Title: Implementation and Analysis of Different Probing Techniques in Hash Tables

Objective:

The objective of this lab is to implement different probing methods in hash tables, namely Linear Probing, Quadratic Probing, Double Hashing, and Open Chaining. The lab aims to analyze the performance and behavior of these techniques in terms of insertion, deletion, search operations, and their efficiency in handling collisions.

Introduction:

Hash tables are a crucial data structure used in various applications for efficient data retrieval. The efficiency of a hash table largely depends on the hashing function and the collision resolution technique used. This lab focuses on implementing and comparing four different probing methods: Linear Probing, Quadratic Probing, Double Hashing, and Open Chaining.

Methodology:

1. **Implementation:**
    - **Linear Probing:** Involves resolving collisions by linearly searching for the next available slot.
    - **Quadratic Probing:** Similar to Linear Probing but the interval between probes is increased quadratically.
    - **Double Hashing:** Uses a secondary hash function to calculate the step size for probing.
    - **Open Chaining:** Utilizes linked lists to manage collisions, allowing more than one element to be stored at each table index.
2. **Operations:** Each probing technique supports the following operations:

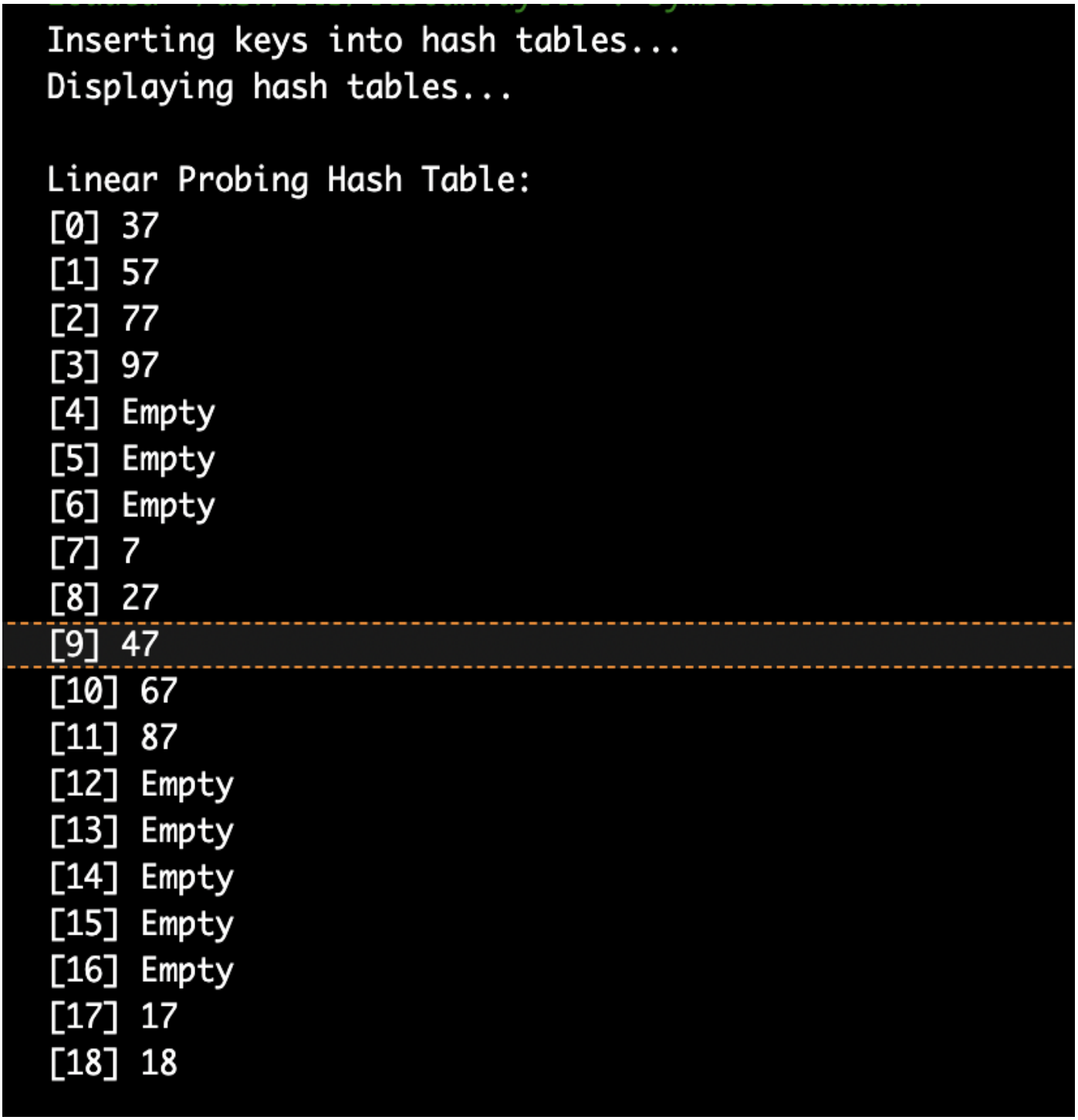
- **Insert:** Adds a new key to the hash table.
- **Find:** Searches for a key in the hash table.
- **Remove:** Deletes a key from the hash table.
- **Display:** Outputs the current state of the hash table.

