**Department of Computer Science**

**Islamia University Bahawalpur.**

**(Project)**

Project Title: **Phone Directory Application.**

**Subject:** **Data Structures & Algorithms.**

**Submitted by:** **Muhammad Huzaifa And Rehan Raza.**

**Submitted to**: **Mr. Muhammad Usman Ghani.**

**Roll No:** **S23BDOCS1M01143 or S23BDOCS1M01145**.

**Section:** **BS-CS(2M)**

First, let's discuss my project, which is implemented in C++. I'll outline the steps involved to explain it more effectively.

In this we use doubly linked list for efficient and better application and data flow for my Phone directory Application.

I also add a small database of Document for save contact for see outside the application.

1. Contact.h:

Initially, I created a contact header file to facilitate the creation of multiple classes, which made the program more manageable and modular.



(#ifndef CONTACT\_H

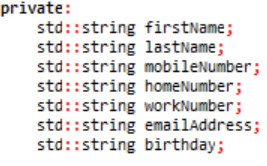
#define CONTACT\_H)

* These lines are header guards, which prevent the contents of the header file from being included more than once in the same compilation unit. If CONTACT\_H is not defined, the contents of the header file will be processed otherwise, they will be skipped.

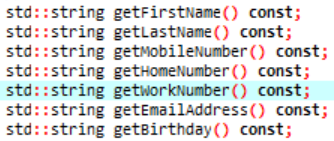
In above code we Initializes a **Contact** object with details.

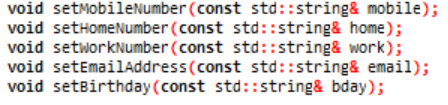
#include <string>

* This line includes the **<string>** standard library header, which is necessary since the **Contact** class uses **std::string** for storing string data.



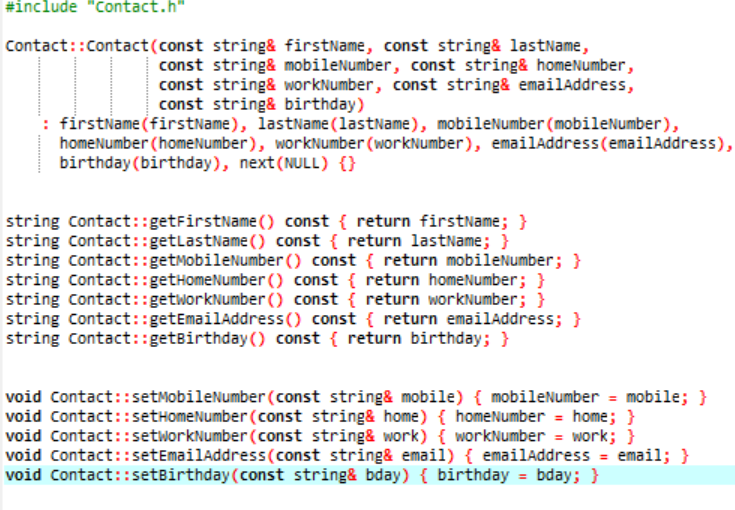
This defines the Contact class with private member variables representing different attributes of a contact: first name, last name, mobile number, home number, work number, email address, and birthday. These member variables are all of type std::string.





* **Getter and Setter Functions:** Provide access to private member variables and allow modification of the contact's details.

1. Contact Function.cpp:



**Constructor:**

* + - In this above code **firstName(firstName)**: Initializes the **firstName** member variable with the **firstName** parameter.
    - **lastName(lastName)**: Initializes the **lastName** member variable with the **lastName** parameter.
    - **mobileNumber(mobileNumber)**: Initializes the **mobileNumber** member variable with the **mobileNumber** parameter.
    - **homeNumber(homeNumber)**: Initializes the **homeNumber** member variable with the **homeNumber** parameter.
    - **workNumber(workNumber)**: Initializes the **workNumber** member variable with the **workNumber** parameter.
    - **emailAddress(emailAddress)**: Initializes the **emailAddress** member variable with the **emailAddress** parameter.
    - **birthday(birthday)**: Initializes the **birthday** member variable with the **birthday** parameter.
    - **next(NULL)**: Initializes the **next** pointer to **NULL**.

