REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Devoir Surveillé ☐	Examen ■	Session principale	■
		Session de contrôle	☐
Matière :	Algorithmique et structures de données I	**Semestre** : 1	
Enseignant(s) :	MajdiJribi et Imène Mami	**Date**:12Janvier 2022	
Filière(s) :	MPI	**Durée**:1h30	
Barème :	7-13	**Documents** : autorisés	☐
Nombre de pages :	02	non autorisés	■

Exercice 1 (7 pts)

1- Ecrire, en langage C, la fonction **int lire_chaine(char *s, char *inv, int limite)** qui permet de lire une chaine de caractères **s** saisie au clavier dont la longueur ne doit pas dépasser une certaine **limite** et de l'inverser. La fonction enregistre la chaine de caractères dans **s**, son inverse dans **inv** et retourne sa longueur.

Exemple : limite = 6

Chaine en entrée : "ABCD EFG" **Chaine tronquée (enregistrée dans s) : "ABCD E"** **Chaine inversée: "E DCBA"**	**Chaine en entrée : "ABCD "** **Chaine tronquée (enregistrée dans s) : "ABCD "** **Chaine inversée: "DCBA"**

2- Ecrire, en langage C, la **fonction void crypt(char *chaine, int decalage, int limite)** qui permet de chiffrer les caractères de la chaine inversée obtenue dans la question précédente. On ne chiffre que les lettres de l'alphabet (minuscules et majuscules). Le remplacement des lettres se fait suivant la longueur du décalage. Si le déclage dépasse la dernière lettre, il faut dans ce cas revenir au début. Si on choisit par exemple un décalage de 3, la lettre *a* sera remplacée par la lettre *d* et la lettre *X* sera remplacée par le lettre *A*.

Exemple : limite = 6 et décalage=3

Chaine en entrée : "V./WyXABc"
Chaine tronquée : "V./WyX"
Chaine inversée (enregistrée dans inv de la question 1) : "XyW/.V"
Chaine cryptée: "AbZ/.Y"

Exercice 2 (13 pts)

On se propose de faire des traitements sur les objets 3D. Un objet 3D est représenté sous la forme de facettes triangulaires (sous la forme de triangles).

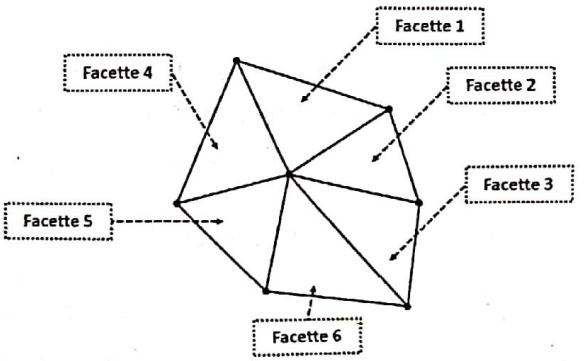

Figure 1 : Illustration d'un objet 3D

Un objet 3D est défini par la structure de données suivante :

typedef struct{

```
float ** points ;
int nb_points ;
int ** facettes;
int nb_facettes;
} Objet_3D ;
```

- Le champ **points** est une matrice de **nb_points** lignes et de trois colonnes. Une ligne **i** de cette matrice contient les composantes X, Y et Z du point d'indice **i** dans l'objet 3D.
- Le champ **nb_points** dénote le nombre de points dans la matrice **points**.
- Le champ **facettes** est une matrice d'entiers de **nb_facettes** lignes et de trois colonnes. Chaque ligne de cette matrice contient les trois indices des points de la matrice **points** qui forment les sommets du triangle.
- Le champ **nb_facettes** dénote le nombre de facettes (triangles) dans l'objet 3D.

Les figures 2 et 3 illustrent un élément de type Objet_3D.

	X	Y	Z
Coordonnées du point d'indice 0	0	0	0
Coordonnées du point d'indice 1	-1	0	0
	0	1	0
	1	0	0
Coordonnées du point d'indice 4	0	-1	0

Figure 2 : Exemple de la matrice **points** d'un objet 3D avec **nb_points** égal à 5

Indices des points sommets du triangle 1	0	1	2
Indices des points sommets du triangle 2	0	2	3
	0	3	4
Indices des points sommets du triangle 4	0	4	1

Figure 3 : Exemple de la matrice **facettes** d'un objet 3D avec **nb_facettes** égal à 4

La figure 3 montre les triangles formés à partir de la matrice **points.** Quatre triangles sont construits. A titre d'exemple, Les sommets du premier triangle correspondent aux points d'indices 0, 1 et 2 de la matrice **points** de **la figure 2**

1- Ecrire, en langage C,la fonction **void Lect_Obj3D(Objet_3D* A)** qui permet de lire un objet 3D. Le remplissage des matrices **points** et **facettes** se fera d'une manière dynamique.

2- On définit un voisin d'un point d'indice **i** d'un objet A de type **Objet_3D**, le point d'indice **j** tel que **i** et **j** sont les indices de deux points sommets qui appartiennent à une même facette (triangle) de l'objet A.
Ecrire, en langage C, la fonction **int nb_Voisins(Objet_3D A, int indice)** qui permet de déterminer le nombre de voisins du point d'indice **indice** dans l'élément A de type **Objet_3D**.

3- On dit qu'un élément de type Objet_3D est régulier si tous ses points ont le même nombre de voisins.
Ecrire, en langage C, la fonction **int Est_Regulier(Objet_3D A)** qui permet de vérifier si l'objet 3D A est régulier.

4- Etant donné un point P de coordonnées X, Y et Z (P(X,Y,Z)), écrire, en langage C, la fonction **Void Exist_NbVoisin(Objet_3D A, float X, float Y, float Z, int* exist, int* nb_voisins)** qui permet de vérifier si le point P appartient à l'objet 3D A et de déterminer le nombre de ses voisins.