3. **Load Factor Calculation:**
    - Calculated as the ratio of the number of keys inserted to the total size of the hash table.

Results and Discussion:

1. **Efficiency in Collision Handling:**
    - **Linear Probing:** Simple but suffers from clustering.
    - **Quadratic Probing:** Reduces clustering compared to Linear Probing but can still suffer from secondary clustering.
    - **Double Hashing:** Offers better performance by minimizing clustering using a secondary hash function.
    - **Open Chaining:** Efficiently handles collisions by using linked lists, but can lead to uneven distribution of data.
2. **Load Factor Analysis:**
    - The calculated load factor provides insight into how full the hash table is, which is crucial for performance considerations.
3. **Performance:**
    - The performance of each method varies based on the number of collisions and the load factor. Open Chaining generally provides better performance in high-collision scenarios.

Output:


Inserting keys into hash tables...
Displaying hash tables...
Linear Probing Hash Table:
[0] 37
[1] 57
[2] 77
[3] 97
[4] Empty
[5] Empty
[6] Empty
[7] 7
[8] 27
[9] 47
[10] 67
[11] 87
[12] Empty
[13] Empty
[14] Empty
[15] Empty
[16] Empty
[17] 17
[18] 18

```
Open Chaining Hash Table:
[0] 57 -> nullptr
[1] 77 -> nullptr
[2] 97 -> nullptr
[3] nullptr
[4] nullptr
[5] nullptr
[6] nullptr
[7] 7 -> nullptr
[8] 27 -> nullptr
[9] 47 -> nullptr
[10] 67 -> nullptr
[11] 87 -> nullptr
[12] nullptr
[13] nullptr
[14] nullptr
[15] nullptr
[16] nullptr
[17] 17 -> nullptr
[18] 18 -> 37 -> nullptr
```

