JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

DATA STRUCTURES LAB (15B17CI371)

MINI PROJECT REPORT

GROUP MEMBERS:-

NAME	ENROLLMENT NUMBER	BATCH
DRISHIKA CHAUHAN	20103110	B4

SUBMITTED TO FACULTY:

- Dr. VIVEK KUMAR SINGH
- Dr. MANISH KUMAR THAKUR
- Ms. NISHTHA AHUJA

CERTIFICATE

This is to certify that Drishika Chauhan of B-Tech, second year from CSE Branch of Batch B4 have successfully created the project on the topic "Phone Directory using data structures" under the guidance of Dr. Vivek Kumar Singh, Dr. Manish Kumar Thakur and Ms. Nishtha Ahuja.

Dr. VIVEK KUMAR SINGH

(LAB FACULTY)

Dr. MANISH KUMAR THAKUR

(LAB FACULTY)

Ms. NISHTHA AHUJA

(LAB FACULTY)

Date of submission: 17-12-2021

ACKNOWLEDGEMENT

Every successful venture has the blessing and the support of many behind it. It would be unjust if every such support is not revealed and thanked in person. If words were considered as symbol of approval and token of appreciation then let the words play the heralding role expressing our gratitude. I would like to place on record our deep sense of gratitude to Dr. Vivek Kumar Singh and Ms. Nishtha Ahuja for giving me an opportunity to work and their constant guidance throughout the project. Also, we would like to thank Dr. Manish Kumar Thakur for his support and guidance. Finally, an honorable mention goes to our friends and family for their support to do us this project. Without help of the particular's mentioned above, we would have faced many difficulties while pursuing this project.

PHONE DIRECTORY

PROBLEM-STATEMENT: **Implementation of Phone Directory using hashing.**

INTRODUCTION:

In this project, a phone directory has been implemented using hashing in C++ programming language. Here, the type of hashing used lies under open hashing i.e., chaining. Hashing is one of the best techniques for the implementation of phone directory because it is one of the fastest searching techniques and one of the main functionalities of a phone directory involves searching.

When searching for an element in the hash table, in the best case, the element is directly found at the location indicated by its key. So, we only need to calculate the hash key and then retrieve the element. This is achieved in constant time. So, best case complexity is O(1). Worst case complexity is O(n).

We have: Space Complexity: O(nm) => O(n)

Where, m is the size of the hash table (here, m = 26) and n is the number of items inserted.

FEATURES:

Features of the Phone Directory are as follows:

- To insert new records in the directory
- To remove any existing records
- To modify the name of user
- To modify the phone number of user
- To search for any record
- To display all the records in alphabetical order to make searching easier

OUTPUT:

➢ INSERTION OF DATA:

```
PS C:\Users\Drishika\OneDrive\Desktop\DS Project> cd "c:\Users\Drishika\OneDrive\Desktop\DS Project\" ;
|~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~|
|--------------|---------------------------------------------|--------------------|
|**************|**************** PHONE DIRECTORY *************|*********************|
|--------------|---------------------------------------------|--------------------|
|~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~|


Enter number of records you want to enter: 5

Enter your choice which you want to execute:
1: Adding a contact
2: Deleting a contact
3: Displaying contacts
4: Searching a contact by name
5: Modify name of the user
6: Modify phone number of the user
7: Exit

1
Enter the name and phone number respectively:
Drishika 6307633826
Enter the name and phone number respectively:
Ram 9792345689
Enter the name and phone number respectively:
Lucky 9876543210
Enter the name and phone number respectively:
Dev 1234567890
Enter the name and phone number respectively:
Sandhya 8299038552
```

➢ DISPLAYING DATA:

```
3
Displaying Phone Directory:
|-------------------------------------------------------|
 User: Dev || Phone number: 1234567890
|-------------------------------------------------------|

|-------------------------------------------------------|
 User: Drishika || Phone number: 6307633826
|-------------------------------------------------------|

|-------------------------------------------------------|
 User: Lucky || Phone number: 9876543210
|-------------------------------------------------------|

|-------------------------------------------------------|
 User: Ram || Phone number: 9792345689
|-------------------------------------------------------|

|-------------------------------------------------------|
 User: Sandhya || Phone number: 8299038552
|-------------------------------------------------------|
```

➢ SEARCHING FOR A RECORD:

```
Enter your choice which you want to execute:
1: Adding a contact
2: Deleting a contact
3: Displaying contacts
4: Searching a contact by name
5: Modify name of the user
6: Modify phone number of the user
7: Exit

4
Enter the name of person whose details are to be searched: Drishika
Name: Drishika, Phone Number: 6307633826
```

