CHANDIGARH COLLEGE OF ENGINEERING & TECHNOLOGY (DEGREE WING)

Government institute under Chandigarh (UT) Administration, affiliated to Punjab University, Chandigarh

Department of Computer Science & Engineering

Semester: CSE 3rd

SUBJECT: Data Structures Practical (CS351)

Problem 10: Case Study of Hashing

Submitted by:	**Submitted to:**
Bhavyam Dhand	Dr. R.B. Patel
(CO23316)	(Professor)

CODE

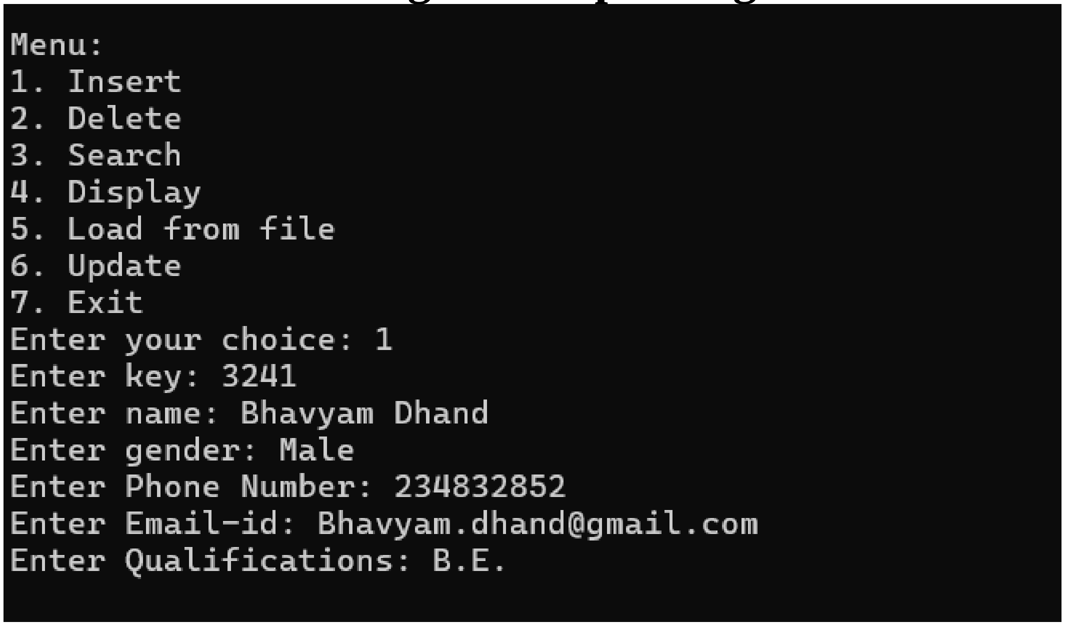
```
#include <bits/stdc++.h>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
//to depict empty entries as empty
#define EMPTY -1
#define DELETED -2
//function to verify name is alphabetical
bool VerifyName(string name) {
    for (char c : name) {
        if (!isalpha(c) && c != '.' && c != ' ') return false;
    }
    return true;
}

struct EmployeeHashNode {
    int key;
    string name;
    string gender;
    string mobile;
    string email;
    string qualifications;
    bool occupied;
    // Constructor to initialize an empty node
    EmployeeHashNode() : key(EMPTY), name(""), gender(""), mobile(""), email(""), 
qualifications(""), occupied(false) {}
    //Function to input Node data
    void Input(int Key) {
        this->key = Key;
        do {
            cout << "Enter name: ";
            getline(cin, name);
        } while (!VerifyName(name));
        cout << "Enter gender: ";
        getline(cin, gender);
        cout << "Enter Phone Number: ";
        getline(cin, mobile);
        cout << "Enter Email-id: ";
        getline(cin, email);
        cout << "Enter Qualifications: ";
        getline(cin, qualifications);
        occupied = true;
    }
    //function which displays Node Data
    void Display() const {
        if (occupied) {
            cout << "Key: " << key << "  Name: " << name
```

```
                 << "  Gender: " << gender
                 << "  Mobile: " << mobile << "  Email: " << email
                 << "  Qualifications: " << qualifications << endl;
        } else {
            cout <<" Empty" << endl;
        }
    }
};
//Datatype which depicts HashTable
struct HashTable {
    vector<EmployeeHashNode> Table;
    int M;

    HashTable(int MaxSize) : M(MaxSize) {
        Table.resize(MaxSize);
    }
};

// Hash function to compute the index
int HashFunction(int key, int M) {
    return key % M;  // Simple modulus hash function
}

// Insert a node into the hash table
void insert(HashTable& H, const EmployeeHashNode& Node) {
    int index = HashFunction(Node.key, H.M);
    int OGIndex = index;
    while (H.Table[index].occupied && (H.Table[index].key != EMPTY &&
H.Table[index].key != DELETED)) {
        index = (index + 1) % H.M;
        if (index == OGIndex) {
            cout << "Hash table is full!\n";
            return;
        }
    }
    H.Table[index] = Node;
}

// Delete a node from the hash table
void Delete(HashTable& H, int key) {
    int index = HashFunction(key, H.M);
    int originalIndex = index;

    while (H.Table[index].occupied) {
        if (H.Table[index].key == key) {
            H.Table[index] = EmployeeHashNode();
            H.Table[index].key = DELETED;
            cout << "Deleted employee with key: " << key << "\n";
            return;
```

```cpp
        }
        index = (index + 1) % H.M;
        if (index == originalIndex)
            break;
    }
    cout << "Employee with key " << key << " not found.\n";
}

// Search for a node by key in the hash table
void Search(const HashTable& H, int key) {
    int index = HashFunction(key, H.M);
    int OGindex = index;
    while (H.Table[index].occupied) {
        if (H.Table[index].key == key) {
            cout << "Employee found:\n";
            H.Table[index].Display();
            return;
        }
