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: Varad Amol Pisale PRN : 124B1B043**

**Department: Computer Engineering Division : A**

**Course Name :** **Data Structures Laboratory Code:BCE23PC02**

**Completion Date : 13/08/2025**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Assignment No. 12**

Problem Statement: Consider an employee database of N employees considering emp Id and name as data members. Make use of a hash table implementation to quickly look up the employer's id number. Implement above scenario using hashing and linear probing.

Source Code :

#include <iostream>

#include <string>

using namespace std;

#define SIZE 10 // Size of hash table

class Employee

{

    int empID;

    string name;

public:

    Employee()

    {

        empID = -1; // -1 indicates empty slot

        name = "";

    }

    void setEmployee(int id, const string &n)

    {

        empID = id;

        name = n;

    }

    int getID() const

    {

        return empID;

    }

    string getName() const

    {

        return name;

    }

    bool isEmpty() const

    {

        return empID == -1;

    }

};

class HashTable

{

    Employee table[SIZE];

    bool occupied[SIZE]; // to track filled slots

public:

    HashTable()

    {

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

            occupied[i] = false;

    }

    int hash(int key)

    {

        return key % SIZE;

    }

    void insert(int empID, const string &name)

    {

        int index = hash(empID); // empID%SIZE;

        int startIndex = index;

        while (occupied[index])

        {

            index = (index + 1) % SIZE;

            if (index == startIndex)

            {

                cout << "Hash table full! Cannot insert employee " << empID << endl;

                return;

            }

        }

        table[index].setEmployee(empID, name);

        occupied[index] = true;

        cout << "Employee inserted at index " << index << endl;

    }

    void display()

    {

        cout << "\nEmployee Database:\n";

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

        {

            if (occupied[i])

            {

                cout << i << " -> ID: " << table[i].getID()

                     << ", Name: " << table[i].getName() << endl;

            }

            else

                cout << i << " -> Empty" << endl;

        }

    }

};

int main()

{

    HashTable ht;

    ht.insert(100, "Alice");

    ht.insert(101, "Alice");

    ht.insert(112, "Bob");

    ht.insert(122, "Charlie");

    ht.insert(133, "David");

    ht.insert(144, "Eve");

    //  ht.display();

    ht.insert(145, "John");

    ht.insert(156, "John");

    ht.insert(167, "John");

    ht.insert(178, "John");

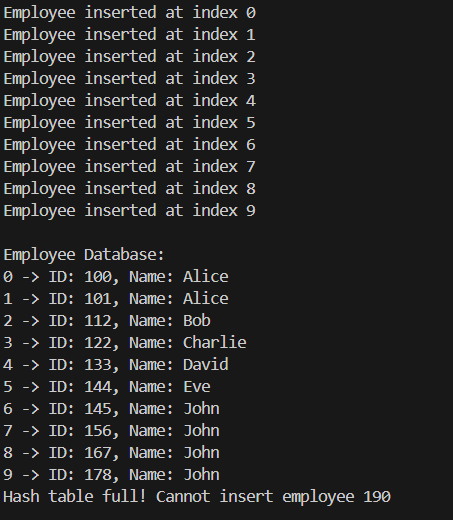
    ht.display();

    ht.insert(190, "John");

    return 0;

}

Screen Shot of Output :



Conclusion: Hence we have implemented an Employee database using hashtable