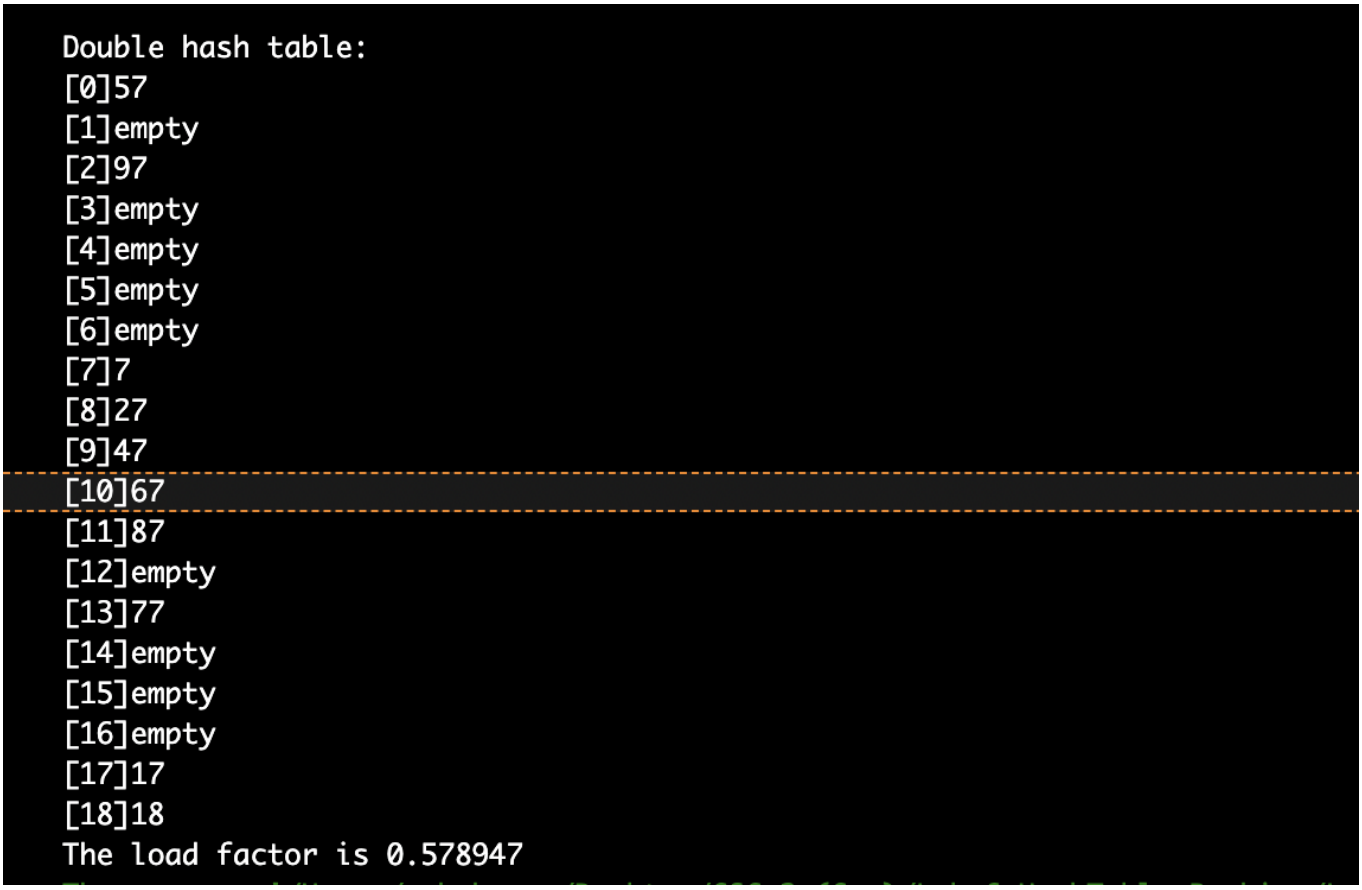
```
quadratic hash table
[0]57
[1]18
[2]37
[3]empty
[4]17
[5]47
[6]67
[7]27
[8]87
[9]empty
[10]77
[11]7
[12]empty
[13]97
[14]empty
[15]empty
[16]empty
[17]empty
[18]empty
```

```
Double hash table:
[0]57
[1]37
[2]97
[3]empty
[4]empty
[5]empty
[6]empty
[7]7
[8]27
[9]47
[10]67
[11]87
[12]empty
[13]77
[14]empty
[15]empty
[16]empty
[17]17
[18]18
Searching for key 47 in hash tables:
Linear Probing: Found
Open Chaining: Found
Quadratic Probing: Found
Double Hashing: Found
Removing key 37 from hash tables:
Linear Probing: Removed
Open Chaining: Removed
Quadratic Probing: Removed
Double Hashing: Removed
Displaying updated hash tables...
```

```
Linear Probing Hash Table:
[0] Empty
[1] 57
[2] 77
[3] 97
[4] Empty
[5] Empty
[6] Empty
[7] 7
[8] 27
[9] 47
[10] 67
[11] 87
[12] Empty
[13] Empty
[14] Empty
[15] Empty
[16] Empty
[17] 17
[18] 18
```

```
Open Chaining Hash Table:
[0] 57 -> nullptr
[1] 77 -> nullptr
[2] 97 -> nullptr
[3] nullptr
[4] nullptr
[5] nullptr
[6] nullptr
[7] 7 -> nullptr
[8] 27 -> nullptr
[9] 47 -> nullptr
[10] 67 -> nullptr
[11] 87 -> nullptr
[12] nullptr
[13] nullptr
[14] nullptr
[15] nullptr
[16] nullptr
[17] 17 -> nullptr
[18] 18 -> nullptr
```

```
quadratic hash table
[0]57
[1]18
[2]empty
[3]empty
[4]17
[5]47
[6]67
[7]27
[8]87
[9]empty
[10]77
[11]7
[12]empty
[13]97
[14]empty
[15]empty
[16]empty
[17]empty
[18]empty
```



Conclusion:

Each probing method has its strengths and weaknesses. Linear and Quadratic Probing are easy to implement but can suffer from clustering issues, affecting performance. Double Hashing shows improved performance by reducing clustering. Open Chaining, though potentially using more memory, provides an efficient solution to handle collisions without much impact on search times.

Further investigation can include a deeper analysis of the impact of load factors on performance and a comparison of the time complexity of each method under different scenarios.

Extra Observations:

- **Rehashing based on Load Factor:** Implementing rehashing when the load factor reaches a certain threshold could further optimize the performance, especially for Linear and Quadratic Probing.
- **Clustering Study:** A detailed study on the impact of clustering in Linear and Quadratic Probing could provide more insights into their efficiency in practical scenarios.