        index = (index + 1) % H.M;
        if (index == OGindex)
            break;
    }
    cout << "Employee not found.\n";
}

// Display all nodes in the hash table
void Display(const HashTable& H) {
    for (int i = 0; i < H.M; ++i) {
            cout << "Index " << i << ": ";
            H.Table[i].Display();
    }
}

// Update the details of an employee
void Update(HashTable& H, int key) {
    int index = HashFunction(key, H.M);
    int originalIndex = index;

    while (H.Table[index].occupied) {
        if (H.Table[index].key == key) {
            cout << "Employee found. Enter new details:\n";
            H.Table[index].Input(key);
            cout << "Employee details updated successfully.\n";
            return;
        }
        index = (index + 1) % H.M;
        if (index == originalIndex)
            break;
    }
    cout << "Employee with key " << key << " not found.\n";
}
//load data from emp.dat file
void LoadFromFile(HashTable& H, const string& filename) {
    ifstream infile(filename);

    if (!infile) {
        cout << "Error opening file: " << filename << endl;
        return;
    }

    string name, gender, mobile, email, qualifications;
    int key;

    while (infile >> key) {
        // Skip the newline character after the key
        infile.ignore();

        // Read the name, gender, mobile, email, and qualifications
        getline(infile, name, ' ');  // Read until the first space for name
        getline(infile, gender, ' '); // Read until the next space for gender
        getline(infile, mobile, ' '); // Read until the next space for mobile
        getline(infile, email, ' ');  // Read until the next space for email
        getline(infile, qualifications); // Read the rest for qualifications

        // Create an EmployeeHashNode with the parsed data
        EmployeeHashNode Node;
        Node.key = key;
        Node.name = name;
        Node.gender = gender;
        Node.mobile = mobile;
        Node.email = email;
        Node.qualifications = qualifications;
        Node.occupied = true;

        // Insert the node into the hash table
        insert(H, Node);
    }
    infile.close();
    cout << "Data loaded from file successfully.\n";
}

//user interface
int main() {
    int size;
    cout << "Enter hash table size: ";
    cin >> size;
    cin.ignore();
    HashTable table(size);

    int choice;
    do {
        cout << "\nMenu:\n1. Insert\n2. Delete\n3. Search\n4. Display\n5. Load from
file\n6. Update\n7. Exit\nEnter your choice: ";
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1: {
                EmployeeHashNode Node;
                int key;
                cout << "Enter key: ";
                cin >> key;
                cin.ignore();
                Node.Input(key);
                insert(table, Node);
                break;
            }
            case 2: {
                int key;
                cout << "Enter key to delete: ";
                cin >> key;
                Delete(table, key);
                break;
            }
            case 3: {
                int key;
                cout << "Enter key to search: ";
                cin >> key;
                Search(table, key);
                break;
            }
            case 4:
                Display(table);
                break;
            case 5:
                LoadFromFile(table, "EMP.dat");
                break;
            case 6: {
                int key;
                cout << "Enter key to update: ";
                cin >> key;
                Update(table, key);
                break;
            }
            case 7:
                cout << "Exiting program.\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 7);

    return 0;
}
```

