B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Course –Object oriented programming using C++
Course Code –18IS3PCOOP
AY 2020-21

Title:

PHONE REGISTRY USING LINKED LIST

Submitted to – Dr V Shubha Rao
Submitted by -

Jahnavi L	Meghashree B.B.
1BM19IS063	1BM19IS088

B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

INDEX

ABSTRACT

Being in contact with our family, friends, relatives, colleagues and other important people is necessary. To make this easy, all their contact information can be stored in one file where it's easily accessible.
In this project, we aim to build a phone registry using concepts of OOPS(C++) , Modular programming and File handling, where contacts of the people are stored as records . This report highlights the problem statement, system requirements, implementation and snapshots demonstrating the working model. The introduction part gives an insight into the problem definition and project description. This report firstly highlights the definitions of the OOPS concepts that we have used in this project followed by the implementation of the project. The working will be demonstrated using snapshots of the outputs of the program.

INTRODUCTION

A Book that gives a list of the names , addresses and telephone numbers of the people is called a phone directory. A phone directory is one of the most useful tools of information in communication. The traditional hard copy prints of the yellow and white pages have been in existence since 1878, and in the late 20th century telephone address books went online. From time to time we have come across many directories, whether in hard or soft print, but most people however overlook their importance due to lack of proper knowledge on their use and benefits. Hence we have tried to recreate the phone directory in our program “**Phone Registry**” using OOPs concepts in C++ . The difference between these two is that we can add, modify and delete a record in the phone registry but not in the phone directory.

Our project Telephone Registry is a program that enables users to easily store and retrieve contact information of an individual. The features of phone registry are adding, listing, searching, modifying, deleting records and also reversing the order of the records. All these operations are done through file handling, i.e all the added and modified data are recorded in a file called ”**Record**” (both text and csv format), and the deleted information is removed from the program file. Here the information of the people are linked using a data structure called **Linked list**. The phone registry contains Name, Address, Birth year and Phone number of a person.
The following operations can be performed in this program:-

- Add a new contact to the record book
- Modify a record using name as the key
- Print record details of a person using name as the key
- Print all the record details
- Delete an existing record using name as the key
- Reverse order of the record

OOPS CONCEPTS USED:

Classes and objects:

A class in C++ is the building block that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A C++ class is like a blueprint for an object. An object is an instance of a class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

The concepts of classes and objects are used in this project. The project contains a class called “listt”, which consists of various functions and variables necessary to implement the phone registry. The instance of the class is pushed to the contactList which is a list to drive the program into working. The methods used in the class allow us to feed the data to the linked list and modify. Delete, reverse and search are performed using the list operations.

Polymorphism:

In C++, polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function. In this project we use constructor overloading which is a type of function overloading and is a compile time polymorphism. We use constructor overloading to specify the file to be used for file manipulation or use a file by default if not specified.

Encapsulation and Abstraction:

In C++, encapsulation helps us keep related data and functions together, which makes our code cleaner and easy to read.It helps to control the modification of our data members. In this project we use encapsulation to create, keep, modify, delete the data related to each other inorder to frame a phone registry using suitable access specifiers.

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. fstream used in the file handling also comes under abstraction.

Operator overloading:

In C++, we can change the way operators work for user-defined types like objects and structures. This is known as operator overloading. We have used the stream insertion operator to print all the records existing in the phone registry.

Function overloading:

Function overloading is a feature in C++ where two or more functions can have the same name but different parameters. As mentioned earlier, constructor overloading is a type of function overloading and the same is discussed in the polymorphism section.

Friend function:

A friend function can be given a special grant to access private and protected members.In operator overloading, if an operator is overloaded as a member, then it must be a member of the class of the object on the left side of the operator.The operators '<<' and '>>' are called like 'cout << ob1' and 'cin >> ob1'. So if we want to make them a member method, then they must be made members of ostream and istream classes, which is not a good option most of the time. Hence we overload them as friend functions instead of member functions. The same concept has been used in the project to print all the records in the registry.

Files:

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it. We use files to store and manipulate the record data in the registry.

Modularity:

Modularity is an important OOPS concept. Modularity means the written program can be splitted up into modules by using classes and each class can be considered as a module. The program written for the project is modular, meaning the code is split into different modules for reusability and readability. The program contains header files where the definitions of structures, variables and functions with the name of the module are included. The implementation of these definitions are written in a different file. Finally, in the main file, header files are included and this file binds all the definitions and implementations to get the program work efficiently.

STL:

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized.The project is implemented using the list template library. The list is of the type “listt” which as a class was defined. Iterator, .begin(), .end(), .push_back(), .erase(), .reverse() are used to get the project working.

