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: Hariom Shrikrishna Gundale **PRN : 124B1B036**

Department: Computer Engineering **Division : A**

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

Completion Date : 20/10/2025

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.

```
/*
#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;
}
*/
```

Source Code :

```
#include <iostream>
#include <string>
```

```
using namespace std;

#define SIZE 10

class Employee {
    int empID;
    string name;

public:
    Employee() {
        empID = -1;
        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];

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);
        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 search(int empID) {
        int index = hash(empID);
        int startIndex = index;

        while (occupied[index]) {
            if (table[index].getID() == empID) {
                cout << " Employee Found at index " << index << endl;
                cout << "   ID: " << table[index].getID()
                     << ", Name: " << table[index].getName() << endl;
                return;
            }

            index = (index + 1) % SIZE;
            if (index == startIndex)
                break;
        }
        cout << " Employee with ID " << empID << " not found.\n";
    }

    void display() {
        cout << "\n Employee 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\n";
        }
    }
};

int main() {
    HashTable ht;
    int choice, id;
    string name;

    while (true) {
```

```
        cout << "\n===== EMPLOYEE HASH TABLE MENU =====\n";
        cout << "1. Insert Employee\n";
        cout << "2. Search Employee\n";
        cout << "3. Display Employee Database\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter Employee ID: ";
            cin >> id;
            cout << "Enter Employee Name: ";
            cin >> name;
            ht.insert(id, name);
            break;

        case 2:
            cout << "Enter Employee ID to search: ";
            cin >> id;
            ht.search(id);
            break;

        case 3:
            ht.display();
            break;

        case 4:
            cout << "Exiting program...\n";
            return 0;

        default:
            cout << "Invalid choice! Try again.\n";
        }
    }
    return 0;
}
```

Screen Shot of Output :

```
PS P:\DSA_Asssignment> ./Assignment_12
3. Display Employee Database
4. Exit
Enter your choice: 1
Enter Employee ID: 111
Enter Employee Name: Kamlesh
 Employee inserted at index 2

===== EMPLOYEE HASH TABLE MENU =====
1. Insert Employee
2. Search Employee
3. Display Employee Database
4. Exit
Enter your choice: 3

 Employee Database:
0 -> Empty
1 -> ID: 101, Name: Hariom
2 -> ID: 111, Name: Kamlesh
3 -> ID: 103, Name: Om
4 -> Empty
5 -> ID: 105, Name: Varad
6 -> Empty
7 -> Empty
8 -> Empty
9 -> Empty
```

Conclusion:

Thus, we have successfully implemented the C++ Program on above given scenario using Hashing and Linear Probing.