INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
CENTRE VAL DE LOIRE

Sécurité et Technologies informatiques

3ème Année Cycle Ingénieur

TD4: Listes chainées

VERSION 1.0

PAR JF LALANDE, A. ABDALLAH

TABLE DES MATIERES

- Liste chaînée
- Compilation automatique avec l'outil ***make***, un gestionnaire de **dépendances entre fichiers**.
- Librairies dynamiques
- GNU gdb

2016/2017

1 Objectifs

Comme je suis sympa, vous trouverez sur la plateforme **Célène** un code "propre" de gestion d'une liste de véhicules. On dispose donc des fonctions suivantes :

```
// Initialise une liste vide
liste_vehicule * init_liste();

// Ajoute un vehicule en tete
void add_vehicule(vehicule *v,liste_vehicule * l);

// Enleve le premier vehicule et le retourne
vehicule * remove_first_vehicule(liste_vehicule * l);

// Retourne la tete de la liste
vehicule * get_first_vehicule(liste_vehicule * l);

// Retourne le dernier element de la liste
vehicule *get_last_vehicule(liste_vehicule * l);

// Detruit la liste des vehicules et les vehicules
void destroy_list_and_vehicules(liste_vehicule * l);

// Taille de la liste
int size(liste_vehicule * l);
```

Bien sûr, les véhicules sont toujours :

```
typedef struct {
        char nom_modele[20];
        int puissance;
        float vitesse_max;
} vehicule;
```

Le premier problème à résoudre lorsque l'on à affaire avec du code source est de se familiariser avec celui-ci. Il n'est pas forcément indispensable d'avoir compris l'intégralité du code source pour pouvoir l'utiliser car en général, les signatures des fonctions suffisent.

NB : Créer un fichier de compilation automatisée *Makefile* pour votre projet

Exercice 1

Etudiez le code de gestion de la liste. Vous êtes confronté au problème classique du programmeur qui doit utiliser un code qu'il n'a pas programmé lui-même. Il vous faudra bien 10 minutes pour vous familiariser avec l'implémentation choisie. Vous devez être capable de répondre aux questions suivantes :

- Dans la structure *liste_vehicule*, à quoi sert le pointeur v ?
- Dans la structure *liste_vehicule*, à quoi sert le pointeur next ?
- Dans le code de *get_first_vehicule*, on retourne directement *l->next->v*, et pas *l->v*. Pourquoi ?

La dernière question est importante :

Car elle montre comment le programmeur a choisi d'implémenter sa liste. La liste vide est constituée d'un élément de type *liste_vehicule* qui possède un *l->v* nul et un *l->next* nul lui aussi. Ce "premier" élément ne sert à rien (c'est une sorte de sentinelle). Quand on ajoute un élément à la liste vide, cet élément est chainé au premier (il se retrouve donc second dans le chainage).

Exercice 2

En vous aidant du code source, dessinez :
– La liste vide
– La liste contenant un élément
– La liste contenant 2 éléments

Exercice 3

Ecrire une fonction qui permet de chercher un véhicule dont on connait le nom (le nom est représenté par un ***char** **). Cette fonction renvoie un pointeur sur ce véhicule.

vehicule* cherche_vehicule(char nom[20],liste_vehicule * L) ;

Exercice 4

Ecrire une fonction qui permet d'enlever un véhicule dont on connait le nom. Cette fonction enlèvera l'élément de la liste. Si cet élément se trouve à plusieurs endroits de la liste, seule la première occurrence est retirée.

vehicule* enlever_vehicule(char nom[20],liste_vehicule * L) ;

Exercice 5

Ecrire une fonction non récursive qui permet de renverser une liste.

liste_vehicule *renverser_liste(liste_vehicule *l) ;

N.B : réutiliser certaines fonctions déjà définies.

Exercice 6

Ecrire une fonction qui permet de faire l'union de deux listes. Prévoir un entier "booléen" qui permet d'indiquer si l'on souhaite conserver les doublons ou non. Si l'on souhaite conserver les doublons, votre fonction est en $O(1)$ sinon, quelle est sa complexité ?

liste_vehicule *union_liste(liste_vehicule *l1, liste_vehicule *l2, int doublons_autorises) ;

Exercice 7

Construire les librairies dynamiques *libvehicule* et *libliste*. Modifier votre fichier Makefile et régénérer le projet.

Exercice 8

Compiler votre projet avec l'option –g (changer dans Makefile). Exécuter votre programme pas-à-pas avec l'outil gdb.

Exercice 9

Réimplanter la fonction enlever_vehicule avec un pointeur double
vehicule* enlever_vehicule2(char nom[20],liste_vehicule ** L) ; // tester

Exercice 10

Réimplanter la fonction renverser_liste avec un pointeur double :
void renverser_liste(liste_vehicule **L) ;// tester