SOFTWARES USED:

The requirements necessary for this project are:

- Operating system - windows/linux/iOS
- Code editor - Visual Studio Code

CODE

list.h - class definition

```
#include<iostream>
class listt
{
   public:
      char name[25];
      char address[80];
      int yob;
      char number[15];
      void addRecord(char *name, char *number, char *address, int yob);
      void display();
      void modify(char *name, char *number, char *address);
      friend std::ostream&operator <<(std::ostream &out,const listt& );
};
```

ClassFunc.cpp - implementation of the methods of the class listt

```
#include <string.h>
#include "listt.h"
#include <iostream>
#include <fstream>
#include <list>
#include <iterator>
#include<iomanip>
using namespace std;

void listt::addRecord(char name[],char number[],char address[],int yob)
{
  strcpy(this->name,name);
  strcpy(this->number,number);
  strcpy(this->address,address);
  this->yob=yob;
}
void listt::display()
{
    cout<<"Name: "<<name<<endl<<endl;
    cout<<"Number: "<<number<<endl<<endl;
    cout<<"Address: "<<address<<endl<<endl;
    cout<<"YOB: "<<yob<<endl<<endl;
    cout << "---------------------------------------------------------------------------------------------------\n";


}

void listt::modify(char *name, char *number, char *address)
{
    strcpy(this->number,number);
    strcpy(this->address,address);
}

std::ostream&operator <<(std::ostream &out,const listt&l )
{
    out<<setw(10)<<l.name<<setw(15)<<l.number<<setw(50)<<l.address
    <<setw(15)<<l.yob<<endl;
    out<<endl;
    return out;
}
```

Files.h - class definition

```
#include<list>
#include<iostream>
#include"listt.h"
using namespace std;
class files
{
    public:
        char fileName[25];
        files();
        files(char f[]);
        void readfile(list<listt>&contactList);
        void writefile(list<listt>&contactList);
        void writefilecsv(list<listt>&contactList);
};
```

filesDef.cpp - implementation of the methods of the class files

```
#include<iostream>
#include<fstream>
#include<string.h>
#include<list>
#include<iterator>
#include "files.h"

using namespace std;

files::files()
{
   strcpy(fileName,"Record.txt");
}
files::files(char f[])
{
   strcpy(fileName,f);
}

void files::readfile(list<listt>&contactList)
{
   char name[25];
   char address[80];
   int yob;
   char number[15];

   ifstream file(fileName);

   char l;
   if(file.is_open())
   {
      if(file.get()!=EOF)
      {
         if(file.get()=='\n')
         {
            do
            {
               /* code */
               listt A;
               file.getline(name,25,'\n');
               file.getline(address,80);
```

```
                file>>yob;
                file.get(l);
                file.getline(number,15,'\n');
                A.addRecord(name,number,address,yob);
                contactList.push_back(A);
            } while (file.get()!=EOF);

        }
    }

  }

}
void files::writefile(list<listt>&contactList)
{

  ofstream file(fileName);

  if(file.is_open())
  {
     if(!contactList.empty())
     {
        file<<"\n";
        list <listt>::iterator i;
        i=contactList.begin();
        for(i=contactList.begin();i!=contactList.end();i++)
        {
           file<<"\n";
           file<<i->name<<endl;
           file<<i->address<<endl;
           file<<i->yob<<endl;
           file<<i->number<<endl;
        }

     }

  }
  file.close();
}
void files::writefilecsv(list<listt>&contactList)
{
    ofstream file("Record.csv");
    file<<"Name,Address,YOB,Number\n";
    if(file.is_open())
    {
        if(!contactList.empty())
        {
            // file<<"\n";
            list <listt>::iterator i;
            i=contactList.begin();
            for(i=contactList.begin();i!=contactList.end();i++)
            {
                file<<"\n";
                file<<i->name<<" ";
                file<<",";
                file<<i->address<<" ";
                file<<",";
                file<<i->yob<<",";
                file<<i->number;
            }

        }

    }
    file.close();
}
```