**Getter Function:**

These are the definitions of the getter functions declared in the header file. Each function returns the value of the corresponding private member variable. They are marked const because they do not modify the object.

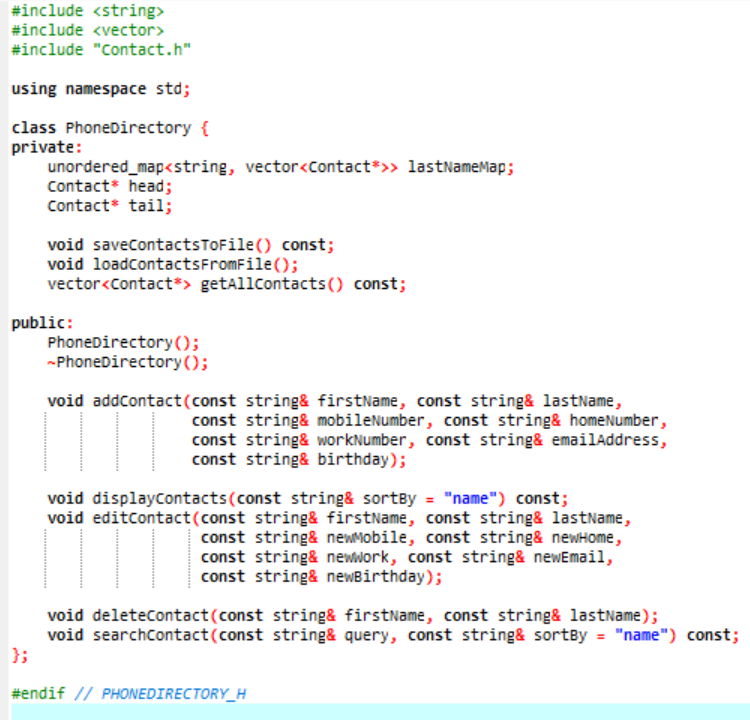
* + - getFirstName(): Returns the firstName of the contact.
    - getLastName(): Returns the lastName of the contact.
    - getMobileNumber(): Returns the mobileNumber of the contact.
    - getHomeNumber(): Returns the homeNumber of the contact.
    - getWorkNumber(): Returns the workNumber of the contact.
    - getEmailAddress(): Returns the emailAddress of the contact.
    - getBirthday(): Returns the birthday of the contact.

**Setter Functions:**

These are the definitions of the setter functions declared in the header file. Each function sets the value of the corresponding private member variable to the value provided as a parameter.

* setMobileNumber(const string& mobile): Sets the mobileNumber of the contact to the given mobile string.
* setHomeNumber(const string& home): Sets the homeNumber of the contact to the given home string.
* setWorkNumber(const string& work): Sets the workNumber of the contact to the given work string.
* setEmailAddress(const string& email): Sets the emailAddress of the contact to the given email string.
* setBirthday(const string& bday): Sets the birthday of the contact to the given bday string.

1. **Phone Directory Header.h:**

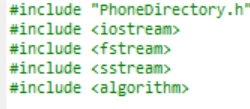


These are the public member functions of the **PhoneDirectory** class:

* **PhoneDirectory()**: The constructor. Initializes a new **PhoneDirectory** object.
* **void addContact(...)**: Adds a new contact to the directory with the provided details (first name, last name, mobile number, home number, work number, email address, birthday).
* **void displayContacts(const string& sortBy = "") const**: Displays all contacts. The optional **sortBy** parameter can specify a sorting criterion (e.g., by last name, first name, etc.). It's marked **const** because it does not modify the **PhoneDirectory** object.
* **void editContact(...)**: Edits an existing contact identified by first name and last name with new details (new mobile number, new home number, new work number, new email, new birthday).
* **void deleteContact(const string& firstName, const string& lastName)**: Deletes a contact identified by first name and last name.
* **void searchContact(const string& query, const string& sortBy = "") const**: Searches for contacts that match the query string and optionally sorts the results by the specified criterion. It's marked **const** because it does not modify the **PhoneDirectory** object.
* **~PhoneDirectory()**: The destructor. Cleans up any resources used by the **PhoneDirectory** object.