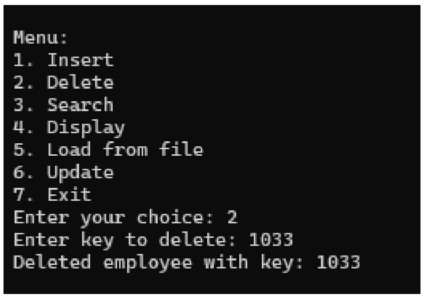
<u>Output</u>

1. Insert a record using Linear probing



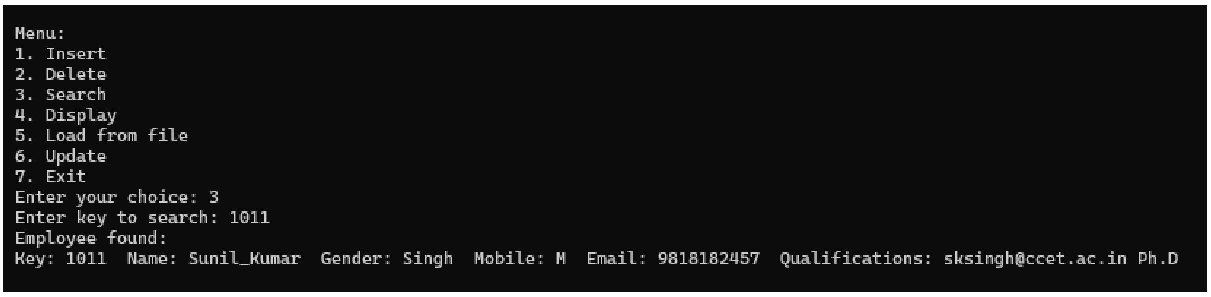
```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 1
Enter key: 3241
Enter name: Bhavyam Dhand
Enter gender: Male
Enter Phone Number: 234832852
Enter Email-id: Bhavyam.dhand@gmail.com
Enter Qualifications: B.E.
```

2. Delete a Record from Hash Table



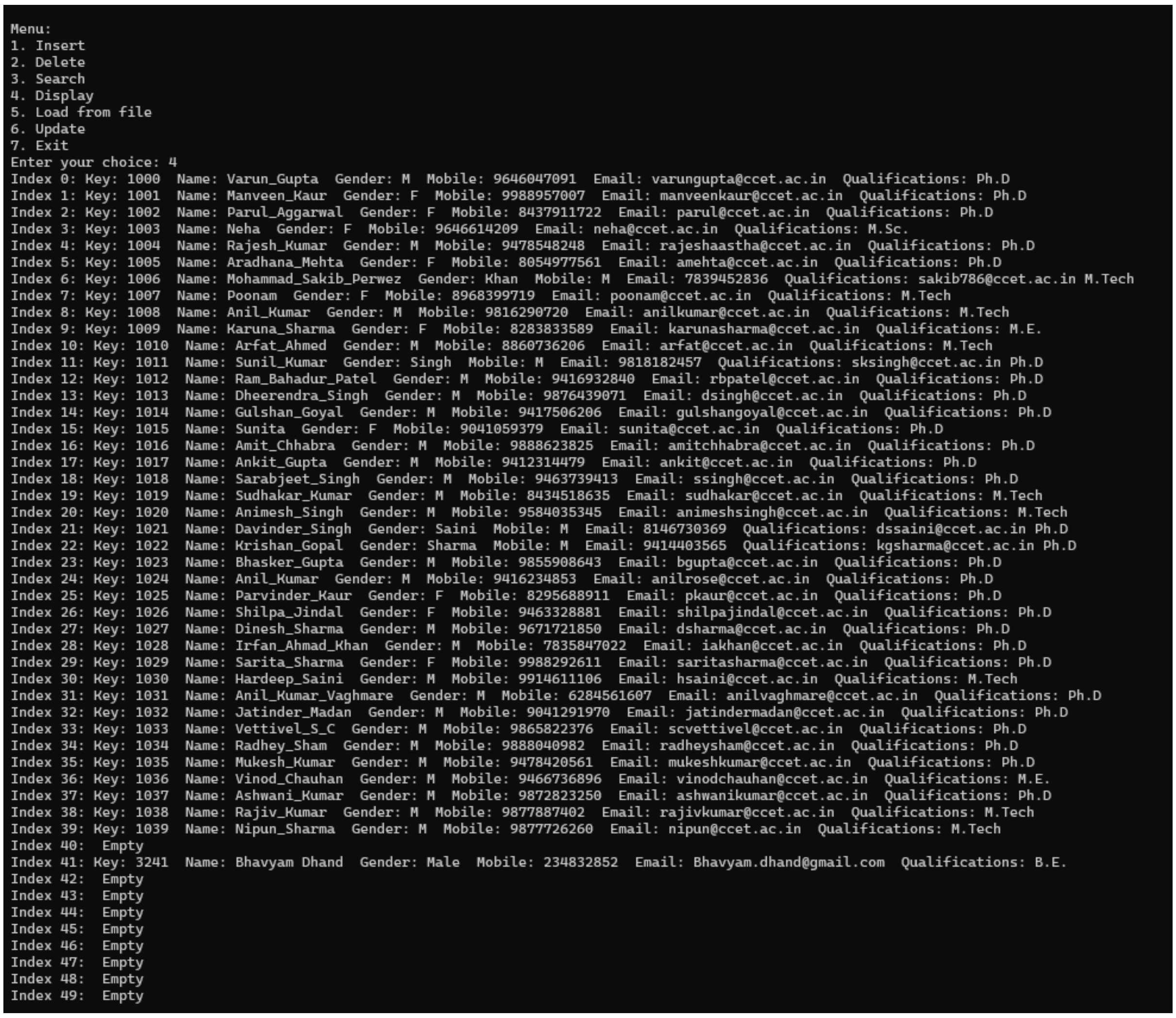
```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 2
Enter key to delete: 1033
Deleted employee with key: 1033
```