Main.cpp

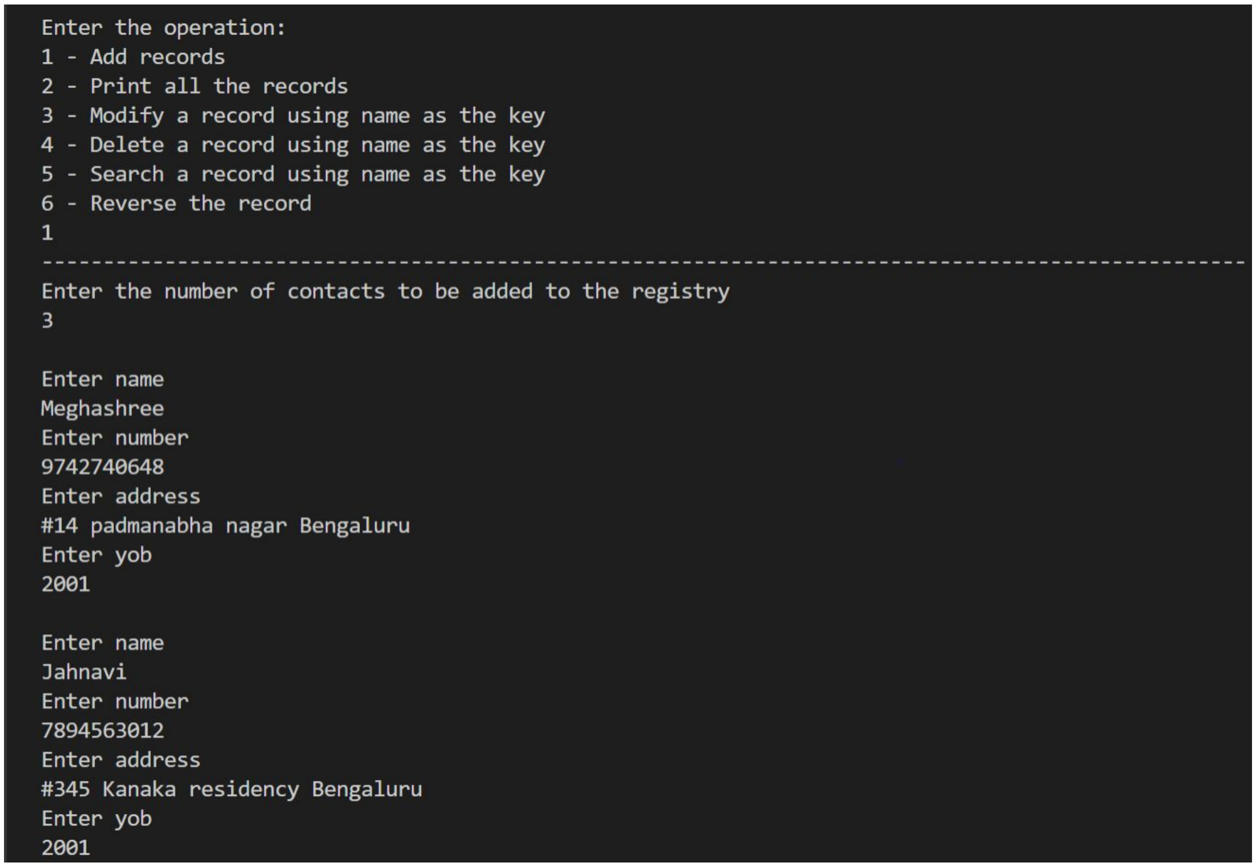
```
#include <iostream>
#include <fstream>
#include "files.h"
#include <list>
#include <string.h>
#include <iterator>
#include<iomanip>
using namespace std;
list<listt> contactList;
void Display()
{
    list<listt>::iterator i;
    cout<<"\n\n\n";
    cout<<setw(49)<<"PHONE REGISTRY\n\n";

    cout<<setw(6)<<"Name"<<setw(15)<<"Number"<<setw(40)<<"Address"
    <<setw(28)<<"YOB"<<endl;
    cout << "---------------------------------------------------------------------------------------------------\n";

    for (i = contactList.begin(); i != contactList.end(); i++)
    {

        // (*i).display();
        cout<<(*i);
    }
    cout << "---------------------------------------------------------------------------------------------------\n";



}
void add()
{
    int n, yob;
    char name[25], number[15], address[80];
    listt *A;
    A = new listt[100];
    list<listt>::iterator i;
        cout << "---------------------------------------------------------------------------------------------------\n";
    cout << "Enter the number of contacts to be added to the registry\n";
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cout << "Enter name\n";
        fflush(stdin);
        cin.getline(name, 25);
        cout << "Enter number\n";
        fflush(stdin);
        cin.getline(number, 15);
        cout << "Enter address\n";
        fflush(stdin);
        cin.getline(address, 80);
        cout << "Enter yob\n";
        cin >> yob;
        A[i].addRecord(name, number, address, yob);
        contactList.push_back(A[i]);
    }
}
void Modify()
{
    char name[25],number[15],address[80];
    list<listt>::iterator i;
    cout << "Enter the name whose record you want to modify\n";
    fflush(stdin);
    cin.getline(name, 25);

    for (i = contactList.begin(); i != contactList.end(); i++)
    {
        if (strcmp((*i).name, name) == 0)
        {
            cout<<"\nContact "<<name<<" found\n";
            cout<<"Enter the number \n";
            fflush(stdin);
            cin.getline(number,15);
            cout<<"Enter the address\n";
            fflush(stdin);
            cin.getline(address,80);
            (*i).modify(name, number, address);
            return;
        }
    }
    cout<<"\nContact "<<name<<" not found\n";
    return;

}
void Delete()
{
    char name[25];
    cout<<"Enter the name whose record you wish to delete\n";
    fflush(stdin);
    cin.getline(name,25);

    list<listt>::iterator i;
    for(i=contactList.begin();i!=contactList.end();i++)
    {
        if (strcmp((*i).name, name) == 0)
        {
            cout<<"\nContact "<<name<<" found\n";
            i=contactList.erase(i);
            cout<<"Contact "<<name<<" deleted\n";
            return;
        }

    }
    cout<<"\nContact "<<name<<" not found\n";
    return;
}
void search()
{
    char name[25];
    cout<<"Enter the name whose record you wish to search\n";
    fflush(stdin);
    cin.getline(name,25);

    list<listt>::iterator i;
    for(i=contactList.begin();i!=contactList.end();i++)
    {
        if (strcmp((*i).name, name) == 0)
        {
            cout<<"\nContact "<<name<<" found\n\n";
            (*i).display();
            return;
        }

    }
    cout<<"\nContact "<<name<<" not found\n";
    return;

}
void Reverse()
{
    contactList.reverse();
    return;
}

int main()
{
    int c, op;
    files f1;
    f1.readfile(contactList);
```

```
do
{
   cout << "\nEnter the operation:\n";
   cout << "1 - Add records\n";
   cout << "2 - Print all the records\n";
   cout <<"3 - Modify a record using name as the key\n";
   cout <<"4 - Delete a record using name as the key\n";
   cout <<"5 - Search a record using name as the key\n";
   cout <<"6 - Reverse the record\n";
   cin >> op;
   switch (op)
   {
   case 1:
      add();
      Display();
      break;

   case 2:
      Display();
      break;

   case 3:
      Modify();
      Display();
      break;

   case 4:
      Delete();
      Display();
      break;

   case 5:
      search();
      break;

   case 6:
      Reverse();
      Display();
      break;

   default:
      break;
```

