Abdul Basit Anees
21600659
CS 202-3

Homework 4

Question 2

Part 1:

My **HashTable** class is implemented by using the following private variables.

private:

int table;*

LocationStatus status;*

int hashTableSize;

CollisionStrategy strategy;

int curSize;

Where I defined enum LocationStatus { OCCUPIED, EMPTY, DELETED } for knowing the status of a location in the hashTable. The variables table and status are used as the arrays for hashTable storage. Note that they could also have been packed into a struct but the current implementation keeps them separate.

While probing, the *stopping conditions* are designed such that a search can be stopped and returned unsuccessful if the search reaches an EMPTY location. In order to avoid *infinite loops*, the total number of probes is limited to hashTableSize, since the indices repeat after that. This can be proved mathematically for all strategies as follows.

Abdul Basit Anees
21600659
CS 202-3

Proof:

Initial starting index is,

index = key % tableSize

for ith probing iteration,

For linear probing:

newIndex = (index + i) % tableSize

after tableSize probes (i = tableSize + k),

newIndex = (index + tableSize + k) % tableSize = (index + k) % tableSize

which shows that index repeats after tableSize probes.

For quadratic Probing:

newIndex = (index + i^2) % tableSize

after tableSize probes (i = tableSize + k),

newIndex = (index + $tableSize^2$ + k^2 + ((2k)* tableSize)) % tableSize = (index + k^2) % tableSize

which shows that index repeats after tableSize probes.

For double hashing:

newIndex = (index + (i * hash2(item))) % tableSize

after tableSize probes (i = tableSize + k),

newIndex = (index + ((tableSize + k) * hash2(item))) % tableSize

newIndex = (index + ((tableSize * hash2(item)) + (k * hash2(item)))) % tableSize

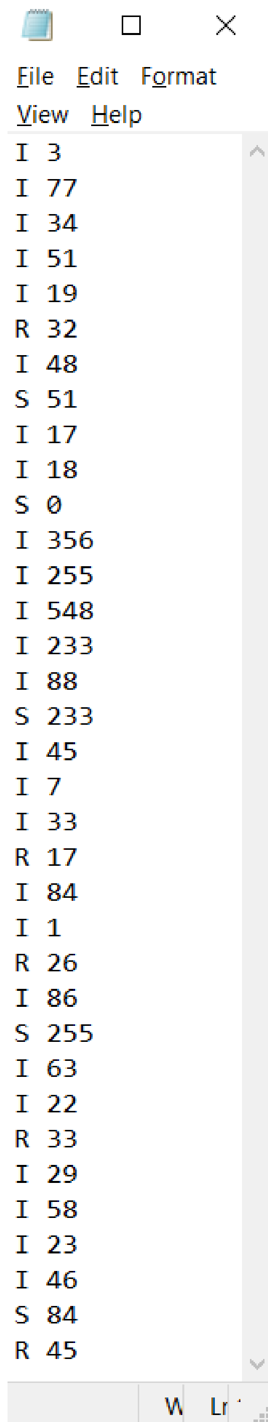
newIndex = (index + (k * hash2(item))) % tableSize

which shows that index repeats after tableSize probes.

Abdul Basit Anees
21600659
CS 202-3

Part 2:

The table size used was 29. The data used for the driver function and the corresponding outputs are as follows:



File Edit Format View Help

```
I 3
I 77
I 34
I 51
I 19
R 32
I 48
S 51
I 17
I 18
S 0
I 356
I 255
I 548
I 233
I 88
S 233
I 45
I 7
I 33
R 17
I 84
I 1
R 26
I 86
S 255
I 63
I 22
R 33
I 29
I 58
I 23
I 46
S 84
R 45
```

