/**

* Author : Ahmet Faruk Ulutas

* ID: 21803717

* Section : 1

* Assignment : 4

*/

Question 2 – Part 1:

While making the hash table, I first determined the type of the table with CollisionStrategy in the constructor. The hash table consists of nodes named HashNode and the table keeps it as an array. The Insert method works if the Table is not full and the item has not been inserted. The Remove method works if the item has not been removed and is found. The display method returns the whole table and outputs each index in the desired format. The Analyze method creates two unique arrays named successfulItems and unsuccessfulItems. Then these arrays test numProbes for successful and unsuccessful and return the results with pass by reference. As a helper, I used the hash, f and reverse methods related to the hash as stated. In order not to repeat the codes, I gathered all the loops under a single loop in the FindAndDo method. This method has a for loop equal to its tableSize. Inside this method, it calculates the hash value at each loop step and checks the index. If it is an insert scenario and the item is not inserted, it adds the item to the first location that is Empty. If it is a Remove scenario and the item exists, it deletes the item when it is found. For the Search scenario, numProbes is incremented by 1 for each search. Returns true if the condition is successful in each of insert, remove and search . For the Analyze scenario, successfulAnalyze checks whether its status is Deleted when it finds the item, and numProbes increases by 1. The condition of the unsuccessful scenario is the opposite and the numProbes increase by one again. At the end of each loop, it is checked whether the reached location is empty, and if the reached location is empty, the loop is exited with a break. Thus, it is prevented from recalculating the hash and going to the wrong locations or looping more than necessary.

Question 2 – Part 2:

Content of the input.txt

I 3

I 14

I 26

I 25

I 31

I 21

I 11

I 8

I 10

I 10

R 8

R 334

R 8

S 21

S 31

S 3

S 14

S 26

S 25

S 11

S 10

S 1

S 2

S 36

S 4

S 5

S 6

S 7

S 8

S 9

S 32

S 33

Output from the driver function:

```
LINEAR TEST
3 inserted
14 inserted
26 inserted
25 inserted
31 inserted
21 inserted
11 inserted
8 inserted
10 inserted
10 not inserted
8 removed
334 not removed
8 not removed
21 found after 1 probes
31 found after 1 probes
3 found after 1 probes
14 found after 1 probes
26 found after 1 probes
25 found after 1 probes
11 found after 1 probes
10 found after 1 probes
1 not found after 1 probes
2 not found after 1 probes
36 not found after 1 probes
4 not found after 1 probes
5 not found after 1 probes
6 not found after 1 probes
7 not found after 1 probes
8 not found after 2 probes
9 not found after 1 probes
32 not found after 1 probes
33 not found after 1 probes
```

```
DISPLAY
0: 31
1:
2:
3: 3
4:
5:
6:
7:
8:
9:
10: 10
11: 11
12:
13:
14: 14
15:
16:
17:
18:
19:
20:
21: 21
22:
23:
24:
25: 25
26: 26
27:
28:
29:
30:

ANALYZE
Successful Probes: 8
Unsuccessful Probes: 31
```

```
QUADRATIC TEST
3 inserted
14 inserted
26 inserted
25 inserted
31 inserted
21 inserted
11 inserted
8 inserted
10 inserted
10 not inserted
8 removed
334 not removed
8 not removed
21 found after 1 probes
31 found after 1 probes
3 found after 1 probes
14 found after 1 probes
26 found after 1 probes
25 found after 1 probes
11 found after 1 probes
10 found after 1 probes
1 not found after 1 probes
2 not found after 1 probes
36 not found after 1 probes
4 not found after 1 probes
5 not found after 1 probes
6 not found after 1 probes
7 not found after 1 probes
8 not found after 2 probes
9 not found after 1 probes
32 not found after 1 probes
33 not found after 1 probes
```

```
DISPLAY
0: 31
1:
2:
3: 3
4:
5:
6:
7:
8:
9:
10: 10
11: 11
12:
13:
14: 14
15:
16:
17:
18:
19:
20:
21: 21
22:
23:
24:
25: 25
26: 26
27:
28:
29:
30:

ANALYZE
Successful Probes: 8
Unsuccessful Probes: 31
```

```
DOUBLE TEST
3 inserted
14 inserted
26 inserted
25 inserted
31 inserted
21 inserted
11 inserted
8 inserted
10 inserted
10 not inserted
8 removed
334 not removed
8 not removed
21 found after 1 probes
31 found after 1 probes
3 found after 1 probes
14 found after 1 probes
26 found after 1 probes
25 found after 1 probes
11 found after 1 probes
10 found after 1 probes
1 not found after 1 probes
2 not found after 1 probes
36 not found after 1 probes
4 not found after 1 probes
5 not found after 1 probes
6 not found after 1 probes
7 not found after 1 probes
8 not found after 2 probes
9 not found after 1 probes
32 not found after 1 probes
33 not found after 1 probes
```

```
DISPLAY
0: 31
1:
2:
3: 3
4:
5:
6:
7:
8:
9:
10: 10
11: 11
12:
13:
14: 14
15:
16:
17:
18:
19:
20:
21: 21
22:
23:
24:
25: 25
26: 26
27:
28:
29:
30:

ANALYZE
Successful Probes: 8
Unsuccessful Probes: -1
```

In the example table size was 31 which is a prime number.

Question 2 – Part 3:

# Linear Probing -- Analysis

- **Linear Probing –** approximate average number of comparisons (probes) that a search requires :

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

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

In my implementation, there are 8 items in the hash table and size is 31 so load factor is 8/31.

Theoratical result for successful search is 1.174, empirical result is 0.81.

Theoratical result for unsuccessful search is undefined since load factor is 1 and 1-1 makes 1/0, empirical result is 1.

## Quadratic Probing & Double Hashing -- Analysis

- The approximate average number of comparisons (probes) that a search requires is given as follows:

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

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

In my implementation, there are 8 items in the hash table and size is 31 so load factor is 8/31.

Theoratical result for unsuccessful search is undefined since load factor is 1 and 1-1 makes 1/0

Theoratical result for unsuccessful search is undefined since load factor is 1 and 1-1 makes 1/0, empirical result is –1 as requested.

In general, for successful and unsuccessful scans, I thought it was lower than theoretical values in terms of performance because the number of my hash table was small. If we give the very large table size and the number of data, we can approach the theoretical value.