K. J. Somaiya College of Engineering, Mumbai-77
(A constituent College of Somaiya Vidyavihar University)

Batch: C1	**Roll. No.: 16010122257**
Experiment:8	
Grade: AA / AB / BB / BC / CC / CD /DD	

Title: Implementation of hashing concept

Objective: To understand various hashing methods

Expected Outcome of Experiment:

CO	Outcome
CO4	Demonstrate sorting and searching methods.

Websites/books referred:

1.Ma'am's classroom notes

2. *https://www.geeksforgeeks.org/hashing-data-structure/*

Abstract: -

(Define Hashing ,hash function, list collision handling methods)

Hashing is the process of transforming any given key or a string of characters into another value. This is usually represented by a shorter, fixed-length value or key that represents and makes it easier to find or employ the original string. A hash function is any function that can be used to map data of arbitrary size to fixed-size values. The

values returned by a hash function are called hash values, hash codes, digests, or

simply hashes. The values are usually used to index a fixed-size table called a hash

table.What are the methods for collision handling in hashing?

Collision Resolution Techniques

1) Linear Probing.

2) Quadratic Probing.

3) Double Hashing.

Example:

20, 33, 65, 23, 11, 32, 78, 64, 3, 87, 10, 7

Linear - 20,11,32,33,64,65,3,87,78,10,7,0,0,0,0,0,

Quadratic - 20,11,32,33,64,65,0,87,78,7,10,0,0,3,0,0,

Chaining - 20,10 ,0 ,0, 33,23,3 64 65 0 87,7, 78 ,0 ,0, 0, 0, 0, 0, 0, 0, 0

Code and output screenshots:

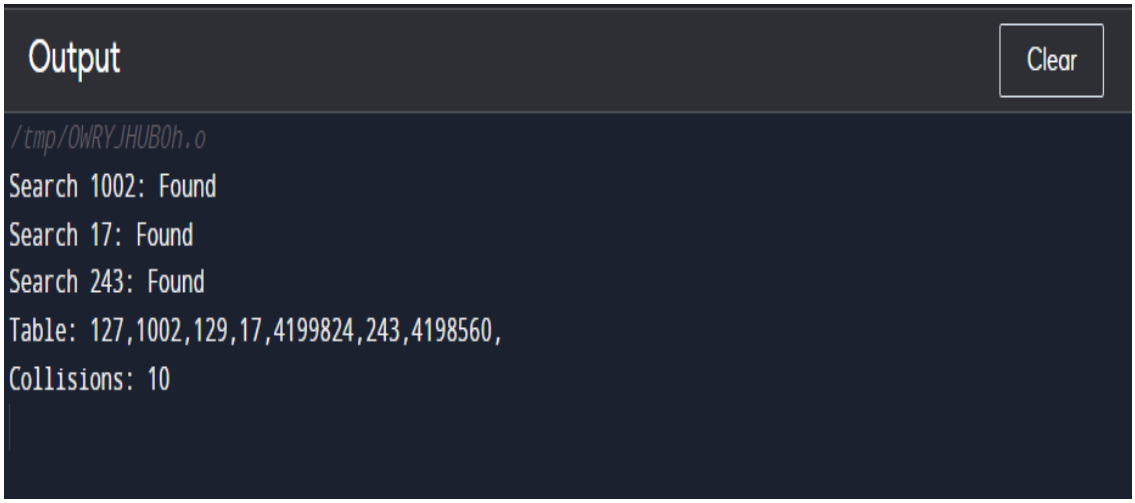
Quadratic probing:

```
#include <iostream>


const int len = 7;


class HashTable {

private:

   int arr[len];

   int collision = 0;


   int hash(int key) {

      return key % len;

   }


public:

   void print() {

      for (int i = 0; i < len; i++) {

         std::cout << arr[i] << ",";

      }

   }


   void add(int key) {
```

```
    int index = hash(key);
    int i = 1;

    while (arr[index] != 0) {
        collision++;
        index = (index + i * i) % len;
        i++;
    }

    arr[index] = key;
}

bool search(int key) {
    int index = hash(key);
    int i = 1;

    while (arr[index] != 0) {
        if (arr[index] == key) {
            return true;
        }
        collision++;
        index = (index + i * i) % len;
        i++;
    }
```

```
        return false;
    }

    int getCollisionCount() {
        return collision;
    }
};

int main() {
    HashTable table;
    table.add(1002);
    table.add(17);
    table.add(243);
    table.add(129);
    table.add(127);

    std::cout << "Search 1002: " << (table.search(1002) ? "Found" : "Not Found") <<
std::endl;
    std::cout << "Search 17: " << (table.search(17) ? "Found" : "Not Found") <<
std::endl;
    std::cout << "Search 243: " << (table.search(243) ? "Found" : "Not Found") <<
std::endl;

    std::cout << "Table: ";
```

```
    table.print();

    std::cout << std::endl;

    std::cout << "Collisions: " << table.getCollisionCount() << std::endl;


    return 0;
}
```