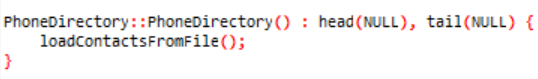
1. Phone Directory Function.cpp:

**Header Includes:**

****

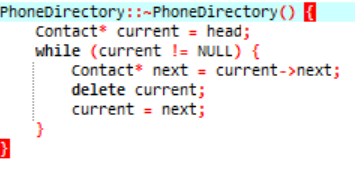
**Constructor**

* This is the constructor for the **PhoneDirectory** class. It initializes the **head** pointer to **NULL**, indicating that the directory starts empty with no contacts.



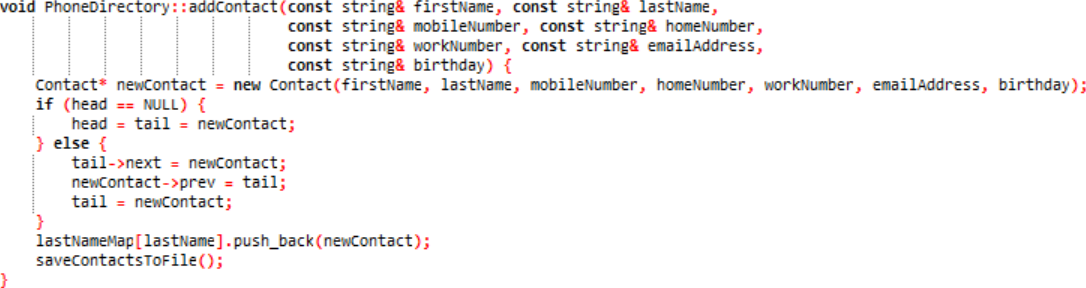
**Destructor:**

* This is the destructor of the PhoneDirectory class. It deallocates memory by deleting all contacts in the linked list.



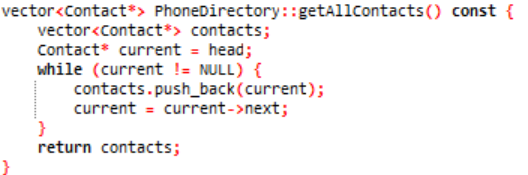
**Add Contact Method**:

* This method adds a new contact to the phone directory. It creates a new Contact object, updates the linked list pointers, adds the contact to the last name map, and saves contacts to a file.



