SR01 Devoir 1
Programmation C

• Ce TP peut se faire en binôme. Les deux étudiants formant un binôme doivent appartenir au même groupe de TD.

• Les différents binômes devront remettre un seul fichier *.zip contenant les codes sources ainsi qu'un rapport électronique (pdf) de quelques pages.

Exercice 1

Pour chacun de ces programmes, veuillez l'exécuter et donner une explication au résultat obtenu:

```c
#include<stdio.h>
int main(){
  int A=20,   B=5,   C=-10,   D=2;
  printf("%d \n", A&&B||!0&&C++&&!D++);
  printf("c=%d d=%d \n", C, D);
}
```

Programme 1

```c
#include<stdio.h>
int main(){
  int p[4]={1,-2,3,4};
  int *q=p;
  printf("c=%d\n", *++q**q++);
  printf("c=%d \n",*q);
}
```

Programme 2

```c
#include<stdio.h>
int main(){
  int A=20,   B=5;
  int C=!--A/++!B;
  printf("A=%d B=%d c=%d \n", A,B,C);
}
```

Programme 3

```c
#include<stdio.h>
int main(){
  int a=-8,b=3;
  a>>=2^b;
  printf("a=%d\n",a);
}
```

Programme 4

```c
#include<stdio.h>
int main(){
  int p[4]={1,-2,3,4};
  int *q=p;
  int d=*q&*q++|*q++;
  printf("d=%d\n", d);
  printf("q=%d \n",*q);
}
```

Programme 5

Exercice 2

1. Écrire un programme qui lit les notes de N étudiants de l'UTC dans un devoir de l'UV SR01 et les mémorise dans un tableau POINTS de dimension N.
2. Écrire des programmes pour rechercher et afficher :
   - La note maximale du devoir SR01,
   - La note minimale du devoir SR01,
   - La moyenne des notes du devoir SR01.
3. A partir des POINTS des étudiants, établir un tableau NOTES de dimension 7 qui est composé de la façon suivante :
   - NOTES[6] contient le nombre de notes 60
   - NOTES[5] contient le nombre de notes de 50 à 59
   - NOTES[4] contient le nombre de notes de 40 à 49
   - ...
   - NOTES[0] contient le nombre de notes de 0 à 9
4. Établir un graphique en nuage de points représentant le tableau notes. Utilisez le symbole 'o' pour représenter le point dans le graphique et affichez le domaine des notes en dessous du graphique.
5. Établir un graphique en bâtons représentant le tableau NOTES. Utilisez des symboles de votre choix pour la représentation des bâtons et affichez le domaine des notes en dessous du graphique.

Indication : Déterminer la valeur maximale MAXN dans le tableau NOTES et afficher autant de lignes sur l'écran. (Dans l'exemple ci-dessous, MAXN = 6).

Exemple :

- La note maximale du devoir SR01 est 58
- La note minimale du devoir SR01 est 13
- La moyenne des notes du devoir SR01 est 37.250000

