COMP 132: Advanced Programming
Spring 2017

Problem Session 8

A hash table is a data structure to store data items in the form of (key,value). Hash table stores data in an array format. Each entry of the array corresponds to a unique key. All the values corresponding to data items with key K are stored as a linked-list under the entry of array holding that key. A sample of hash table is given in Figure 1. The given hash table supports keys in the range of 0 to 7. Data items whose keys fall in this range can be inserted to this hash-table. The following data items are used to construct the hash table shown in Figure 1:

(0,Al), (0,John), (2,Fred), (2,Zak), (2, Amy), (5,Cal), (7,Sam), (7,Ted)

The data items with the same key are inserted under the same entry of hash table and in the form of a linked-list.

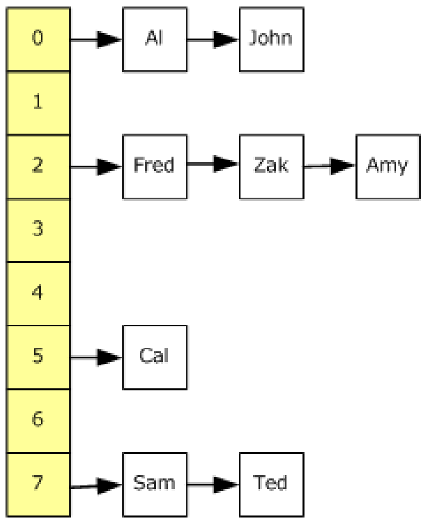


Figure 1. Sample hash table

The question consists of two parts. In the first part, you are required to implement a hash table whereas in the second part you edit the hash table using the user's inputs.

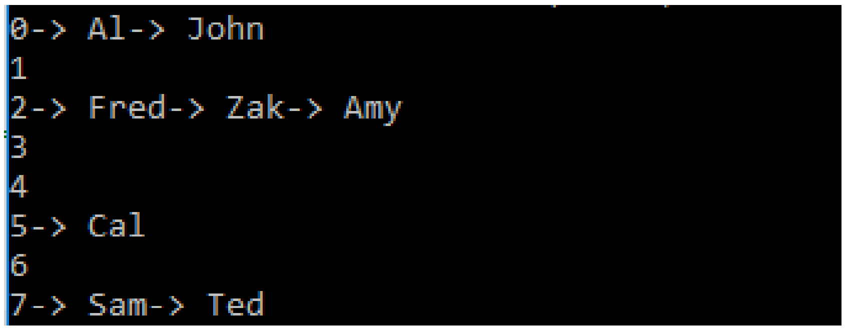
Create a user-defined library for the type definitions, constants, and the required function prototypes in the header file (HT.h). Put all the function implementations in the source code file (HT.c). Put the main function in another source code file (Source.c).

Part I: You are asked to implement a hash-table of a predefined size . The size is already hardcoded in the C project as `HASH_TABLE_SIZE`.

- Data items are key value pairs where key is an integer and value is a string. Each data item is a struct named **item** consisting of an integer named as **key**, a string named as **value** and a pointer to another data item named as **next**. The length of **value** does not exceed `VALUE_LENGTH` given in the c project.
- Define **dItem** as a struct of item.
- Define **dItemPtr** as a pointer to a struct of item.
- Each entry of hash table's array is a struct named **entry** consists of an integer named **key** and a dItemPtr as **ItemList**.
- Define **htEntry** as a struct of entry.

To construct the hash table, you need to implement the body of the following functions:

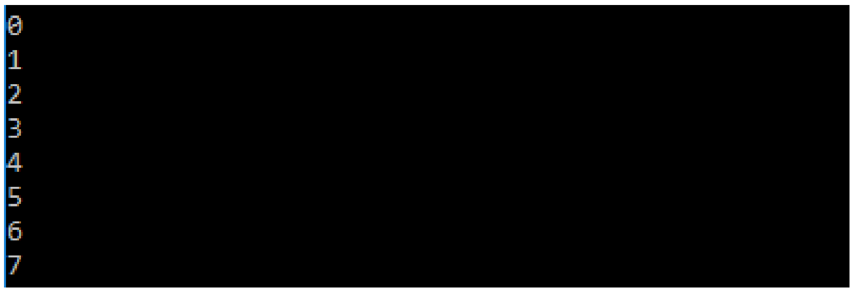
`void hDisplay(htEntry Htable[])`: Displays the hash table. For the hash table given in Figure 1 the function's output is visualized in Figure 2.



```
0-> Al-> John
1
2-> Fred-> Zak-> Amy
3
4
5-> Cal
6
7-> Sam-> Ted
```

Figure 2. Sample output for hDisplay

`void hInit(htEntry *Htable)`: This function initializes the hash table Htable. The ith entry of hash table is a struct of `htEntry` whose ItemList and key are set to NULL and i, respectively. Indices start from zero. If you call `hDisplay` on a hash table of size 7 which is initialized by `hInit`, you must see the output look like the Figure 3.



```
0
1
2
3
4
5
6
7
```

Figure 3. Sample output of hDisplay for a hash table initialized by hInit

`void  hInsert(htEntry *Htable, int key, char *value)`: `hInsert` receives a hash table Htable to which inserts a data item with the given key and value. Each new data item must be added at the end of the corresponding linked-list. You are required to allocate enough space for each new item.