Abdul Basit Anees
21600659
CS 202-3

Using Linear probing:

```
3 inserted
77 inserted
34 inserted
51 inserted
19 inserted
32 not removed
48 inserted
51 found after 1 probes
17 inserted
18 inserted
0 not found after 1 probes
356 inserted
255 inserted
548 inserted
233 inserted
88 inserted
233 found after 1 probes
45 inserted
7 inserted
33 inserted
17 removed
84 inserted
1 inserted
26 not removed
86 inserted
255 found after 1 probes
63 inserted
22 inserted
33 removed
29 inserted
58 inserted
23 inserted
46 inserted
84 found after 2 probes
45 removed
```

```
0: 29
1: 233
2: 88
3: 3
4: 58
5: 34
6: 1
7: 7
8: 356
9: 63
10:
11:
12:
13:
14:
15:
16:
17: 46
18: 18
19: 77
20: 19
21: 48
22: 51
23: 255
24: 22
25: 23
26: 548
27: 84
28: 86
```

Abdul Basit Anees
21600659
CS 202-3

Using Quadratic probing:

```
3 inserted
77 inserted
34 inserted
51 inserted
19 inserted
32 not removed
48 inserted
51 found after 1 probes
17 inserted
18 inserted
0 not found after 1 probes
356 inserted
255 inserted
548 inserted
233 inserted
88 inserted
233 found after 1 probes
45 inserted
7 inserted
33 inserted
17 removed
84 inserted
1 inserted
26 not removed
86 inserted
255 found after 2 probes
63 inserted
22 inserted
33 removed
29 inserted
58 inserted
23 inserted
46 inserted
84 found after 2 probes
45 removed
```

```
0: 29
1: 233
2: 88
3: 3
4: 58
5: 34
6: 63
7: 7
8: 356
9: 22
10: 1
11:
12:
13:
14: 23
15:
16:
17: 46
18: 18
19: 77
20: 19
21:
22: 51
23: 48
24: 255
25:
26: 548
27: 84
28: 86
```

Abdul Basit Anees
21600659
CS 202-3

Using Double Hashing:

```
3 inserted
77 inserted
34 inserted
51 inserted
19 inserted
32 not removed
48 inserted
51 found after 1 probes
17 inserted
18 inserted
0 not found after 1 probes
356 inserted
255 inserted
548 inserted
233 inserted
88 inserted
233 found after 1 probes
45 inserted
7 inserted
33 inserted
17 removed
84 inserted
1 inserted
26 not removed
86 inserted
255 found after 2 probes
63 inserted
22 inserted
33 removed
29 inserted
58 inserted
23 inserted
46 inserted
84 found after 3 probes
45 removed
```

```
0: 29
1: 233
2: 88
3: 3
4:
5: 34
6: 84
7: 7
8: 356
9: 1
10:
11: 63
12:
13:
14:
15: 22
16: 48
17: 46
18: 18
19: 77
20:
21: 23
22: 51
23: 19
24: 255
25:
26: 548
27: 58
28: 86
```

Abdul Basit Anees
21600659
CS 202-3

Part 3:

We obtain the following results after using different collision resolution schemes like linear and quadratic probing and double hashing.

Using Linear probing:

```
Load factor: 0.758621
Average empirical Successful probes: 1.51724
Average empirical UnSuccessful probes: 10.5172
Average theoretical Successful probes: 2.57143
Average theoretical UnSuccessful probes: 9.08163
```

Where the theoretical values are obtained using,

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

Using Quadratic probing:

```
Load factor: 0.758621
Average empirical Successful probes: 1.55172
Average empirical UnSuccessful probes: 4.65517
Average theoretical Successful probes: 1.87364
Average theoretical UnSuccessful probes: 4.14286
```

Using Double Hashing:

```
Load factor: 0.758621
Average empirical Successful probes: 1.7931
Average empirical UnSuccessful probes: -1
Average theoretical Successful probes: 1.87364
Average theoretical UnSuccessful probes: 4.14286
```

Where the theoretical values are obtained using,

$$\frac{-\log_e(1-\alpha)}{\alpha} \quad \textit{for a successful search} \quad , \quad \frac{1}{1-\alpha} \quad \textit{for an unsuccessful search}$$

It can be observed that the theoretical and empirical values are very close to each other. Moreover, we also observe that linear probing performs the worst mainly due to the clustering problem. Quadratic probing and double hashing perform fairly better.

Conclusion:

This homework was a good exercise to understand and implement hashTables and the understanding different collision resolution strategies and their comparison and analysis.