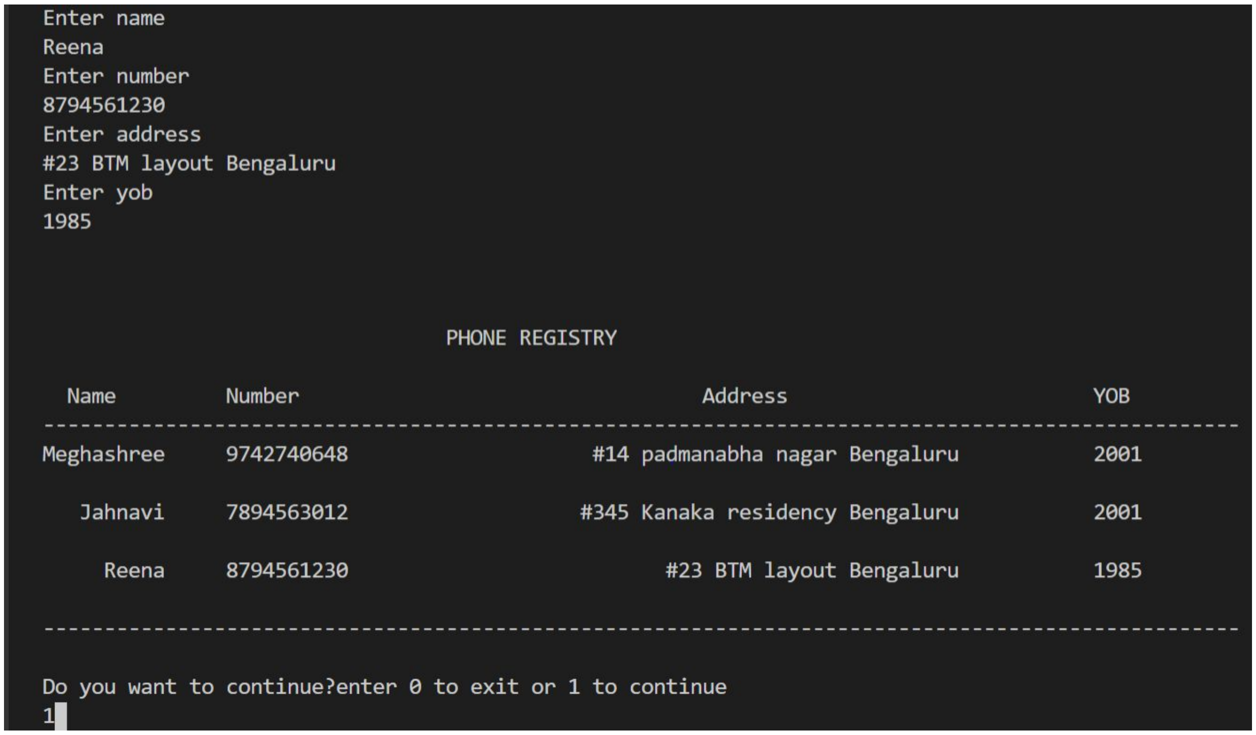
```
        }
        cout << "\nDo you want to continue?enter 0 to exit or 1 to continue\n";
        cin >> c;
    } while (c);

    f1.writefile(contactList);
    char c1[25];
    strcpy(c1,"Record.csv");
    files f2(c1);
    f2.writefilecsv(contactList);
    return 0;
}
```

SNAPSHOTS:

