BLG335E, Analysis of Algorithms I, Fall 2016 Project Report 4

Name: Fatih BUDAK

Student ID: 150130006

Part A. Questions on Hash Tables (20 points)

1) Why do we use Hash Tables as a data structure in our problems? Please explain briefly. **(5 points)**

Hash tables are ideal to search a quickly. For example, if we have an array full of data. We could quickly access any elements of array because we knew its position that is stored in an array. Therefore, we use hash tables.

2) Consider a hash table consisting of M = 11 slots, and suppose nonnegative integer key values are hashed into the table using the hash function `h1()` :

```
int h1 (int key) {
int x = (key + 7) * (key + 7);
x = x / 16;
x = x + key;
x = x % 11;
return x;
}
```

Suppose that collisions are resolved by using linear probing. The integer key values listed below are to be inserted, in the order given. Show the home slot (the slot to which the key hashes, before any probing), the probe sequence (if any) for each key, and the final contents of the hash table after the following key values have been inserted in the given order: **(10+5 points)**

Key Value	Home Slot	Probe Sequence
43	1	
23	6	
1	5	
0	3	
15	1	1
31	0	
4	0	4
7	8	
11	9	
3	9	1

Final Hash Table:

Slot	0	1	2	3	4	5	6	7	8	9	10
Contents	31	43	15	0	4	1	23		7	11	3

Part B. Implementation and Report (80 points)

In this project, we are required to create the hash table. Therefore, we have created a class as called myHash including a string pointer for hash table, integer variables for the hash table size, collisions, and total entries and also functions which do insert, delete, and retrieve operations and constructor and destructor functions.

```
class myHash {

private:
    string *myHashTable;
    int totalEntries;

public:
    int _tableSize;
    int collision ;
    myHash(int tableSize);
    ~myHash();
    void insert(string s);
    int hashFunc(string s);
    bool retrieve(string s);
    bool remove(string s);
    string * get_myHashTable();
    void set_myHashTable(string, int );
};
```

- ***hashFunc*** function: This function provides to generate the index for words in words.txt file according to ascii codes of letters.
- ***insert*** function: This function provides to put in the words appropriate places according to index number that is generated by ***hashFunc*** function. If first appropriate slot is filled for any words, we use linear probing.
- ***remove*** function: If any word desiring to delete is avaiable in hash table, this function find and delete the words. Also it puts in ***'#'*** character in place of deleted word.
- ***retrieve*** function: This function searchs the word desiring to find. If it finds, it print out index of this word. Otherwise, for this word, it finds similar words and print out the screen if avaiable.
- ***myHash constructor*** function: Thanks to constructor function, we create our hash table by table size and integer variable of collision and total entiries is initialized to zero.
- ***myHash*** destructor function: It releases to memory being reserved for string of **myHashTable**.
- ***get_myHashTable*** function: it is only getter function. It returns string of **myHashTable**.
- ***set_ myHashTable*** function: it is only setter function. It sets elements of **myHashTable**.