➢ DELETION OF RECORD:

```
2
Enter the name of person whose details are to be deleted: Ram
Ram's details were removed from the phone directory successfully...

Enter your choice which you want to execute:
1: Adding a contact
2: Deleting a contact
3: Displaying contacts
4: Searching a contact by name
5: Modify name of the user
6: Modify phone number of the user
7: Exit

3
Displaying Phone Directory:
|-----------------------------------------------------------|
 User: Dev || Phone number: 1234567890
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Drishika || Phone number: 6307633826
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Lucky || Phone number: 9876543210
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Sandhya || Phone number: 8299038552
|-----------------------------------------------------------|
```

➢ MODIFY NAME AND PHONE NUMBER OF THE USER:

```
5: Modify name of the user
6: Modify phone number of the user
7: Exit

5
Enter the name of person whose details are to be modified: Lucky
Enter the new name: Lakshman
Record has been modified!!!

Do you wish to modify the phone number as well?
1.Yes
2.No
Enter your choice: 1
Enter the name of person whose phone number is to be modified: Lakshman
Enter the new phone number: 9792202416
Name: Lakshman, Modified Phone Number: 9792202416


Enter your choice which you want to execute:
1: Adding a contact
2: Deleting a contact
3: Displaying contacts
4: Searching a contact by name
5: Modify name of the user
6: Modify phone number of the user
7: Exit

3
Displaying Phone Directory:
|-----------------------------------------------------------|
 User: Dev || Phone number: 1234567890
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Drishika || Phone number: 6307633826
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Lakshman || Phone number: 9792202416
|-----------------------------------------------------------|

|-----------------------------------------------------------|
 User: Sandhya || Phone number: 8299038552
|-----------------------------------------------------------|
```

```
Enter your choice which you want to execute:
1: Adding a contact
2: Deleting a contact
3: Displaying contacts
4: Searching a contact by name
5: Modify name of the user
6: Modify phone number of the user
7: Exit

7
$$ ------------------------Project Made By:---------------------- $$
--> Drishika Chauhan
```