1. **Adding a new contact to the record book.**

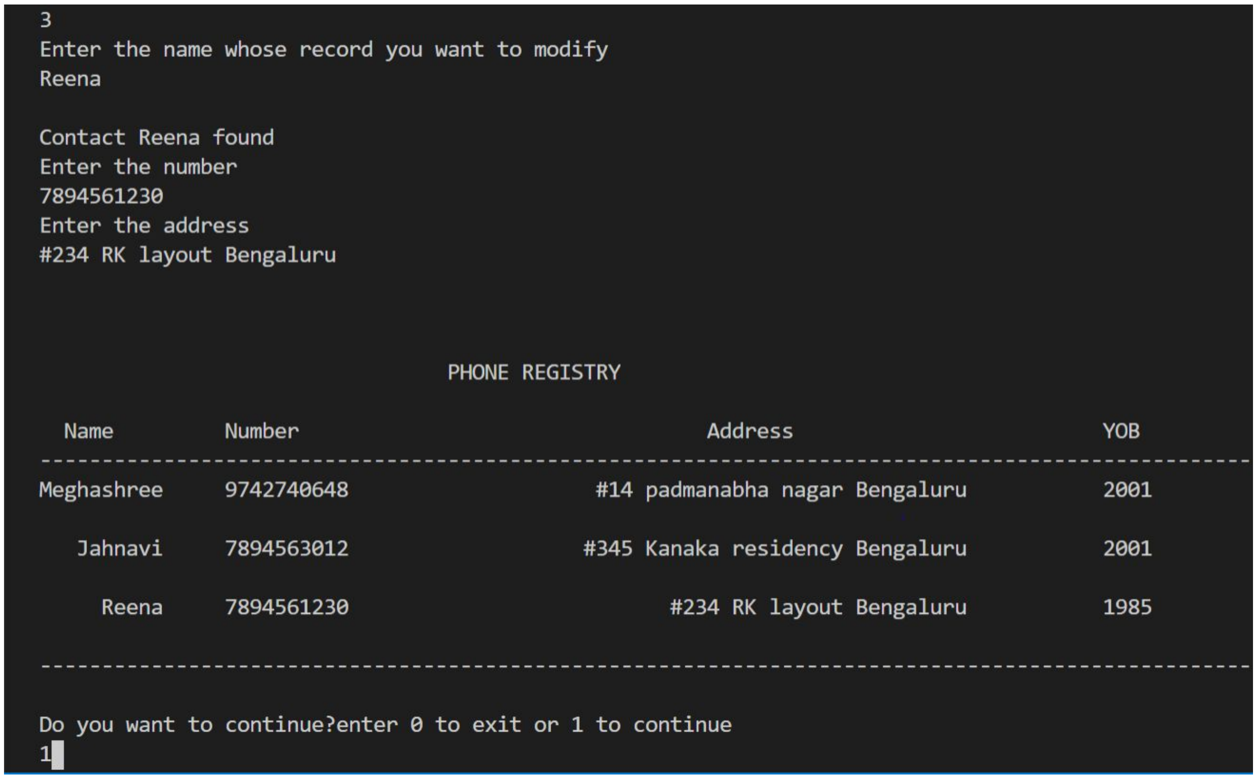


```
Enter the operation:
1 - Add records
2 - Print all the records
3 - Modify a record using name as the key
4 - Delete a record using name as the key
5 - Search a record using name as the key
6 - Reverse the record
1
-------------------------------------------------------------------------------------------
Enter the number of contacts to be added to the registry
3

Enter name
Meghashree
Enter number
9742740648
Enter address
#14 padmanabha nagar Bengaluru
Enter yob
2001

Enter name
Jahnavi
Enter number
7894563012
Enter address
#345 Kanaka residency Bengaluru
Enter yob
2001
```



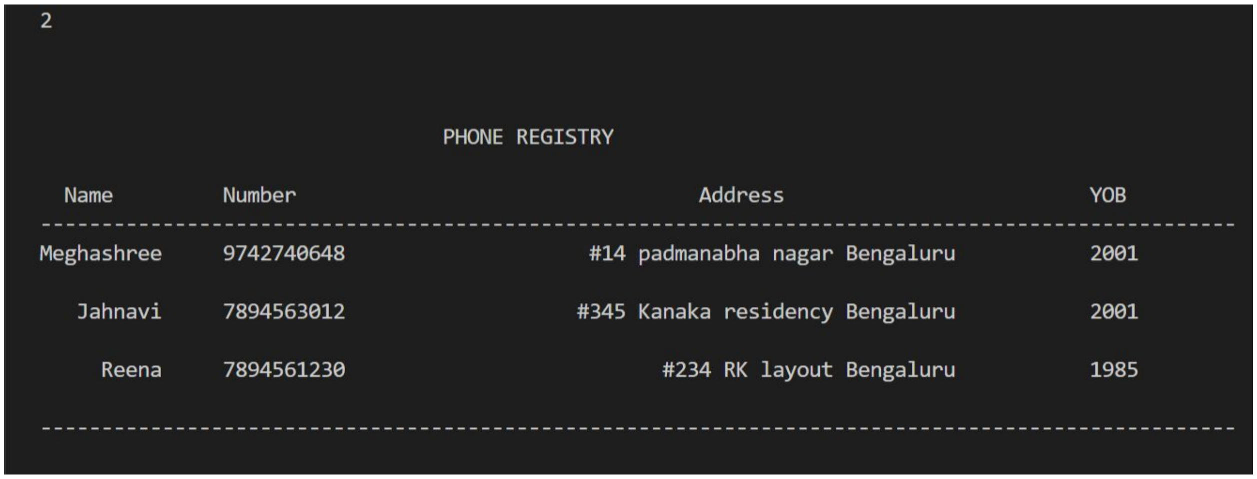
```
Enter name
Reena
Enter number
8794561230
Enter address
#23 BTM layout Bengaluru
Enter yob
1985



                                 PHONE REGISTRY

  Name          Number                                  Address                         YOB
------------------------------------------------------------------------------------------------
Meghashree     9742740648                    #14 padmanabha nagar Bengaluru            2001

   Jahnavi     7894563012                   #345 Kanaka residency Bengaluru            2001

     Reena     8794561230                          #23 BTM layout Bengaluru            1985

------------------------------------------------------------------------------------------------

Do you want to continue?enter 0 to exit or 1 to continue
1
```

