Data Structures and Algorithms Assignment 4: Analysis of Algorithms and Hash Table

This assignment contains two parts: 5 MCQ and 3 Programming questions. The templates for Programming Questions 1-3 are given as separated files (Q1_template.c, Q2_template.c, and Q3_template.c). You must use them to implement your functions. The program contains a main() function, which includes a switch statement to execute different functions that you should implement. You need to submit your code to the https://www.hackerearth.com (Email invitation will be sent to your school account). You need to submit your code to the NTULearn (State your name and your lab group in your submission). Deadline for assignment submission: March 30, 2024 (Saturday) 11.59 pm.

Programming Question 1 (Coalesced Hashing): Coalesced hashing is a combination of closed addressing and linear probing. Each slot is not only storing the key, but also the link (index) to the next slot. The structure of hash slots is given below.

```
enum Marker {EMPTY,USED};
typedef struct _slot{
    int key;
    enum Marker indicator;
int next;
} HashSlot;
```

The insertion and searching function prototypes are given as follows:

```
int HashInsert(int key, HashSlot hashTable[]);
int HashDelete(int key, HashSlot hashTable[]);
```

Both functions' return value is an index of the slot where the key is inserted and searched respectively. For insertion function, inserting a duplicate key will return -1. Return value larger than the table size implies that the table is full. For searching function, finding a non-existing key will return -1.

For example, key 14 in inserted at slot 0, and is linked to the next slot at index 1.

Hash(key) = key % 7
1) Insert 14
2) Insert 7
3) Insert 5
4) Insert 15
5) Insert 13
6) Insert 19

index	key	next
0	14	1
1	7	2
2	15	-1
3	19	-1
4		-1
5	5	3
6	13	-1

Programming Question 2 (Doubly Linked List for Closed Addressing Hash Table): Modify the closed addressing hash table in Lab 6 Question 4 to perform insertion and deletion by using a doubly linked list below.

```c
typedef struct _listnode{
    int key;
    struct _listnode *next;
    struct _listnode *pre;
} ListNode;

typedef struct _linkedlist{
   int size;
   ListNode *head;
} HashTableNode;

typedef struct _hashTable{
   int hSize;
   int nSize;
   HashTableNode *Table;
} HashTable;
```

The following utility functions are provided. The details can refer to the given template:

```c
ListNode* HashSearch(HashTable, int);
```

The insertion and deletion function prototypes are given as follows:

```c
int HashInsert(HashTable* Q1, int key);
int HashDelete(HashTable* Q1, int key);
```

Both functions' return value indicates success (1) or failure (0).

Programming Question 3 (Open Addressing of Double Hashing): Implement an open addressing hash table by double hashing to perform insertion and deletion. The structure of hash slots is given below and a hash table with **37** hashslots is created in the main function. The structure of hash slots is given below.

```c
enum Marker {EMPTY,USED,DELETED};
typedef struct _slot{int key;enum Marker indicator;} HashSlot;
```

The hash functions are provided. The hash2() is the incremental hash function for double hashing.

```c
int hash1(int key);
int hash2(int key);
```

Note that when hash1 is arithmetic modulo, double hashing function can be simplified as:

H(key, i) = H(key + i D(key)) mod h, where i = [0, 1, ..., h-1].

The insertion and deletion function prototypes are given as follows:

```c
int HashInsert(int key, HashSlot hashTable[]);
int HashDelete(int key, HashSlot hashTable[]);
```

Both functions' return value is the number of key comparisons done during insertion and deletion respectively. Inserting a duplicate key will return -1. If the number of key comparisons is more than the table size, it implies that the table is full. For the deletion function, it will return -1 if the deleting key does not exist. The number of key comparisons in deletion cannot be more than table size.