3. Search a Record in Hash Table



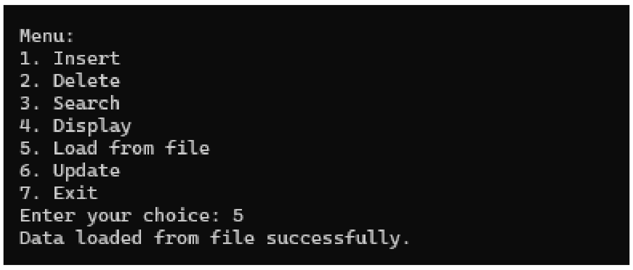
```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 3
Enter key to search: 1011
Employee found:
Key: 1011  Name: Sunil_Kumar  Gender: Singh  Mobile: M  Email: 9818182457  Qualifications: sksingh@ccet.ac.in Ph.D
```

4. Display Hash Table



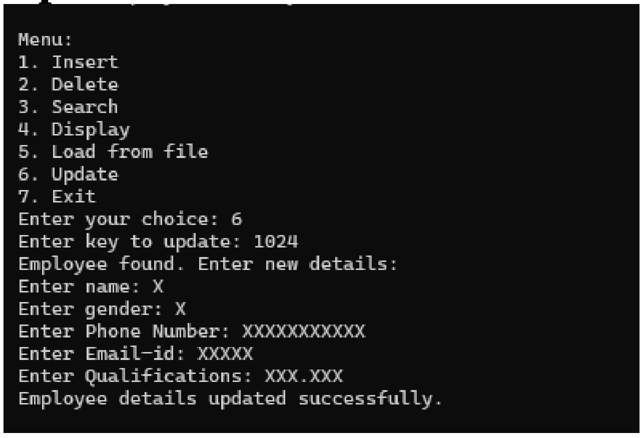
```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 4
Index 0: Key: 1000  Name: Varun_Gupta  Gender: M  Mobile: 9646047091  Email: varungupta@ccet.ac.in  Qualifications: Ph.D
Index 1: Key: 1001  Name: Manveen_Kaur  Gender: F  Mobile: 9988957007  Email: manveenkaur@ccet.ac.in  Qualifications: Ph.D
Index 2: Key: 1002  Name: Parul_Aggarwal  Gender: F  Mobile: 8437911722  Email: parul@ccet.ac.in  Qualifications: Ph.D
Index 3: Key: 1003  Name: Neha  Gender: F  Mobile: 9646614209  Email: neha@ccet.ac.in  Qualifications: M.Sc.
Index 4: Key: 1004  Name: Rajesh_Kumar  Gender: M  Mobile: 9478548248  Email: rajeshaastha@ccet.ac.in  Qualifications: Ph.D
Index 5: Key: 1005  Name: Aradhana_Mehta  Gender: F  Mobile: 8054977561  Email: amehta@ccet.ac.in  Qualifications: Ph.D
Index 6: Key: 1006  Name: Mohammad_Sakib_Perwez  Gender: Khan  Mobile: M  Email: 7839452836  Qualifications: sakib786@ccet.ac.in M.Tech
Index 7: Key: 1007  Name: Poonam  Gender: F  Mobile: 8968399719  Email: poonam@ccet.ac.in  Qualifications: M.Tech
Index 8: Key: 1008  Name: Anil_Kumar  Gender: M  Mobile: 9816290720  Email: anilkumar@ccet.ac.in  Qualifications: M.Tech
Index 9: Key: 1009  Name: Karuna_Sharma  Gender: F  Mobile: 8283833589  Email: karunasharma@ccet.ac.in  Qualifications: M.E.
Index 10: Key: 1010  Name: Arfat_Ahmed  Gender: M  Mobile: 8860736206  Email: arfat@ccet.ac.in  Qualifications: M.Tech
Index 11: Key: 1011  Name: Sunil_Kumar  Gender: Singh  Mobile: M  Email: 9818182457  Qualifications: sksingh@ccet.ac.in Ph.D
Index 12: Key: 1012  Name: Ram_Bahadur_Patel  Gender: M  Mobile: 9416932840  Email: rbpatel@ccet.ac.in  Qualifications: Ph.D
Index 13: Key: 1013  Name: Dheerendra_Singh  Gender: M  Mobile: 9876439071  Email: dsingh@ccet.ac.in  Qualifications: Ph.D
Index 14: Key: 1014  Name: Gulshan_Goyal  Gender: M  Mobile: 9417506206  Email: gulshangoyal@ccet.ac.in  Qualifications: Ph.D
Index 15: Key: 1015  Name: Sunita  Gender: F  Mobile: 9041059379  Email: sunita@ccet.ac.in  Qualifications: Ph.D
Index 16: Key: 1016  Name: Amit_Chhabra  Gender: M  Mobile: 9888623825  Email: amitchhabra@ccet.ac.in  Qualifications: Ph.D
Index 17: Key: 1017  Name: Ankit_Gupta  Gender: M  Mobile: 9412314479  Email: ankit@ccet.ac.in  Qualifications: Ph.D
Index 18: Key: 1018  Name: Sarabjeet_Singh  Gender: M  Mobile: 9463739413  Email: ssingh@ccet.ac.in  Qualifications: Ph.D
Index 19: Key: 1019  Name: Sudhakar_Kumar  Gender: M  Mobile: 8434518635  Email: sudhakar@ccet.ac.in  Qualifications: M.Tech
