PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester : I**

Name of the student: Om Jitendra Khalane **PRN :** 124B1B040

Department: Computer Engineering **Division :** A

Course Name : Data Structures and Laboratory

Course Code: BCE23PC02

Completion Date : 12/08/2025

Problem Statement:

Write a C++ Program to insert elements in Hash Table using Separate Chaining.

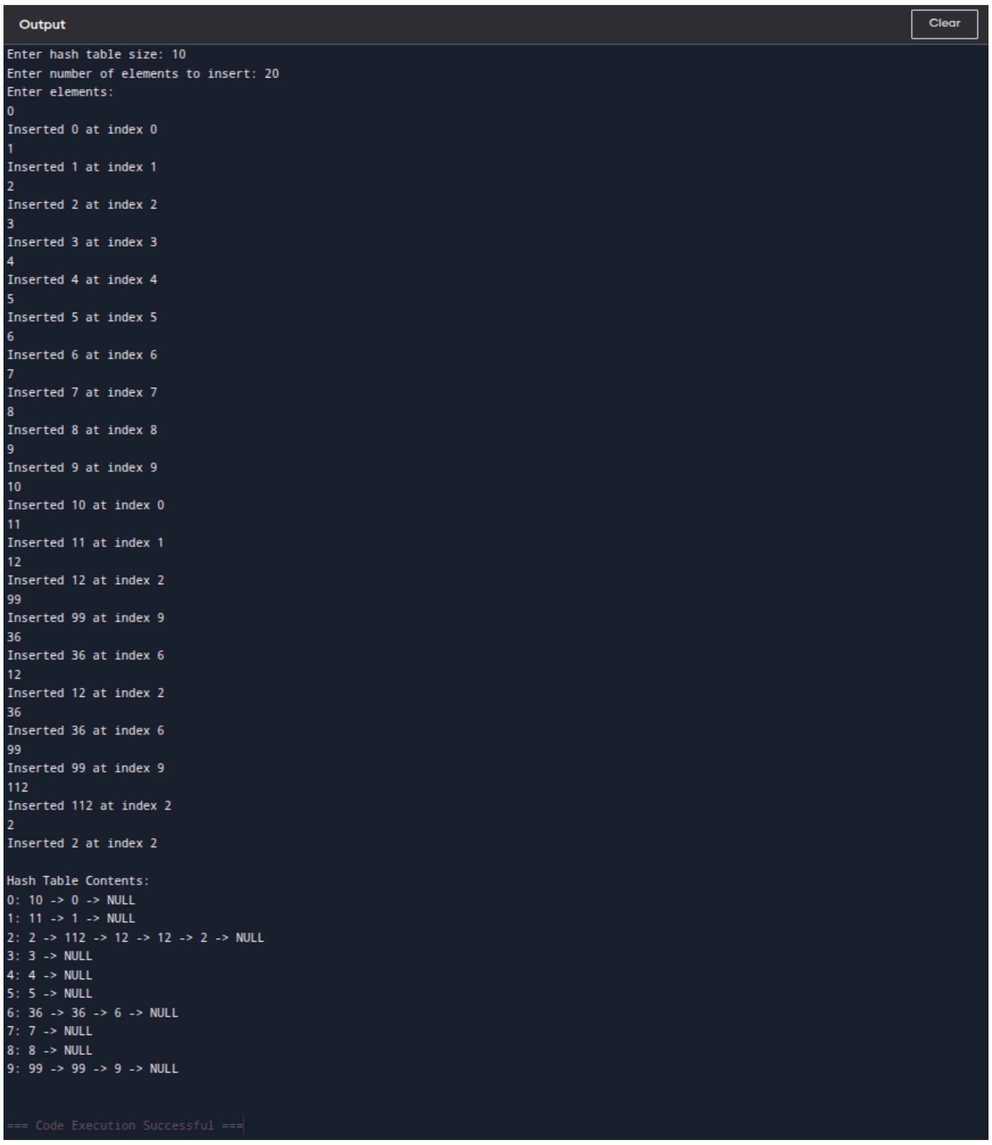
Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/assignment10.cpp

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
    int key;
    Node *next;

    Node(int k)
    {
        key = k;
```

```cpp
            next = nullptr;

        }

    };


    class HashTable

    {

        int tableSize;

        vector<Node *> table;


    public:

        HashTable(int size)

        {

            tableSize = size;

            table.resize(tableSize, nullptr);

        }


        int hashFunction(int key)

        {

            return key % tableSize;

        }


        void insert(int key)

        {

            int index = hashFunction(key);

            Node *newNode = new Node(key);


            newNode->next = table[index];

            table[index] = newNode;


            cout << "Inserted " << key << " at index " << index << endl;

        }
    void display()

    {

        cout << "\nHash Table Contents:\n";

        for (int i = 0; i < tableSize; i++)

        {

            cout << i << ": ";

            Node *curr = table[i];

            while (curr)

            {

                cout << curr->key << " -> ";

                curr = curr->next;

            }

            cout << "NULL\n";

        }

    }

};


int main()

{

    int size;

    cout << "Enter hash table size: ";

    cin >> size;


    HashTable ht(size);


    int n;

    cout << "Enter number of elements to insert: ";

    cin >> n;


    cout << "Enter elements:\n";

    for (int i = 0; i < n; i++)

    {
        int key;

        cin >> key;

        ht.insert(key);

    }


    ht.display();


    return 0;

}
```

Output :



```
Enter hash table size: 10
Enter number of elements to insert: 20
Enter elements:
0
Inserted 0 at index 0
1
Inserted 1 at index 1
2
Inserted 2 at index 2
3
Inserted 3 at index 3
4
Inserted 4 at index 4
5
Inserted 5 at index 5
6
Inserted 6 at index 6
7
Inserted 7 at index 7
8
Inserted 8 at index 8
9
Inserted 9 at index 9
10
Inserted 10 at index 0
11
Inserted 11 at index 1
12
Inserted 12 at index 2
99
Inserted 99 at index 9
36
Inserted 36 at index 6
12
Inserted 12 at index 2
36
Inserted 36 at index 6
99
Inserted 99 at index 9
112
Inserted 112 at index 2
2
Inserted 2 at index 2

Hash Table Contents:
0: 10 -> 0 -> NULL
1: 11 -> 1 -> NULL
2: 2 -> 112 -> 12 -> 12 -> 2 -> NULL
3: 3 -> NULL
4: 4 -> NULL
5: 5 -> NULL
6: 36 -> 36 -> 6 -> NULL
7: 7 -> NULL
8: 8 -> NULL
9: 99 -> 99 -> 9 -> NULL


=== Code Execution Successful ===
```

Conclusion:

- A **Hash Table** is a data structure that maps keys to values using a hash function.
- In **Separate Chaining**, each index of the hash table stores a linked list of elements that hash to the same index.
- When a collision occurs (i.e., two keys produce the same hash index), the new element is added to the linked list at that position.

Time Complexity (TC):

- **Insertion:** O(1) average, O(n) worst case (when all elements collide).
- **Search:** O(1) average, O(n) worst case.
- **Deletion:** O(1) average, O(n) worst case.

Space Complexity (SC):

- **O(n + m)** → *n* = number of keys, *m* = table size (extra space for linked lists).