ECE Lyon - ING1 - Informatique 1 - Partiel

Nom : ..... Prénom : .....

Cahier des charges

Le responsable d'une agence de location de voitures souhaite obtenir un programme permettant de gérer son parc automobile.

Une voiture est caractérisée par les éléments suivants :

- Le type de véhicule (`1` = petit, `2` = moyen, `3` = grand, `4` = utilitaire)
- Le n° d'immatriculation (chaîne de caractères au format `"AA-123-BB"`)
- Le kilométrage
- L'état de disponibilité (`0` = disponible à la location, `1` = en cours de location)

Le programme présentera le menu ci-dessous :

```
------------------MENU------------------
1: Location d'une voiture
2: Retour d'une voiture
3: Etat d'une voiture
4: Etat du parc de voitures
5: Sauvegarde de l'etat du parc
0: Fin du programme
```

- **Choix `1`** (location) : En fonction du n° d'immatriculation :
  - Si la voiture n'existe pas : le programme signale une erreur
  - Si elle est déjà louée : le programme indique qu'elle est déjà en location
    - Sinon, la voiture est marquée comme étant en location.
- **Choix `2`** (retour) : En fonction du n° d'immatriculation :
  - Si la voiture n'existe pas : le programme signale une erreur
  - Si la voiture n'est pas en cours de location : le programme l'indique à l'utilisateur
  - Sinon, la voiture est marquée comme étant disponible.
- **Choix `3`** (état voiture) : En fonction du n° d'immatriculation :
  - 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 à l'écran.
- **Choix `4`** (état parc auto) : Le programme affiche un état résumé de l'ensemble du parc de voitures, c'est-à-dire :
  - Le nombre total de voitures
  - Le nombre de voitures en location
  - Le nombre de voitures disponibles
  - La liste des voitures, avec pour chacun, son n° d'immatriculation et sa disponibilité.
- **Choix `5`** (sauvegarde) : Le programme sauvegarde l'ensemble des données du parc automobile dans un fichier, sur le disque dur de l'utilisateur, dans son répertoire personnel (*`~/<nomDuFichier>.<extension>`*).

Après l'exécution d'une de ces 5 options, le programme réaffiche le menu tant que le choix `0` n'est pas choisi.

***NB1** : Pour chaque exercice, vous pouvez vous aider des fonctions détaillées en annexe, à la fin de ce sujet. Vous n'êtes évidemment pas limités à ces fonctions.*

***NB2** : Pour chaque exercice, vous pourrez supposer que les exercices précédents sont fonctionnels, même si vous n'avez pas réussi à les terminer.*

Exercice 1 : Organigramme du programme principal `main` (2 points)

Donnez un organigramme possible pour le programme principal afin de satisfaire au cahier des charges précédent. Chaque choix sera symbolisé par un appel de la fonction correspondante. Cet organigramme sera implémenté à l'exercice 8.

Exercice 2 : Chaînes de caractères (3 points)

Créez la fonction `char* immat()` qui demande le n° d'immatriculation à l'utilisateur (au clavier), vérifie qu'elle est au bon format (taille, format...) et la renvoie à la fonction appelante. La fonction demandera de rentrer un numéro de plaque d'immatriculation jusqu'à ce que le format soit correct.

Exercice 3 : Structures (2 points)

Définissez la structure « Voiture » avec les éléments indiqués dans le cahier des charges.

Exercice 4 : Tableaux dynamiques (3 points)

Déclarez dynamiquement un tableau nommé « `parcVoitures` », contenant `n` éléments « `Voiture` ». `n` étant saisi par l'utilisateur au clavier.