2. **Modify a record using name as the key.**



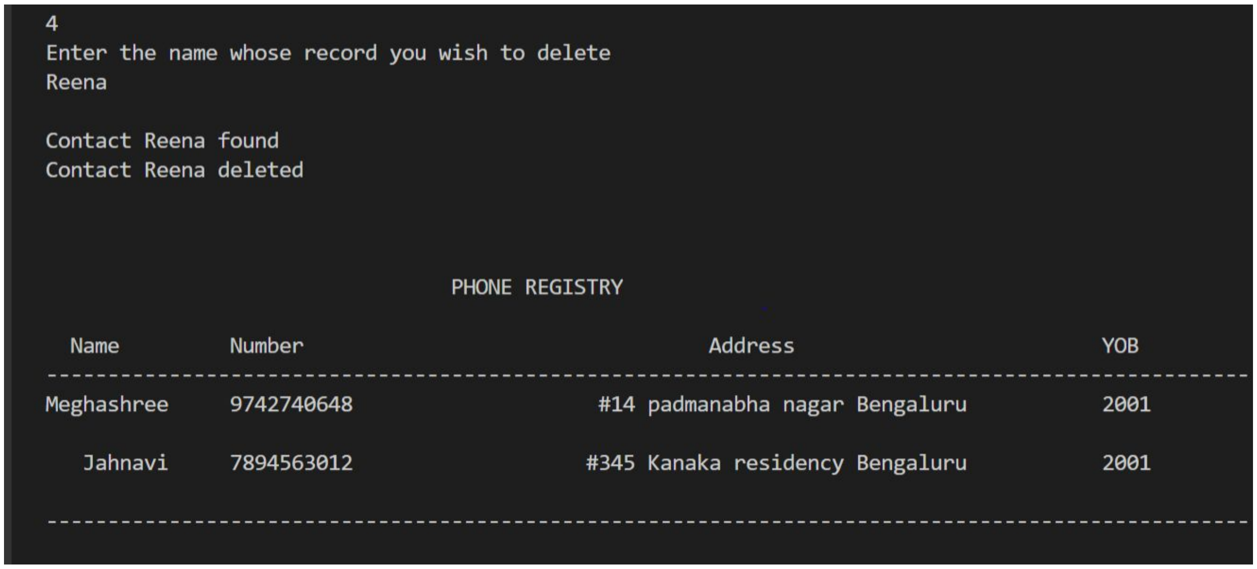
```
3
Enter the name whose record you want to modify
Reena

Contact Reena found
Enter the number
7894561230
Enter the address
#234 RK layout Bengaluru



                                 PHONE REGISTRY

  Name          Number                                  Address                         YOB
------------------------------------------------------------------------------------------------
Meghashree     9742740648                    #14 padmanabha nagar Bengaluru            2001

   Jahnavi     7894563012                   #345 Kanaka residency Bengaluru            2001

     Reena     7894561230                          #234 RK layout Bengaluru            1985

------------------------------------------------------------------------------------------------

Do you want to continue?enter 0 to exit or 1 to continue
1
```

3. **Print all contacts in the phone registry.**



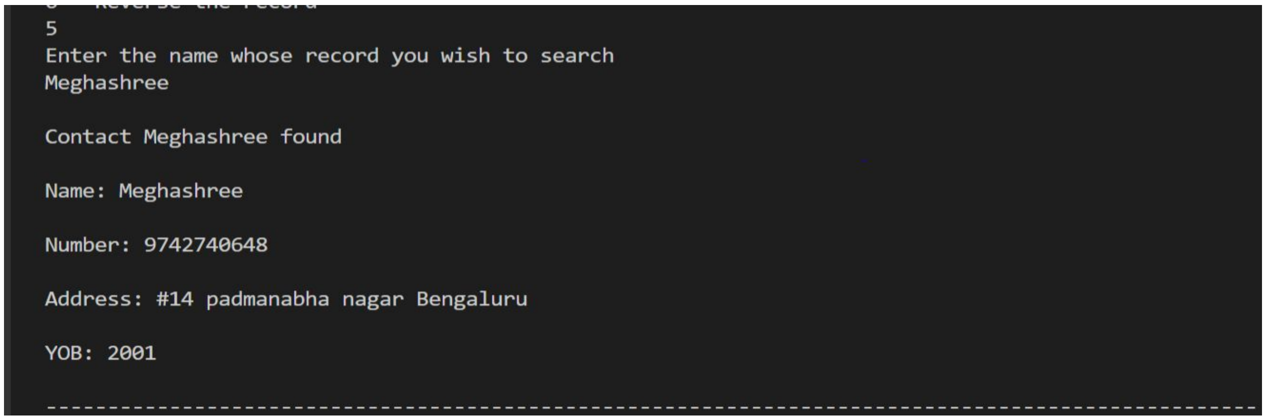
```
2



                                PHONE REGISTRY

  Name          Number                                   Address                         YOB
-----------------------------------------------------------------------------------------------------
Meghashree      9742740648                 #14 padmanabha nagar Bengaluru               2001

   Jahnavi      7894563012                #345 Kanaka residency Bengaluru               2001

     Reena      7894561230                       #234 RK layout Bengaluru               1985

-----------------------------------------------------------------------------------------------------
```