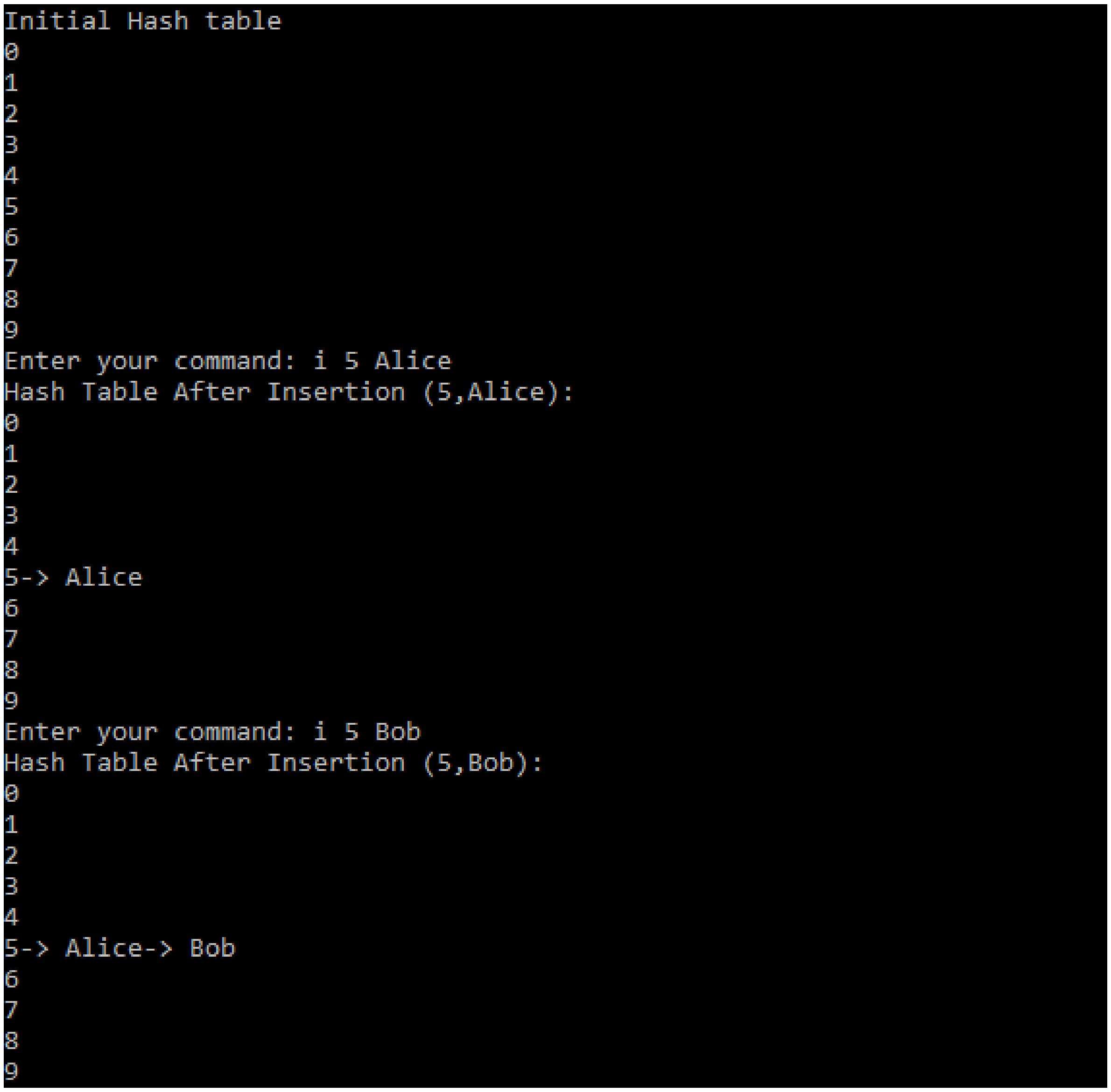
`void  hDelete(htEntry *Htable, int key, char *value)`: `hDelete` receives a hash table Htable and from which deletes the data item with the given key and value. You are required to free up the space of the deleted item.

