NOM :	Prénom :

INF102 – Programmation en C

Durée : 1h30,
Nombre de pages : 8
Documents et calculatrices autorisés

Les exercices seront réalisés directement sur les sujets.

Exercice 1 – Mystère *(2 points)*
On considère le programme en C suivant :

```c
#include <stdio.h>

int main()
{
    const int n = 10;
    int T[n];
    int i,j;

    for (i = 0; i < n; i++)
        T[i] = 0;
    T[1] = T[2] = 1;

    for (j = 0; j < 3; j++)
        for (i = 1; i < n; i++)
            T[i] = T[i] + T[i-1];

    for (i = 0; i < n; i++)
        printf("%d ", T[i]);

    return 0;
}
```

Qu'affiche ce programme ?

Exercice 2 – Fonctions *(2 points)*

Q 2.1) Écrire une fonction int *Puissance (int a, int i)* qui calcule a^i, sans utiliser la fonction *pow()*.

```
int Puissance (int a, int i) {




}
```

Q 2.2) Écrire une fonction *int Somme (int a, int n)* qui calcule : $\sum_{i=0}^{i=n} a^i$

```
int Somme (int a, int n) {




}
```

Exercice 3 – Structures *(7 points)*

Pour un magasin de location de scooters, on souhaite réaliser un programme pour gérer son parc de scooters. Un scooter est représenté par un type Scoot qui est caractérisé par les éléments suivants :

- Le modèle du scooter
- Son no d'identification
- Son kilométrage
- Son état

Le nom du modèle du scooter *modele* est une chaîne de caractères (20 caractères maxi), son kilométrage est float, le no d'identification est un entier, et un autre entier *etat* donnant l'état de location du scooter (1 s'il est en location, 0 sinon)

Le parc de scooters est représenté sous la forme d'un tableau de 100 scooters maximum nommé *TabS*. Ce tableau sera passé comme argument dans toutes les fonctions où il sera nécessaire. Il ne sera pas considéré comme une variable globale.

```
typedef struct {
   char modele[20];
   int id;
   float km;
   int etat;
} Scoot;

#define Taille 100

Scoot TabS[Taille] ;
```

Q 3.1) Ecrivez une procédure *void Menu( )* qui affiche le menu suivant

1 : Louer un scooter

2 : Retour d'un scooter

3 : Etat d'un scooter

4 : Etat du parc de scooter

0 : Fin du programme

```
void Menu( ) {

}
```

Q 3.2) Ecrivez une procédure *void AfficheEtat(Scoot TabS[], int d)* qui prend comme argument un numéro d'identification *d* d'un scooter et affiche tous les élément du scooter : son modèle, son numéro d'identification, son kilométrage et son état de location

```
void AfficheEtat (Scoot TabS[], int d){

}
```

Q 3.3) Ecrivez une fonction *int EstEnParc (Scoot TabS[], int d)* qui prend comme argument un numéro d'identification *d* d'un scooter et renvoie 1 si le scooter existe dans le parc et renvoie 0 sinon.

```
int EstEnParc (Scoot TabS[], int d) {
```

```
}
```

Q 3.4) Ecrivez une procédure *void RetourScoot (Scoot TabS[], int d)* qui prend comme argument un numéro d'identification *d* d'un scooter et ajoute le nombre de kilométrage effectués. Le scooter est alors marqué comme étant disponible.

```
void RetourScoot (Scoot TabS[], int d) {




}
```

Q 3.5) Ecrivez une procédure *void AfficheEtatParc(Scooter TabS[])* qui affiche un état résumé du parc de scooters, c'est à dire :

- Le nombre total scooters en location et leur numéro d'identification
- Le kilométrage moyen de l'ensemble de scooters

```
void AfficheEtatParc(Scoot TabS[]){




}
```

Q 3.6) Ecrivez une procédure *void ElimineDoublons(Scoot TabS[])* qui élimine les doublons, c'est-à-dire les éléments du tableau (les scooters du parc) dont **le nom de modèle ainsi que le numéro d'identification sont identiques**. Si l'on trouve deux éléments ayant ainsi le même nom de modèle et le même numéro d'identification, on supposera qu'un employé du parc aura par erreur entré le scooter en deux fois. Il s'agira donc de corriger cette erreur en rassemblant les deux éléments en un seul : conserver le premier et supprimer le deuxième.

```
void ElimineDoublons(Scoot TabS[]){




}
```

Q 3.7) Ecrire un programme principal **int main()** qui permet d'appeler toutes ces précédentes fonctions et procédures, en n'oubliant pas de déclarer les variables nécessaires.

```
int main(){

// déclaration et initialisation d'un tableau de Scooters
Meuble TabS[Taille] ;
TabS[0].modele = "Yamaha" ;  TabS[0].id =  11 ; TabS[0].km =  120 ; TabS[0].etat =  1 ;
TabS[1].modele = "Honda" ;  TabS[1].id =  15 ; TabS[1].km =  30 ; TabS[1].etat =  0 ;
TabS[2].modele = "Suzuki" ;  TabS[2].id =  3 ; TabS[2].km =  0; TabS[2].etat =  0 ;

// Retourner un scooter, avec un no d'identification 15, qui est en location




// Afficher l'état du parc




// tester si le scooter avec un no d'identification 2 existe ou pas




// Afficher l'état d'un scooter avec un no d'identification 11
```

```
// élimine les doublons dans le tableau du parc




}
```

Exercice 4 – Fonction accordéon *(2 points)*

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19

Écrire une fonction qui *affiche* un parcours en accordéon d'un tableau à deux dimensions. Plus précisément, on parcourt la première ligne de gauche à droite puis la seconde de droite à gauche et ainsi de suite en alternant le sens de parcours des lignes.

Exemple : pour le tableau à gauche :

On doit obtenir l'affichage suivant : 0, 1, 2, 3, 7, 6, 5, 4, 8, 9, 10, 11, 15, 14, 13, 12, 16, 17, 18, 19

```
void accordeon(Tab[m][n], int m, int n) {




}
```

Exercice 5 – Chaînes de caractères (*3 points)*

Q 5.1) Ecrire une fonction *int Longueur(char chaine [])* qui reçoit une chaine de caractères et retourne sa longueur. Il ne faut pas utiliser la fontion *strlen()* de la bibliothèque <*string.h*>.

```
int Longueur(char chaine[] ) {




```

Q 5.2) Ecrire une procédure *void SupprimeNonAlphabetiquequi (char chaine[])* qui reçoit une chaine de caractères *chaine* et supprime de cette chaine les caractères qui ne sont pas alphabétiques

void SupprimeNonAlphabetiquequi (char chaine[]) {

}

Q 5.3) Ecrire une fonction int frequenceLettre(char c, char chaine[]) qui retourne la fréquence d'une lettre *c* donnée dans une chaîne de caractères *chaine*.

int frequenceLettre(char c, char chaine[]) {

}

Exercice 6 – Pointeurs (*4 points)*

5.1) Qu'affichent les programmes suivants sur écran :

```
int x; *(&x) = 7;
printf("%d", x);
```

```
int t[] = {1,2,3}; *(t+2) = 5;
printf("%d %d %d\n",t[0],t[1],t[2]);
```

```
int t[] = {1,2,3,4,5}; int *x = t+2;
int i = 0; for(i=0;i<2;i++)
        printf("%d\n", x[i]);
```

5.2) Ecrire une procédure *void stat (int* tab, int n, int* min, int* max, float* moy)* qui retourne le minimum *min*, le maximum *max* et la moyenne *moy* d'un tableau de *n* entiers.

```
void stat(int* tab, int n, float* moy, int* min, int* max) {




}
```