Index 20: Key: 1020  Name: Animesh_Singh  Gender: M  Mobile: 9584035345  Email: animeshsingh@ccet.ac.in  Qualifications: M.Tech
Index 21: Key: 1021  Name: Davinder_Singh  Gender: Saini  Mobile: M  Email: 8146730369  Qualifications: dssaini@ccet.ac.in Ph.D
Index 22: Key: 1022  Name: Krishan_Gopal  Gender: Sharma  Mobile: M  Email: 9414403565  Qualifications: kgsharma@ccet.ac.in Ph.D
Index 23: Key: 1023  Name: Bhasker_Gupta  Gender: M  Mobile: 9855908643  Email: bgupta@ccet.ac.in  Qualifications: Ph.D
Index 24: Key: 1024  Name: Anil_Kumar  Gender: M  Mobile: 9416234853  Email: anilrose@ccet.ac.in  Qualifications: Ph.D
Index 25: Key: 1025  Name: Parvinder_Kaur  Gender: F  Mobile: 8295688911  Email: pkaur@ccet.ac.in  Qualifications: Ph.D
Index 26: Key: 1026  Name: Shilpa_Jindal  Gender: F  Mobile: 9463328881  Email: shilpajindal@ccet.ac.in  Qualifications: Ph.D
Index 27: Key: 1027  Name: Dinesh_Sharma  Gender: M  Mobile: 9671721850  Email: dsharma@ccet.ac.in  Qualifications: Ph.D
Index 28: Key: 1028  Name: Irfan_Ahmad_Khan  Gender: M  Mobile: 7835847022  Email: iakhan@ccet.ac.in  Qualifications: Ph.D
Index 29: Key: 1029  Name: Sarita_Sharma  Gender: F  Mobile: 9988292611  Email: saritasharma@ccet.ac.in  Qualifications: Ph.D
Index 30: Key: 1030  Name: Hardeep_Saini  Gender: M  Mobile: 9914611106  Email: hsaini@ccet.ac.in  Qualifications: M.Tech
Index 31: Key: 1031  Name: Anil_Kumar_Vaghmare  Gender: M  Mobile: 6284561607  Email: anilvaghmare@ccet.ac.in  Qualifications: Ph.D
Index 32: Key: 1032  Name: Jatinder_Madan  Gender: M  Mobile: 9041291970  Email: jatindermadan@ccet.ac.in  Qualifications: Ph.D
Index 33: Key: 1033  Name: Vettivel_S_C  Gender: M  Mobile: 9865822376  Email: scvettivel@ccet.ac.in  Qualifications: Ph.D
Index 34: Key: 1034  Name: Radhey_Sham  Gender: M  Mobile: 9888040982  Email: radheysham@ccet.ac.in  Qualifications: Ph.D
Index 35: Key: 1035  Name: Mukesh_Kumar  Gender: M  Mobile: 9478420561  Email: mukeshkumar@ccet.ac.in  Qualifications: Ph.D
Index 36: Key: 1036  Name: Vinod_Chauhan  Gender: M  Mobile: 9466736896  Email: vinodchauhan@ccet.ac.in  Qualifications: M.E.
Index 37: Key: 1037  Name: Ashwani_Kumar  Gender: M  Mobile: 9872823250  Email: ashwanikumar@ccet.ac.in  Qualifications: Ph.D
Index 38: Key: 1038  Name: Rajiv_Kumar  Gender: M  Mobile: 9877887402  Email: rajivkumar@ccet.ac.in  Qualifications: M.Tech
Index 39: Key: 1039  Name: Nipun_Sharma  Gender: M  Mobile: 9877726260  Email: nipun@ccet.ac.in  Qualifications: M.Tech
Index 40:  Empty
Index 41: Key: 3241  Name: Bhavyam Dhand  Gender: Male  Mobile: 234832852  Email: Bhavyam.dhand@gmail.com  Qualifications: B.E.
Index 42:  Empty
Index 43:  Empty
Index 44:  Empty
Index 45:  Empty
Index 46:  Empty
Index 47:  Empty
Index 48:  Empty
Index 49:  Empty
```