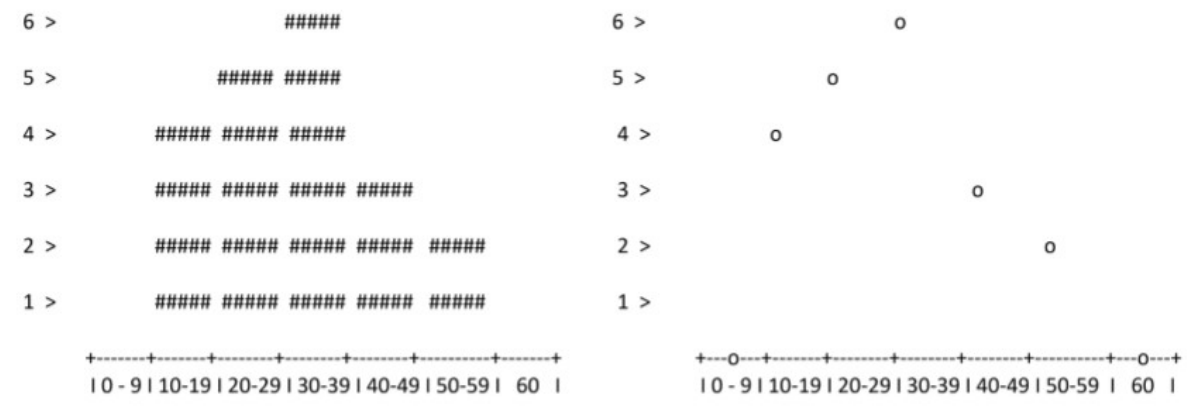

Figure 1 : Exemple d'affichage

Exercice 3

Le but de ce problème est de créer un programme en C pour gérer le parc de voitures d'une agence de location en manipulant les tableaux dynamiques, les structures de données, les fichiers, et l'allocation dynamique de la mémoire.

Une voiture est caractérisée par les éléments suivants:

- le modèle de la voiture,
- son n° d'immatriculation,
- son kilométrage,
- son état (disponible ou en cours de location).

Au final le programme que vous réaliserez devra présenter le menu ci-dessous :

```
------------------MENU--------------
1: Louer une voiture
2: Retour d'une voiture
3: Etat d'une voiture
4: Etat du parc de voitures
5: Sauvegarder l'état du parc
0: Fin du programme
```

Figure 2 : Menu à l'exécution du programme

Choix 1 : Le programme demande le n° d'immatriculation de la voiture à louer. Si la voiture n'existe pas, le programme signale une erreur; s'elle est déjà loué, le programme indique qu'elle est déjà en location sinon la voiture est marquée comme étant en location.

Choix 2 : Le programme demande le n° d'immatriculation de la voiture à retourner. Si la voiture n'existe pas, le programme signale une erreur; si la voiture n'était pas marquée comme étant en location, le programme le stipule sinon le programme demande le nombre de kilomètres effectués et les rajoute au kilométrage de la voiture. La voiture est alors marquée comme étant disponible.

Choix 3 : Le programme demande le n° d'immatriculation de la voiture dont on désire connaitre l'état. Si la voiture n'existe pas dans le parc, une erreur est signalée sinon son modèle, son n° d'immatriculation, son kilométrage et son état de location sont affichés.

Choix 4 : Le programme affiche un état résumé de l'ensemble du parc de voitures, i.e. :

- le nombre total de voitures,
- le nombre de voitures en location,
- le nombre de voitures disponible,

- le kilométrage moyen de l'ensemble des voitures.

Choix 5 : Le programme sauvegarde les voitures du parc dans un fichier binaire, le nom du fichier sera lu au clavier.

Après l'exécution d'une de ces cinq options, le programme réaffiche le menu.

Travail Demandé :

Chaque option du menu devra être implémentée sous la forme d'une fonction séparée de façon à faciliter l'écriture du programme (*Vous pouvez changer les prototypes proposés ci-dessous*).

1. Définir la structure Voiture.
2. Créer une fonction permettant de créer le parc de n voitures, par la suite cette fonction sera appelée au début de programme :

   void init (Voiture * voitures, int n)

3. Créer une fonction permettant de louer une voiture (*Voir Choix 1*):

   int louer (char * matricule, Voiture *voitures, int n)

4. Créer une fonction permettant le retour d'une voiture (*Voir Choix 2*) :

   int retour (char * matricule, Voiture* voitures, int n)

5. Créer une fonction permettant de renvoyer l'état d'une voiture (*Voir choix 3*) :

   int etat (char * matricule, Voiture* voitures, int n)

6. Créer une fonction permettant d'afficher l'état de parc de voitures (*Voir choix 4*) :

   void etatParc (Voiture* voitures, int n)

7. Créer une fonction permettant de sauvegarder les voitures du parc dans un fichier binaire (*Voir choix 5*) :

   void save (char * fichier, Voiture* voitures, int n)

8. Créer le programme principal (main).

Partie Bonus Optionnelle :

Réécrire la fonction init de la question 2 de façon à ce que les voitures soient récupérées à partir du fichier sauvegardé en question 7.