NB1 *: Vous pouvez définir le tableau comme contenant des « `Voiture` » même si vous n'avez pas réussi l'exercice précédent.*

NB2 *: On ne demande pas ici d'initialiser ce tableau mais simplement de le déclarer/l'allouer. Nous supposerons dans la suite de ce sujet que le tableau sera remplit de Voitures avec leurs propres informations. Veillez à ne pas provoquer de fuite mémoire.*

Exercice 5 : Correction d'erreurs (2 points)

Corrigez le programme suivant :

```
int etatParc(Voiture * parcVoitures, int nbVoitures) {
    int i, nbDispo = 0, nbIndispo = 0;
    printf("Il y a %d voitures au total\n, nbVoitures");
    for( i = 0 ; i < nbVoitures ; i++ ) {
        printf("%d", parcVoitures[i].immatriculation);
        if (parcVoitures[i].etat = 1) {
            printf("Voiture "Indisponible"\n");
            nbIndispo++;
        } else {
            printf("Voiture "Disponible"\n");
            nbDispo++;
        }
    }
    printf("Il y a %d voitures disponibles et %d voitures en location",
                    nbIndispo,                  nbDispo);
}
```

Exercice 6 : Pointeurs et structures (5 points)

Créez les fonctions permettant de louer une voiture (choix 1), rendre une voiture (choix 2) et renvoyer l'état d'une voiture (choix 3). Attention, des indications vous sont données page suivante.

```
void louer  (char * matricule, Voiture * parcVoitures, int nbVoitures);
void retour (char * matricule, Voiture * parcVoitures, int nbVoitures);
 int etat   (char * matricule, Voiture * parcVoitures, int nbVoitures);
```

***NB1** : Ces fonctions ne demandent pas à saisir le n° d'immatriculation (il est passé en paramètre).*

***NB2** : Vous pouvez largement vous inspirer du code de l'exercice précédent.*

Exercice 7 : Fichiers (3 points)

Créez la fonction `save` permettant de sauvegarder l'état du parc de voitures dans un fichier binaire (choix 5) :

```
void save (char * nomFichier, Voiture * parcVoitures, int nbVoitures);
```

(Bonus) Exercice 8 : Programme principal (2 points)

Créez le programme principal (`main`) qui affiche le menu et appelle les fonctions définies dans les précédents exercices en fonction du choix de l'utilisateur (organigramme correspondant : voir exercice 1).

(Bonus) Exercice 9 : Question de cours - La mémoire (2 points)

- Donnez la différence entre la mémoire centrale et la mémoire d'un disque dur.
- Donnez la différence entre un fichier texte et un fichier binaire. Où sont-ils stockés ?

Annexe / Aide

Chaînes de caractères (`<string.h>`)

```
char* strcpy(char* destination, char* source); // Copie la chaîne source dans la chaîne destination. Retourne l'adresse de la chaîne de destination.
char* strcat(char* s1, char* s2); // Concatène les chaînes s1 et s2 dans s1. Retourne l'adresse de la chaîne de destination (s1).
int strcmp(char* s1, char* s2); // Compare les chaînes s1 et s2. Retourne une valeur < 0 si s1 précède s2, une valeur > 0 si s1 succède à s2, et 0 si les deux chaînes sont identiques.
size_t strlen(char* s); // Retourne le nombre de caractères présents avant '\0' dans s (size_t éq. à int).
```

Fichiers (`<stdio.h>`)

```
// FICHIERS TEXTES :
fprintf(pFichier, "%d", varEntier); // Ajoute la valeur de l'entier ("%d") varAntier dans le fichier pointé par pFichier. Retourne le nombre de caractères écrits dans le fichier.
fscanf(pFichier, "%d", &varEntier); // Lis la prochaine valeur dans le fichier pointé par pFichier. Cette valeur, qui devrait être un entier ("%d"), sera stockée à l'adresse de varEntier. Retourne le nombre de valeurs qui ont été lues.
// FICHIERS BINAIRES :
fwrite(tabEntiers, sizeof(int), n, pFichier); // Ajoute n éléments de type int dans le fichier pointé par pFichier, à partir du tableau tabEntiers. Retourne le nombre d'éléments ajoutés.
fread(tabEntiers, sizeof(int), n, pFichier); // Lis n éléments de type int dans le fichier pointé par pFichier, puis les stocke dans le fichier tabEntiers. Retourne le nombre d'éléments lus.
fseek(pFichier, n * sizeof(int), SEEK_SET); // Positionne le curseur dans le fichier pointé par pFichier, en le décalant de n fois la taille d'un entier, à partir de SEEK_SET (début du fichier), SEEK_END (fin du fichier) ou SEEK_CUR (position actuelle du curseur). Retourne 0 en cas de succès, une autre valeur sinon.
```