5. Load from EMP.dat file:



```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 5
Data loaded from file successfully.
```

6. Update Record



```
Menu:
1. Insert
2. Delete
3. Search
4. Display
5. Load from file
6. Update
7. Exit
Enter your choice: 6
Enter key to update: 1024
Employee found. Enter new details:
Enter name: X
Enter gender: X
Enter Phone Number: XXXXXXXXXXX
Enter Email-id: XXXXX
Enter Qualifications: XXX.XXX
Employee details updated successfully.
```

Machine Code:

```
01001000 01100001 01110011 01101000 00100000
01110100 01100001 01100010 01101100 01100101
00100000 01101001 01101110 01101001 01110100
01101001 01100001 01101100 01101001 01111010
01100101 01100100 00101110
```

```
01001001 01101110 01110011 01100101 01110010
01110100 01100101 01100100 00100000 01100101
01101101 01110000 01101100 01101111 01111001
01100101 01100101 00100000 01110111 01101001
01110100 01101000 00100000 01001011 01100101
01111001 01001001 01000100 00111010 00100000
00110011 00110100 00110010 00111000 00100000
01100001 01110100 00100000 01101001 01101110
01100100 01100101 01111000 00111010 00100000
00111000
```

```
01001001 01101110 01110011 01100101 01110010
01110100 01100101 01100100 00100000 01100101
01101101 01110000 01101100 01101111 01111001
01100101 01100101 00100000 01110111 01101001
01110100 01101000 00100000 01001011 01100101
01111001 01001001 01000100 00111010 00100000
00110010 00110010 00110100 00110001 00100000
01100001 01110100 00100000 01101001 01101110
01100100 01100101 01111000 00111010 00100000
00110001
```

```
01000100 01101001 01110011 01110000 01101100
01100001 01111001 01100101 01100100 00100000
01101000 01100001 01110011 01101000 00100000
01110100 01100001 01100010 01101100 01100101
00100000 01100011 01101111 01101110 01110100
01100101 01101110 01110100 01110011 00101110
```