4. **Delete a contact using name as the key.**



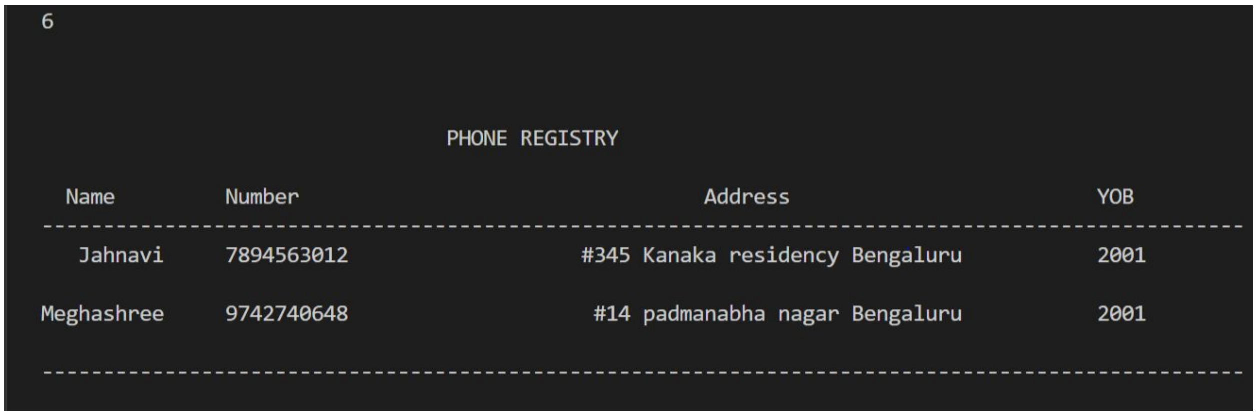
```
4
Enter the name whose record you wish to delete
Reena

Contact Reena found
Contact Reena deleted



                                PHONE REGISTRY

  Name          Number                                   Address                         YOB
-----------------------------------------------------------------------------------------------------
Meghashree      9742740648                 #14 padmanabha nagar Bengaluru               2001

   Jahnavi      7894563012                #345 Kanaka residency Bengaluru               2001

-----------------------------------------------------------------------------------------------------
```

5. **Search a contact using name as the key.**



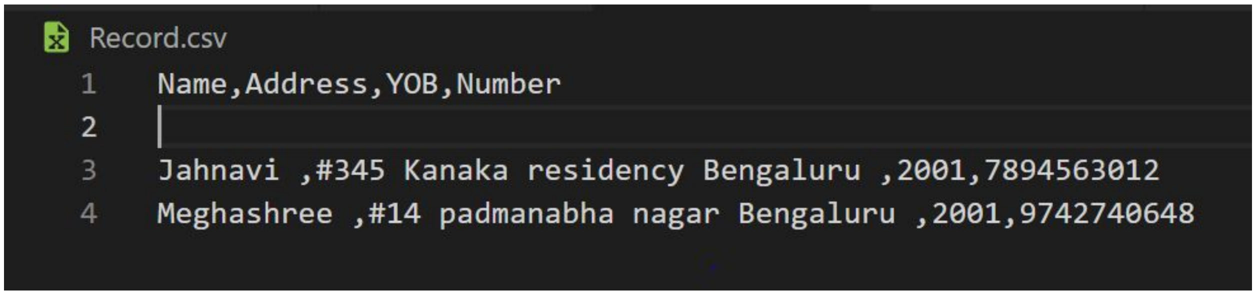
```
6   Reverse the record
5
Enter the name whose record you wish to search
Meghashree

Contact Meghashree found

Name: Meghashree

Number: 9742740648

Address: #14 padmanabha nagar Bengaluru

YOB: 2001

---------------------------------------------------------------------------------------------------
```

6. **Reverse the order of the records in the phone registry.**



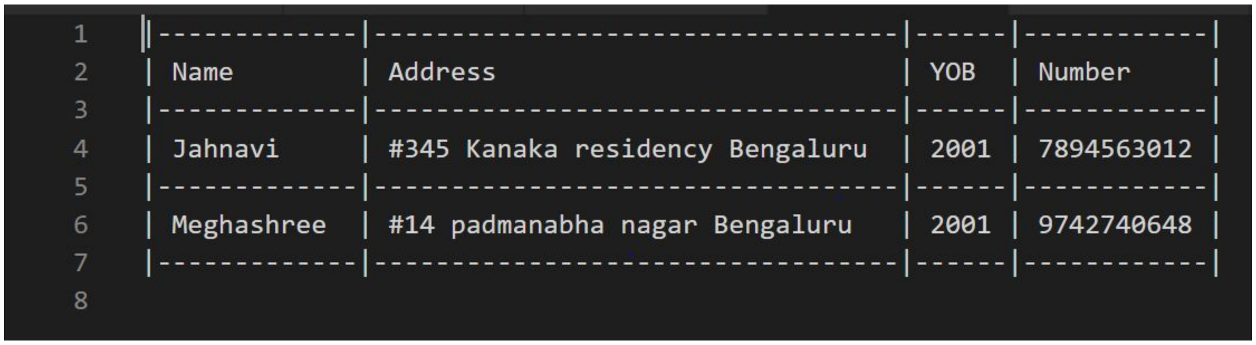
```
6



                                PHONE REGISTRY

  Name          Number                                  Address                            YOB
---------------------------------------------------------------------------------------------------
   Jahnavi      7894563012                    #345 Kanaka residency Bengaluru              2001

Meghashree      9742740648                     #14 padmanabha nagar Bengaluru              2001

---------------------------------------------------------------------------------------------------
```