Part II: Create a hash table of a predefined size (the size is hardcoded in the C project). Your program must be interacting with the user in an infinite loop to insert/delete to/from the hash table using the user's commands. You need to use your implemented functions of part I. Your program must dipsplay "I**nitial Hash table**" message at the first line of screen followed by the visualization of the initialized hash table. Then program asks user's commands by showing **"Enter your commands"** message.

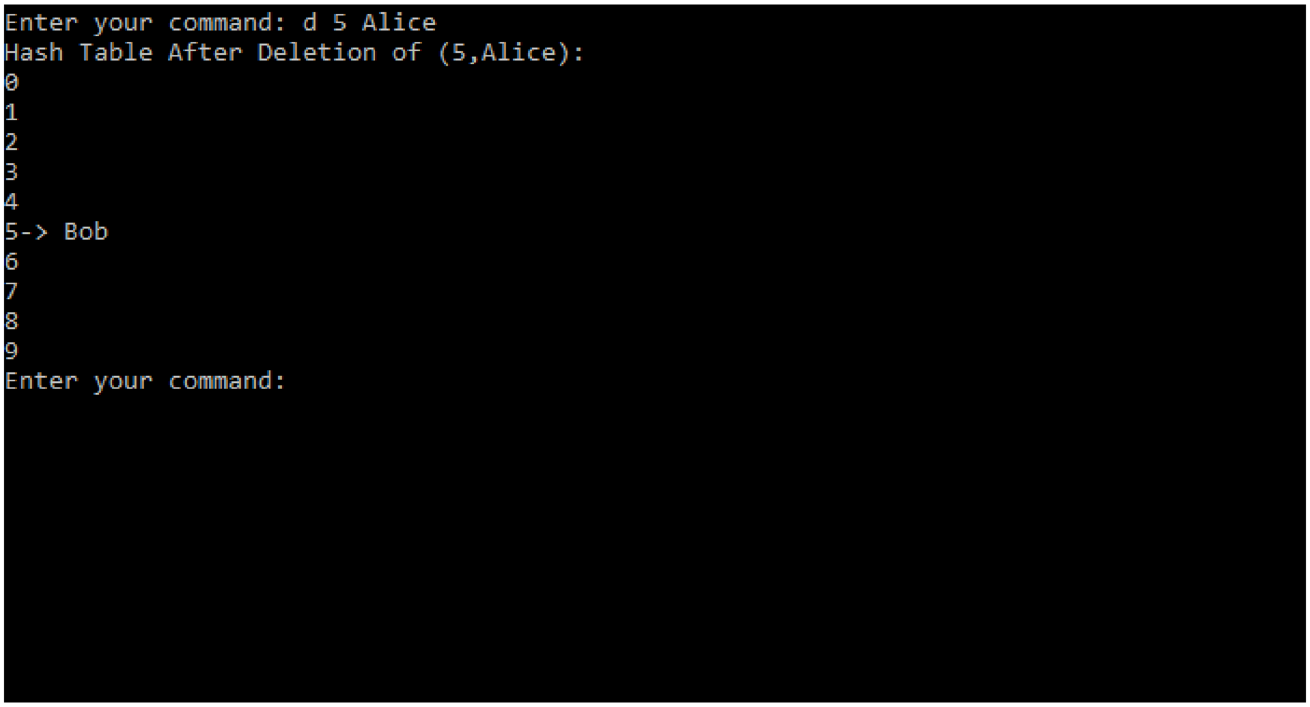
Three types of commands are defined:

- i<space> <key> <space> <value>: Your program must insert a data item with the given key and value. The message of "Hash Table After the Insertion of (key,value)" followed by the resulted hash table must be displayed on the screen.
- d<space> <key><space> <value>: Your program must delete the data item whose key and value equals to the key and value provided by the user. The message of "Hash Table After the Deletion of (key,value)" followed by the resulted hash table must be displayed on the screen.
- q: User may end the program by entering 'q'.

A sample of program execution is given in the Figure 4. You may assume all the characters are in lowercase. You may assume user always enters valid commands and inputs (no need to check the validity of commands or inputs).



```
Initial Hash table
0
1
2
3
4
5
6
7
8
9
Enter your command: i 5 Alice
Hash Table After Insertion (5,Alice):
0
1
2
3
4
5-> Alice
6
7
8
9
Enter your command: i 5 Bob
Hash Table After Insertion (5,Bob):
0
1
2
3
4
5-> Alice-> Bob
6
7
8
9
```



```
Enter your command: d 5 Alice
Hash Table After Deletion of (5,Alice):
0
1
2
3
4
5-> Bob
6
7
8
9
Enter your command:
```

Figure 4. Sample program execution