Conclusion: -

Thus, given experiment was successfully completed and implemented. In this experiment we learnt about hashing and algorithms related to hashing and also coded for the same(my application was Quadratic probing) in C++ language.

Post lab questions-

a. **Compare and contrast various collision handling methods.**
There are two types of collision resolution techniques.

- Separate chaining (open hashing)
- Open addressing (closed hashing)
- Separate chaining: This method involves making a linked list out of the slot where the collision happened, then adding the new key to the list. Separate chaining is the term used to describe how this connected list of

slots resembles a chain. It is more frequently utilized when we are unsure of the number of keys to add or remove.

Time complexity

Its worst-case complexity for searching is o(n).
Its worst-case complexity for deletion is o(n).
Advantages of separate chaining

It is easy to implement.
The hash table never fills full, so we can add more elements to the chain.
It is less sensitive to the function of the hashing.
Disadvantages of separate chaining

In this, the cache performance of chaining is not good.
Memory wastage is too much in this method.
It requires more space for element links.

- Open addressing: To prevent collisions in the hashing table open, addressing is employed as a collision-resolution technique. No key is kept anywhere else besides the hash table. As a result, the hash table's size is never equal to or less than the number of keys. Additionally known as closed hashing.

The following techniques are used in open addressing:

- Linear probing
- Quadratic probing
- Double hashing
- Linear probing: This involves doing a linear probe for the following slot when a collision occurs and continuing to do so until an empty slot is discovered.

The worst time to search for an element in linear probing is O. The cache performs best with linear probing, but clustering is a concern. This method's key benefit is that it is simple to calculate.

Disadvantages of linear probing:

The main problem is clustering.
It takes too much time to find an empty slot.

- Quadratic probing: When a collision happens in this, we probe for the i2-nd slot in the ith iteration, continuing to do so until an empty slot is discovered. In comparison to linear probing, quadratic probing has a worse cache performance. Additionally, clustering is less of a concern with quadratic probing.

- Double hashing: In this, you employ a different hashing algorithm, and in the ith iteration, you look for (i * hash 2(x)). The determination of two hash functions requires more time. Although there is no clustering issue, the performance of the cache is relatively poor when using double probing.

SOURCE:
https://www.geeksforgeeks.org/collision-resolution-techniques/

b. Store the given numbers in bucket of size 16, resolve the collisions if any with
   a. Linear probing
   b. Quadratic hashing
   c. Chaining

20, 33, 65, 23, 11, 32, 78, 64, 3, 87, 10, 7

Linear - 20,11,32,33,64,65,3,87,78,10,7,0,0,0,0,0,

Quadratic - 20,11,32,33,64,65,0,87,78,7,10,0,0,3,0,0,

Chaining - 20,10 ,0 ,0, 33,23,3 64 65 0 87,7, 78 ,0 ,0, 0, 0, 0, 0, 0, 0, 0