7. **Record.csv generated.**

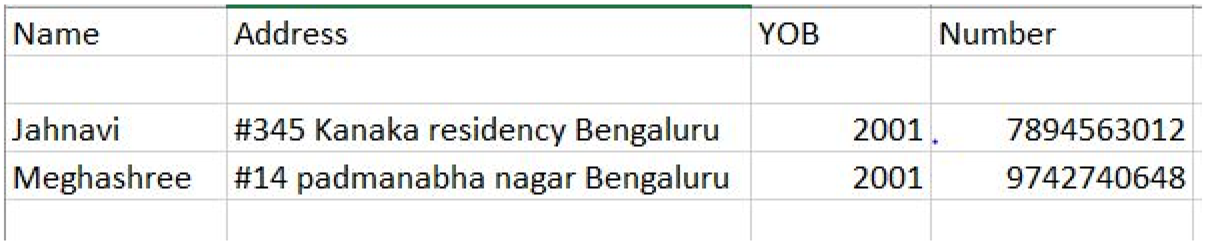


```
Record.csv
Name,Address,YOB,Number

Jahnavi ,#345 Kanaka residency Bengaluru ,2001,7894563012
Meghashree ,#14 padmanabha nagar Bengaluru ,2001,9742740648
```

8. Record.csv in the tabular format

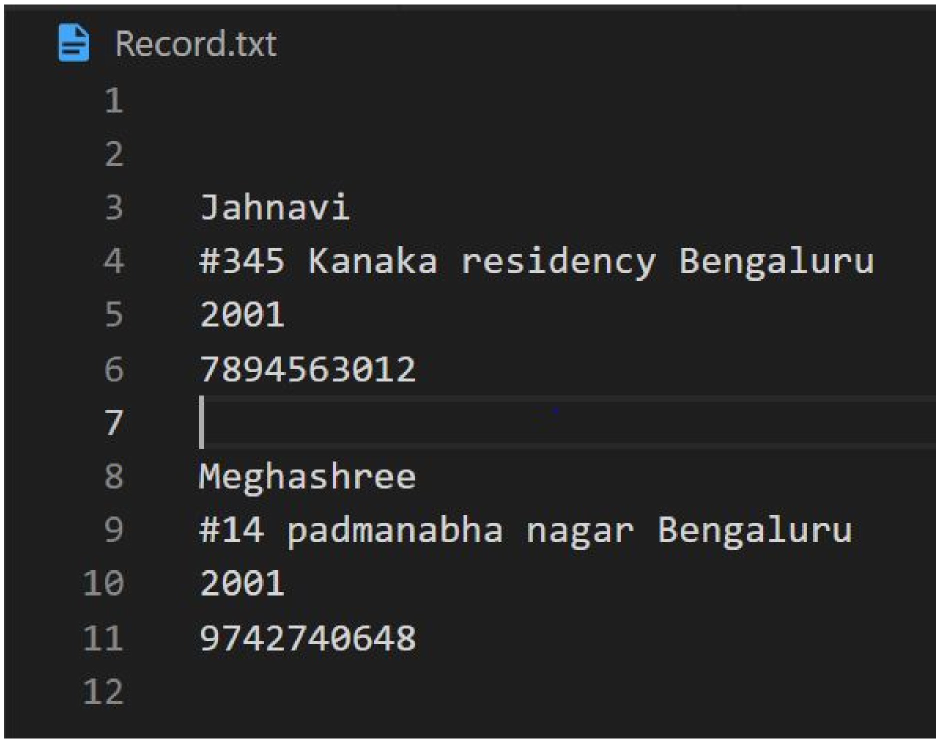


```
|------------|---------------------------------|------|------------|
| Name       | Address                         | YOB  | Number     |
|------------|---------------------------------|------|------------|
| Jahnavi    | #345 Kanaka residency Bengaluru | 2001 | 7894563012 |
|------------|---------------------------------|------|------------|
| Meghashree | #14 padmanabha nagar Bengaluru  | 2001 | 9742740648 |
|------------|---------------------------------|------|------------|
```



Name	Address	YOB	Number
Jahnavi	#345 Kanaka residency Bengaluru	2001	7894563012
Meghashree	#14 padmanabha nagar Bengaluru	2001	9742740648

9. Record.txt generated.



Record.txt

```


Jahnavi
#345 Kanaka residency Bengaluru
2001
7894563012

Meghashree
#14 padmanabha nagar Bengaluru
2001
9742740648

```

REFERENCES:

https://www.geeksforgeeks.org/file-handling-c-classes/

https://www.w3schools.com/cpp/cpp_files.asp