Homework 4
Burak Öztürk
21901841
Section 01
Assignment 4

Part 1:

Linear probing: Since h(key, i) = hash(key) + i, h(key, i) = h(key, 0) probing is finished and whole table iterated once.

Quadratic probing: For quadratic, I assumed tableSize is always prime. In that case, quadratic probing is guaranteed to visit at least half of the table as a loop.[1] That means iterating i in range [0, tableSize) guarantees to visit all possible positions even if all the table can be visited.

Double probing: I iterated i in range [0, tableSize). Because if i = tableSize, (hash(key) + i * reverse(key)) % tableSize becomes hash(key) and completes the loop.

Part 2:

Input file:

```
I 13
I 26
I 39
I 52
I 14
I 45
R 26
S 13
S 65
S 26
S 14
I 89
R 89
S 52
I 13
S 45
I 65
R 26
```

Output (Linear probing):

```
13 inserted
26 inserted
39 inserted
52 inserted
14 inserted
45 inserted
26 removed
13 found after 1 probes
65 not found after 6 probes
26 not found after 6 probes
14 found after 4 probes
89 inserted
89 removed
52 found after 4 probes
13 not inserted
45 found after 1 probes
65 inserted
26 not removed
```

```
Collision strategy: Linear
Table size: 13
0: 13
1:
2: 39
3: 52
4: 14
5: 65
6: 45
7:
8:
9:
10:
11:
12:

Successful probes:   19
Unsuccessful probes: 42
Process finished with exit code 0
```

[1] https://research.cs.vt.edu/AVresearch/hashing/quadratic.php

Output (Quadratic probing):

```
13 inserted
26 inserted
39 inserted
52 inserted
14 inserted
45 inserted
26 removed
13 found after 1 probes
65 not found after 5 probes
26 not found after 5 probes
14 found after 2 probes
89 inserted
89 removed
52 found after 4 probes
13 not inserted
45 found after 1 probes
65 inserted
26 not removed
```

```
Collision strategy: Quadratic
Table size: 13
0: 13
1:
2: 14
3: 65
4: 39
5:
6: 45
7:
8:
9: 52
10:
11:
12:

Successful probes:   16
Unsuccessful probes: 30
Process finished with exit code 0
```

Output (Double probing):

```
13 inserted
26 inserted
39 inserted
52 inserted
14 inserted
45 inserted
26 removed
13 found after 1 probes
65 not found after 2 probes
26 not found after 3 probes
14 found after 1 probes
89 inserted
89 removed
52 found after 2 probes
13 not inserted
45 found after 1 probes
65 inserted
26 not removed
```

```
Collision strategy: Double
Table size: 13
0: 13
1: 14
2: 39
3:
4: 65
5:
6: 45
7:
8:
9:
10:
11:
12: 52

Successful probes:   9
Unsuccessful probes: -1
Process finished with exit code 0
```

Question 3:

Firstly let,
Average successful probes = Successful probes / current items
Average unsuccessful probes = Unsuccessful probes / tableSize
α = Current items / tableSize = 6 / 13 ~= 0.46154

Linear probing:

Theoretical average successful probes = 0.5 * [1 + 1 / (1 - α)] ~= 1.42857
Average successful probes = 19 / 6 ~= 3.166

Theoretical average unsuccessful probes = 0.5 * [1 + 1 / (1 - α^2)] ~= 2.22449
Average unsuccessful probes = 42 / 13 ~= 3.23077

Theoretical successful / unsuccessful ratio = 1.42857 / 2.22449 ~= 0.6422
Average successful / unsuccessful ratio = 3.166 / 3.23077 ~= 0.98016

Quadratic probing:

Theoretical average successful probes = -ln(1 - α) / α ~= 1.34125
Average successful probes = 16 / 6 ~= 2.66

Theoretical average unsuccessful probes = 1 / (1 - α) ~= 1.85714
Average unsuccessful probes = 30 / 13 ~= 2.3077

Theoretical successful / unsuccessful ratio = 1.34125 / 1.85714 ~= 0.72221
Average successful / unsuccessful ratio = 2.66 / 2.3077 ~= 1.15266

Double probing:

Average successful probes = 9 / 6 ~= 1.5

For the first two techniques, average and experimental values for successful searches are far apart by a factor of nearly 2 but unsuccessful searches are closer to each other.

This is because unsuccessful searches are executed for every possible index but successful searches are only executed for current array items and it turns out my example instruction set is not close to the average set. Hence the difference. Unsuccessful searches are effected by current array too since they are based on the first empty bucket but the effect is approximately halved since the array is half full.

The experimental values' success/not success ratios are much better than theoretical values. I don't know what causes this. It might be because of some mistake I did in the coding making algorithm worse but I am not sure.