utiliser un fichier d'enregistrements intitulé "**Deplace.dat**".

   a) Proposer une déclaration d'un type enregistrement permettant de représenter un déplacement , sachant que chaque déplacement est caractérisé par une tour de départ et une tour d'arrivée.

   b) Proposer une déclaration d'un type pour le fichier "**Deplace.dat**".

2) Réécrire l'algorithme de la procédure **Hanoi** en apportant les modifications nécessaires pour qu'il procède à la sauvegarde des différents déplacements des disques dans le fichier d'enregistrements "**Deplace.dat**".
   NB : Le fichier résultat "**Deplace.dat**" est créé et ouvert dans le programme appelant.

3) Ecrire un algorithme d'un module permettant d'afficher le contenu du fichier d'enregistrements "**Deplace.dat**".

Problème (11 points)

Parmi les méthodes de cryptage utilisées pour crypter un texte **msg** formé de caractères alphabétiques, on cite la méthode **OU exclusif simple** utilisant une **clef** de cryptage, qui est une chaîne binaire représentée sur **N** bits avec **N** est un multiple de 7.

Cette méthode consiste à réaliser les étapes suivantes :

Etape1 :

Sachant que chaque caractère de **msg** est représenté sur 7 bits, on procède à un ajout du caractère antislash "\" à la fin de **msg**, autant de fois que nécessaire, jusqu'à ce que la longueur de sa représentation binaire devienne un multiple de **N**.

Etape2 :

On découpe **msg** en blocs de caractères pour que chacun soit représenté sur un nombre de bits égal au nombre de bits de la clef, c'est-à-dire que chaque bloc contient **N Div 7** caractères.

Etape3 :

On détermine la représentation binaire de chaque bloc par la concaténation de l'équivalent binaire, sur 7 bits, du code Ascii de chaque caractère du bloc.

Etape4 :

A chaque bloc, on applique un **Ou exclusif (OUex)** avec la clef, bit par bit.
On rappelle que **0 OUex 0 = 0, 1 OUex 0 = 1** et **1 OUex 1 = 0**

Etape5 :

Le texte crypté est obtenu en concaténant respectivement les résultats trouvés dans l'étape précédente.

Exemple **:**
Pour **clef** = "10101100101101" et **msg** = "Savon"

Etape1 :

Puisque le texte à crypter est formé de 5 caractères et chaque caractère est représenté sur 7 bits, la longueur de la représentation binaire du texte est 35 (5 caractères * 7 bits).

Comme la longueur de la clef est égale à 14 et comme 35 n'est pas un multiple de 14, il est nécessaire d'ajouter à droite du texte le caractère antislash "\" une seule fois afin que la nouvelle longueur (6*7=42) devienne un multiple de 14.

Etape2 :

Etant donné le texte modifié (**msg** = "Savon\"), on procède à un découpage en blocs de 14 bits c'est à dire en blocs de deux caractères (14 Div 7).

D'où, Bloc1 = "Sa", Bloc2 = "vo" et Bloc3 = "n\"

Etape3 :

La représentation binaire de chaque bloc est obtenue par la concaténation de l'équivalent binaire, sur 7 bits, du code Ascii de chaque caractère.

La représentation binaire du Bloc1 est "1010011 1100001"
S a

La représentation binaire du Bloc2 est "1110110 1101111"
v o

La représentation binaire du Bloc3 est "1101110 1011100"
n \

Etape4 :

En appliquant à chaque bloc la fonction **Ou exclusif** avec la clef, on obtient les résultats suivants :

Bloc1 **OUex** clef = 10100111100001 **OUex** 10101100101101 = 00001011001100
Bloc2 **OUex** clef = 11101101101111 **OUex** 10101100101101 = 01000001000010
Bloc3 **OUex** clef = 11011101011100 **OUex** 10101100101101 = 01110001110001

Etape5 :

De ce fait, le texte crypté obtenu en concaténant les résultats trouvés dans l'étape précédente est : **"00001011001100010000010000100111000111000 1"**

On se propose d'écrire un programme permettant de :

- saisir une chaîne binaire non vide représentant la clef de cryptage et ayant une longueur N multiple de 7 et inférieure ou égale à 21,
- crypter le contenu d'un fichier texte intitulé "**Initial.txt**", ligne par ligne, et d'enregistrer le résultat dans un 2ème fichier texte intitulé "**Final.txt**" en appliquant le principe de cryptage décrit précédemment.

NB :

- Chaque ligne du fichier "**Initial.txt**" contient un texte dont la longueur ne dépasse pas 36 caractères.
- Le candidat n'est pas appelé à remplir le fichier "**Initial.txt**".

Travail demandé :

1- Analyser le programme principal en le décomposant en modules.
2- Ecrire les algorithmes des modules envisagés.