Title: Hash Table

Author: Işık Özsoy

ID: 21703160

Section: 01

Assignment: 4

Description: This file contains the answers to question 2

Part 1:

HashTable class has 5 properties. tableItem is integer array which holds integer values in the hash table. tableState is array of States, defined as enum, which holds whether the item is EMPTY, OCCUPIED or DELETED. tableSize is the size of the table and size is the number of items in the list. strategy holds CollisionStrategy, which is defined as enum and hashing is done according to strategy.

In the HashTable class, there is one constructor, one destructor and there are 6 functions;

Constructor takes tableSize and CollisionStrategy as an input and initializes the hash table.

Destructor cleans up any dynamic memory used (tableItems, tableState).

hash function takes item as parameter and calculate the available index value for the item according to the CollisionStrategy of the HashTable and returns this value. For all CollisionStrategies firstly index = item mod tableSize is calculated. If the index is OCCUPIED, according to CollisionStrategy, the index is increased. For linear probing, it is increased by one. For quadratic probing it is increased by i^2 where i start from 1. For double hashing, there is a secondary hash function reverses the digits of the key. The result of the second hash function is added the index until we find an available location.

insert function takes integer item as an input and try to insert it into the table. If the insertion is successful, it returns true, otherwise, it returns false.

In **remove function**, integer item is taken as an input to be deleted, if the deletion is successful, it returns true, otherwise, false. It follows the same steps in the hashing until the item is found or EMPTY location is found.

search function takes integer item and numProbes as inputs. While searching for an item, it counts number of probes and returns it inside numProbes. During searching it follows the same procedure in the hash and remove function. If the search is successful, it returns true, otherwise, false.

Note: In hash, insert, remove and search functions, there might be infinite loops. In order to avoid them, the value of **i** should not be greater than or equal to tableSize. We need to probe a most tableSize times since algorithm starts probing the same indexes beyond tableSize**. Remeber: h(key) =(hash(key) + f(i))**

For example, assume that we have a table with size 11 and it includes 22. The index value of 22 will be 0. If the user searches for 33 by using double hashing, the index value will always be 0. **h(key) = hash(key) + i * reverse(33) = 0 + i * 33 → (3 * 11 * i) mod 11 ≡ 0**
Thus there will be an infinite loop, this is why we write a stopping condition.

display function displays the contents of the Hash Table.

analyze function analyzes the current hash table in terms of the average number of probes for successful and unsuccessful searches, and returns these two values in the variables passed by reference. For analyzing the performance for successful searches, the item values currently stored in the table can be used for searching. For analyzing the performance of unsuccessful searches, a search that starts at each array location (index) can be used, and count the number of probes needed to reach an empty location in the array for each search.

Part 2:

HOW TO RUN:
The user should follow this pattern while testing the program,
./Makefile filename tableSize CollisionStrategy

Example:
./hw4 "test.txt" 11 DOUBLE

→For CollisionStrategy user can enter whether LINEAR, QUADRATIC or DOUBLE, otherwise, program will stop working and print "Please enter a valid input."

An input file is created to test the program described above and the operations inside the test file are;

I 11
I 4
I 22
S 33
I 2
S 2
I 8
R 11
S 22
R 8
I 5

The corresponding output when we run our program for **Linear Probing** and for **table size 11** is;

```
[isik.ozsoy@dijkstra hw4]$ ./hw4 test.txt 11 LINEAR
I 11
11 inserted
I 4
4 inserted
I 22
22 inserted
S 33
33 not found after 3 probes
I 2
2 inserted
S 2
2 found after 1 probes
I 8
8 inserted
R 11
11 removed
S 22
22 found after 2 probes
R 8
8 removed
I 5
5 inserted
0:
1: 22
2: 2
3:
4: 4
5: 5
6:
7:
8:
9:
10:
Average number of successful probes is 1.25
Average number of unsuccessful probes is 1.90909
Table Size: 11
```

The corresponding output when we run our program for **Quadratic Probing** and for **table size 11** is;

```
[isik.ozsoy@dijkstra hw4]$ ./hw4 test.txt 11 QUADRATIC
I 11
11 inserted
I 4
4 inserted
I 22
22 inserted
S 33
33 not found after 4 probes
I 2
2 inserted
S 2
2 found after 1 probes
I 8
8 inserted
R 11
11 removed
S 22
22 found after 2 probes
R 8
8 removed
I 5
5 inserted
0:
1: 22
2: 2
3:
4: 4
5: 5
6:
7:
8:
9:
10:
Average number of successful probes is 1.25
Average number of unsuccessful probes is 2.18182
Table Size: 11
```

The corresponding output when we run our program for **Double Hashing** and for **table size 11** is;

```
[isik.ozsoy@dijkstra hw4]$ ./hw4 test.txt 11 DOUBLE
I 11
11 inserted
I 4
4 inserted
I 22
22 not inserted
S 33
33 not found after 11 probes
I 2
2 inserted
S 2
2 found after 1 probes
I 8
8 inserted
R 11
11 removed
S 22
22 not found after 11 probes
R 8
8 removed
I 5
5 inserted
0:
1:
2: 2
3:
4: 4
5: 5
6:
7:
8:
9:
10:
Average number of successful probes is 1
Average number of unsuccessful probes is -1
Table Size: 11
```

Part 3:

α= (current number of items) / tableSize

For linear probing the approximate average number of probes:

$$\frac{1}{2}\left[1+\frac{1}{1-\alpha}\right] \qquad \textit{for a successful search}$$

$$\frac{1}{2}\left[1+\frac{1}{(1-\alpha)^2}\right] \qquad \textit{for an unsuccessful search}$$

$$\left[\frac{1}{\alpha}(\log_e \frac{1}{1-\alpha})\right] = \frac{-\log_e(1-\alpha)}{\alpha}$$ *for a successful search*

$$\frac{1}{1-\alpha}$$ *for an unsuccessful search*

Table for Successful Searches

	Value of Analyze Function	Theoretical Value
Linear Probing	1.25	1.29
Quadratic Probing	1.25	1.24
Double Hashing	1	-

Table for Unsuccessful Searches

	Value of Analyze Function	Theoretical Value
Linear Probing	1.91	1.08
Quadratic Probing	2.18	1.57
Double Hashing	-1 (don't care)	-

All values are calculated basing on the previous example input above (given in part 2). Considering the theoretical values and the result of the analyze function in the previous example, it can be observed that they are no exactly the same, sometimes they are much more close, sometimes the difference is greater. For example, the values in the first table are close to each other, meanwhile there are more differences in the second table. However, those differences are expected. In order to get more similar results, we may increase the operation count and table size. In other words, with more appropriate test cases, the theoretical results and the results of the analyze function may become much closer to each other.