CODE:

```
#include<iostream>
#include<string.h>
#include<math.h>
using namespace std;

void exitFunction();

class Details
{
public:
    string name;
    long long int phone_num;
    Details *next;
};

class HashTable
{
public:
    Details **directory;
    HashTable();
    int hashFunction(string key);
    void insert(long long int phoneNumber, string name);
    int countItemsInIndex(int index);
    void deletion();
    void searchByName();
    void displayItemsInIndex(int index);
    void modifyPhoneNum();
    void modifyName();
    ~HashTable();
};

HashTable::HashTable()
{
    //directory = new Details*[26];
    for (int i = 0; i < 26; i++)
    {
        directory[i] = new Details;
        directory[i]->name = "empty";
        directory[i]->phone_num = -1;
        directory[i]->next = NULL;
    }
}
```

```
int HashTable::hashFunction(string key)
{
    int idx = 0;
    if(key[0] >= 'A' && key[0] <= 'Z')
    {
        idx = tolower(key[0]);
    }
    else
    {
        idx = key[0];
    }
    return (idx-97);
}

void HashTable::insert(long long int phoneNumber, string Uname)
{
    int index = hashFunction(Uname);
    Details* t = new Details;
    t->name = Uname;
    t->phone_num = phoneNumber;
    t->next = nullptr;

    // cout << "Index: " << index << endl;

    if(directory[index] == nullptr)
    {
        directory[index] = t;
    }
    else
    {
        Details* p = directory[index];
        Details* q = directory[index];
        while(p != nullptr)
        {
            q = p;
            p = p->next;
        }

        // Case: insert position is first
        if (q == directory[index])
        {
            t->next = directory[index];
            directory[index] = t;
        }
        else
        {
            t->next = q->next;
```

```
            q->next = t;
        }
    }
}

int HashTable::countItemsInIndex(int index)
{
    int count = 0;
    if(directory[index]->name == "empty")
    {
        return count;
    }
    else
    {
        count++;
        Details *temp=(directory[index]);
        while(temp->next!=NULL)
        {
            count++;
            temp=temp->next;
        }
    }
}

void HashTable::searchByName()
{
    string name;
    cout<<"Enter the name of person whose details are to be searched: ";
    cin >> name;

    int index = hashFunction(name);

    // cout << "index: " << index << endl;
    bool flag = false;
    int i = 0;

    long long int phone;
    Details *ptr = (directory[index]);

    while(ptr != NULL)
    {
        if(ptr -> name == name)
        {
            flag = true;
            phone = ptr -> phone_num;
            break;
        }
```

```
        ptr = ptr->next;
    }

    if(flag == true)
    {
        cout << "Name: " << ptr->name << ", " << "Phone Number: " << ptr-
>phone_num << endl;
    }
    else
    {
        int choice;
        cout << "No such entry is present in the directory!!! Would you
like to view all the entries present? \n1.Yes\n2.No" << endl;
        cin >> choice;
        if(choice == 1)
        {
            for(int i = 0; i < 26; i++)
            {
                displayItemsInIndex(i);
            }
        }
        else
        {
            exitFunction();
        }
    }
}

void sortList(Details* head)
{
    Details* temp = head;
    while (temp)
    {
        Details* min = temp;
        Details* r = temp->next;

        while (r)
        {
            if (min->name > r->name)
                min = r;

            r = r->next;
        }

        swap(temp->name, min->name);
        swap(temp->phone_num, min->phone_num);
```

```
        temp = temp->next;
    }
}

void HashTable::displayItemsInIndex(int i)
{
    Details *temp = (directory[i]);
    sortList(temp);

    Details *curr = temp->next;

    if(temp->name != "empty")
    {
        // Condition to check for duplicate elements
        while(curr->next != NULL)
        {
            int res = curr->next->name.compare(curr->name);
            if(res == 0)
            {
                if(curr->next->phone_num == curr->phone_num)
                {
                    curr->name = "empty";
                    curr->phone_num = -1;
                    break;
                }
                curr = curr->next;
            }
            else
            {
                curr = curr->next;
            }
        }

        while (temp != nullptr)
        {
            if(temp->name == "empty")
            {
                temp = temp->next;
                continue;
            }

            cout << "|-------------------------------------------------
-----|" << endl;
            cout << " User: " << temp->name << " || Phone number: " <<
temp->phone_num << "\t" << endl;
            cout << "|-------------------------------------------------
-----|" << endl << endl;
```

```
            temp = temp->next;
        }
    }

}

void HashTable::modifyName()
{
    string name;
    cout<<"Enter the name of person whose details are to be modified: ";
    cin >> name;

    int index = hashFunction(name);
    bool flag = false;
    int i = 0;

    long long int phone;
    Details *ptr = (directory[index]);

    while(ptr != NULL)
    {
        if(ptr -> name == name)
        {
            flag = true;
            phone = ptr -> phone_num;
            break;
        }

        ptr = ptr->next;

    }

    if(flag == true)
    {
        string newName;
        int ch;
        cout << "Enter the new name: ";
        cin >> newName;

        insert(ptr->phone_num, newName);
        ptr->name = "empty";
        ptr->phone_num = -1;
        cout << "Record has been modified!!!" << endl << endl;
        cout << "Do you wish to modify the phone number as
well?\n1.Yes\n2.No\nEnter your choice: ";
        cin >> ch;
```

```
        if(ch == 1)
        {
            modifyPhoneNum();
        }
        else if(ch == 2)
        {
            cout << "Displaying Phone Directory: " << endl << endl;
            for(int i = 0; i < 26; i++)
            {
                displayItemsInIndex(i);
            }
        }
    }

    else
    {
        int choice;
        cout << "No such entry is present in the directory!!! Would you
like to view all the entries present? \n1.Yes\n2.No" << endl;
        cin >> choice;
        if(choice == 1)
        {
            for(int i = 0; i < 26; i++)
            {
                displayItemsInIndex(i);
            }
        }
        else
        {
            exitFunction();
        }
    }
}

void HashTable::modifyPhoneNum()
{
    string name;
