Lebanese University
Faculty of Sciences 1

I2204 F – INFO216 F

Programmation Impérative

Final 2019-2020
Mer. 12 Fév. am
Durée : 2 heures

Exercice I : Échange de nœuds dans une LDC (25 pts)

Écrire la function **swap** qui, étant donnés une liste doublement chaînée (LDC) et deux indices entiers, échange les nœuds à ces indices. La fonction doit échanger les nœuds et non pas les données contenues dans les nœuds car les nœuds peuvent être pointés par d'autres pointeurs. Par conséquent, nous voulons manipuler les pointeurs dans la liste chaînée afin de réorganiser les nœuds. Au cas où les indices donnés seraient hors limites, aucun échange ne devrait avoir lieu. De plus, il faut prendre soin lorsque les indices donnés sont des nombres consécutifs, afin de ne pas faire de boucles à l'intérieur de la LDC. Nous donnons ci-dessous un exemple d'application de la fonction swap. On rappelle le type :

```
typedef struct node{
    int data;
    struct node *prev, *next;
}node;
```

Exemple d'application :

```
List: 10 ⇌ 11 ⇌ 12 ⇌ 13 ⇌ 14
//swap(&head, 2, 1);
List: 10 ⇌ 12 ⇌ 11 ⇌ 13 ⇌ 14
//swap(&head, 0, 3);
List: 13 ⇌ 12 ⇌ 11 ⇌ 10 ⇌ 14
```

Exercice II : Tables de hachage (25 pts = 10 +15)

Les tableaux, par opposition aux listes chaînées, ont l'avantage de l'accès rapide aux éléments et l'inconvénient de l'altération coûteuse de la structure (opérations d'insertion et de suppression). Les tables de hachage tirent le meilleur parti de chaque monde. Les tables de hachage (hashtables) sont des tableaux de listes chaînées appelées compartiments (buckets). Ci-dessous, un exemple de table de hachage, et les types de données utilisés.

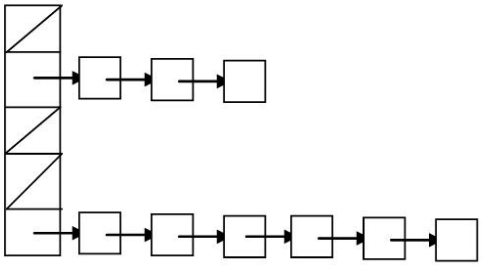

table de hachage

```
#define size 5

typedef struct element{ //...
} element;

typedef struct node{
    element data;
    struct node *next;
} node;

typedef struct hashtable{
    node* array[size];
} hashtable;
```

types de données utilisés

1- Écrire la fonction **avgBucketLen** qui calcule le nombre moyen d'éléments dans les compartiments (buckets) non vides de la table de hachage. Exemple: en considérant la table de hachage de la figure ci-dessus; nous avons cinq compartiments, dont deux non vides contenant neuf éléments au total, donc la longueur moyenne du compartiment est de 4.5.

Afin d'ajouter un élément donné à une table de hachage, il faut d'abord calculer le code de hachage de l'élément en appelant la fonction hash(). Ensuite, calculer le code modulo la taille du tableau pour obtenir un indice d'entrée. Enfin pousser l'élément à la tête de la liste chaînée située à cet indice.

```
int hash(element elt);
```

2- On supposons la fonction hash() déjà définie. Écrire la fonction **add** qui ajoute un élément donné dans une table de hachage donnée comme décrit.

Exercice III : Manipulation de fichiers et de chaînes (20 pts)

Un moyen simple de stocker un carnet d'adresses ou un répertoire téléphonique est dans un fichier texte avec une ligne par entrée. Nous avons un fichier contenant une collection de noms, de bureaux et de numéros de téléphone. Voici quelques exemples de lignes du fichier testfile.txt.

testfile.txt
```
Ima EE Professor#EE B 735#3-1415
Fearless Leader#EE1 116#3-6515
E. Fudd#EEGAD 548#2-4784
B. Bunny#EE 037#254-5512
```

Chaque ligne contient le nom d'une personne, l'emplacement du bureau (bâtiment et numéro de chambre) et le numéro de téléphone. Il y a un seul caractère de hachage (#) entre les champs consécutifs.
Pour une raison quelconque, le bâtiment EE a été désigné de plusieurs façons au fil des ans: 'EE', 'EE B' (avec un espace entre 'EE' et 'B'), 'EE1', 'EE 1' (avec un espace) et 'EEB'. Nous aimerions produire une copie nette de ce fichier en remplaçant les cinq nominations du bâtiment EE par 'EEB'. Par exemple, lors de l'exécution sur l'entrée ci-dessus, la sortie doit être la suivante:

sortie standard
```
Ima EE Professor#EEB 735#3-1415
Fearless Leader#EEB 116#3-6515
E. Fudd#EEGAD 548#2-4784
B. Bunny#EEB 037#254-5512
```

Pour ce problème, écrire en C la function **normalize** qui, étant donné le nom d'un fichier au format décrit, copie son contenu dans la sortie standard, en changeant toutes les références au bâtiment EE en 'EEB' comme décrit ci-dessus, sinon, la fonction ne doit pas modifier le contenu. On suppose que les données sont propres: chaque ligne contient exactement trois champs séparés par des caractères '#', il n'y a pas d'espaces supplémentaires avant ou après chaque caractère '#', il y a un espace entre le nom du bâtiment et le numéro de la pièce, etc.

Bonne Chance!