    cout<<"Enter the name of person whose phone number is to be modified:
";
    cin >> name;

    int index = hashFunction(name);
    bool flag = false;
    int i = 0;

    long long int phone;
    Details *ptr = (directory[index]);
```

```
    while(ptr != NULL)
    {
        if(ptr -> name == name)
        {
            flag = true;
            phone = ptr -> phone_num;
            break;
        }

        ptr = ptr->next;

    }

    if(flag == true)
    {
        long long int newPhoneNumber;
        int ch;
        cout << "Enter the new phone number: ";
        cin >> newPhoneNumber;
        ptr->phone_num = newPhoneNumber;
        cout << "Name: " << ptr->name << ", " << "Modified Phone Number:
" << ptr->phone_num << endl << endl;
    }
    else
    {
        int choice;
        cout << "No such entry is present in the directory!!! Would you
like to view all the entries present? \n1.Yes\n2.No" << endl;
        cin >> choice;
        if(choice == 1)
        {
            for(int i = 0; i < 26; i++)
            {
                displayItemsInIndex(i);
            }
        }
        else
        {
            exitFunction();
        }
    }
}

void HashTable::deletion()
{
    string name;
```

```
    cout<<"Enter the name of person whose details are to be deleted: ";
    cin >> name;
    int index = hashFunction(name);
    Details *temp1, *temp2, *ptr;

    // Case1 : Bucket is empty
    if (directory[index]->name == "empty" && directory[index]->phone_num
== -1)
    {
        cout << name << " was not found in the phone directory!!!" <<
endl;
    }


    // Case2 : Only 1 item in bucket and it matches the item to be
deleted
    else if(directory[index]->name == name && directory[index]->next ==
NULL)
    {
        directory[index]->name = "empty";
        directory[index]->phone_num = -1;
        cout << name << "'s details were removed from the phone directory
successfully..." << endl;
    }

    // Case3 : Match is located in the first item in the bucket but there
are more items in bucket
    else if(directory[index]->name == name)
    {
        ptr = directory[index];
        directory[index] = directory[index]->next;
        delete ptr;
        cout << name << "'s details were removed from the phone directory
successfully..." << endl;
    }

    // Case4 : Bucket contains items but first item is not a match ->
match found -> match not found
    else
    {
        temp1 = directory[index]->next;
        temp2 = directory[index];

        while (temp1 != NULL && temp1->name != name)
        {
            temp2 = temp1;
            temp1 = temp1->next;
```

```
        }

        // no match found
        if(temp1 == NULL)
        {
            cout << name << " was not found in the phone directory!!!" <<
endl;
        }
        // match found
        else
        {
            ptr = temp1;
            temp1 = temp1->next;
            temp2->next=temp1;

            delete ptr;
            cout << name << "'s details were removed from the phone
directory successfully..." << endl;
        }
    }
}

HashTable::~HashTable()
{
    for (int i=0; i<26; i++){
        Details* p = directory[i];
        while (directory[i]){
            directory[i] = directory[i]->next;
            delete p;
            p = directory[i];
        }
    }
    delete []directory;

}

void exitFunction()
{
    cout << "$$ -------------------------Project Made By:----------------
------- $$" << endl;
    cout << "--> Drishika Chauhan\n" << endl;
    cout << "Batch: B4";
    exit(0);
}
```

```
int main()
{
    cout
<<"|~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|~~~~~
~~~~~~~~~~~~~~~~|\n";
    cout <<"|---------------|---------------------------------------------
---|---------------------|\n";
    cout <<"|***************|**************** PHONE DIRECTORY
**************|*********************|\n";
    cout <<"|---------------|---------------------------------------------
---|---------------------|\n";
    cout
<<"|~~~~~~~~~~~~~~~|~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|~~~~~
~~~~~~~~~~~~~~~~|\n\n\n";

    Details d;
    HashTable h;
    int c;
    string name;
    int numberOfEntries, n;
    long long int phone_num;

    cout << "Enter number of records you want to enter: ";
    cin >> numberOfEntries;

    do
    {
        cout<<"\nEnter your choice which you want to execute:" << endl;
        cout<< "1: Adding a contact"<<endl;
        cout<< "2: Deleting a contact"<<endl;
        cout<< "3: Displaying contacts"<<endl;
        cout<< "4: Searching a contact by name"<<endl;
        cout<< "5: Modify name of the user"<<endl;
        cout<< "6: Modify phone number of the user"<<endl;
        cout<< "7: Exit" << endl << endl;
        cin >> c;

        switch(c)
        {
           case 1:
                if(numberOfEntries != 0)
                {
                    while(numberOfEntries--)
                    {
                        cout << "Enter the name and phone number
respectively: " << endl;
                        cin >> name >> phone_num;
```

```
                        h.insert(phone_num, name);
                    }
                    numberOfEntries = 0;
                }
                else
                {
                    cout << "Enter number of records you want to enter:
";
                    cin >> numberOfEntries;

                    while(numberOfEntries--)
                    {
                        cout << "Enter the name and phone number
respectively: " << endl;
                        cin >> name >> phone_num;
                        h.insert(phone_num, name);
                    }
                    numberOfEntries = 0;
                }
                break;

            case 2:
                h.deletion();
                break;

            case 3:
                cout << "Displaying Phone Directory: " << endl;
                for(int j = 0; j < 26; j++)
                {
                    h.displayItemsInIndex(j);
                }
                break;

            case 4:
                h.searchByName();
                break;

            case 5:
                h.modifyName();
                break;

            case 6:
                h.modifyPhoneNum();
                break;

            case 7:
                exitFunction();
```

```
                break;
            default:
                cout<< "Sorry could find the match you are looking for.
Enter the correct choice.";
        }
    } while(c);
    return 0;
}
```

THANK YOU!!! ☺