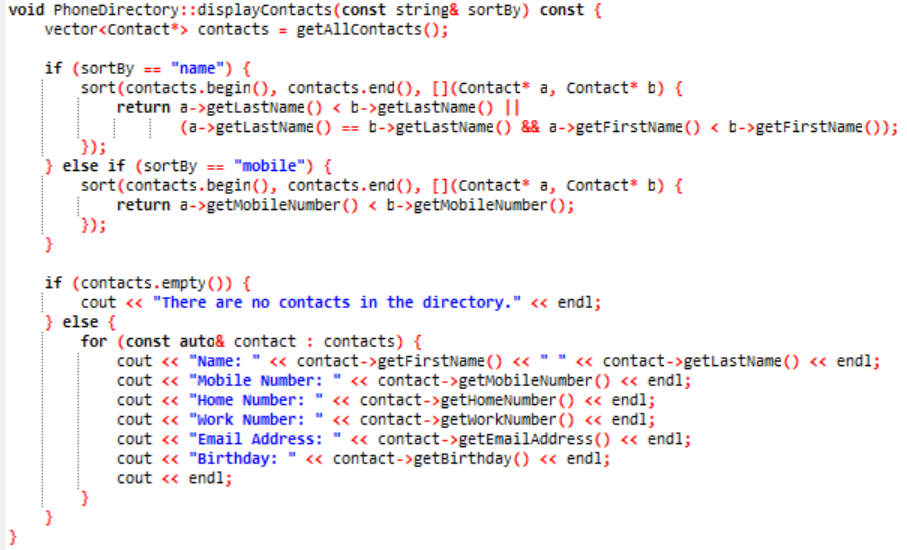
**Get All Contacts Method:**

* This method retrieves all contacts from the phone directory and returns them as a vector of pointers to Contact objects.



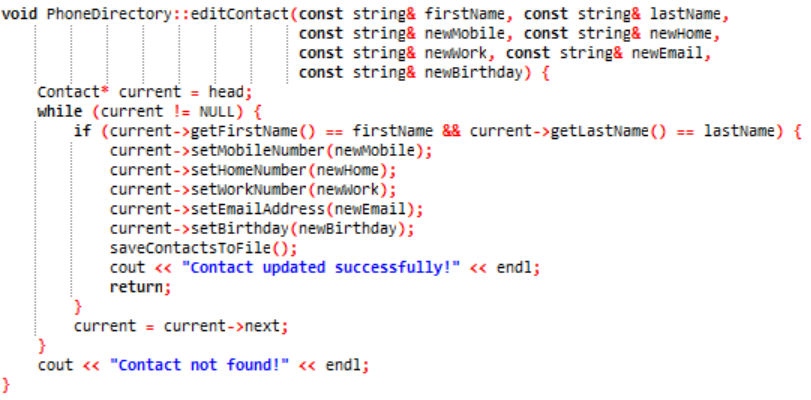
**Display Contacts Method:**

* This method displays all contacts in the phone directory, optionally sorted by name or mobile number.



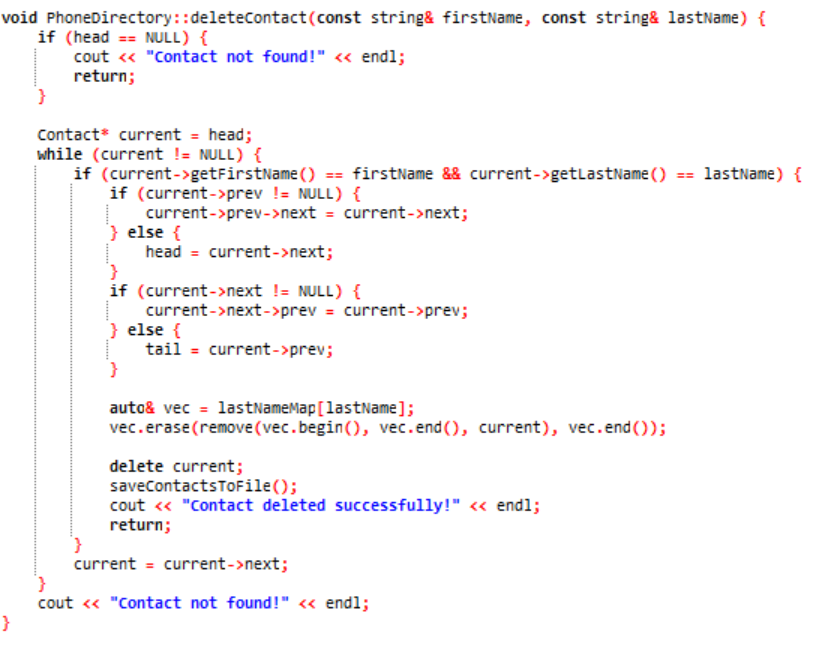
**Edit Contact Method:**

* This method edits the information of an existing contact in the phone directory.



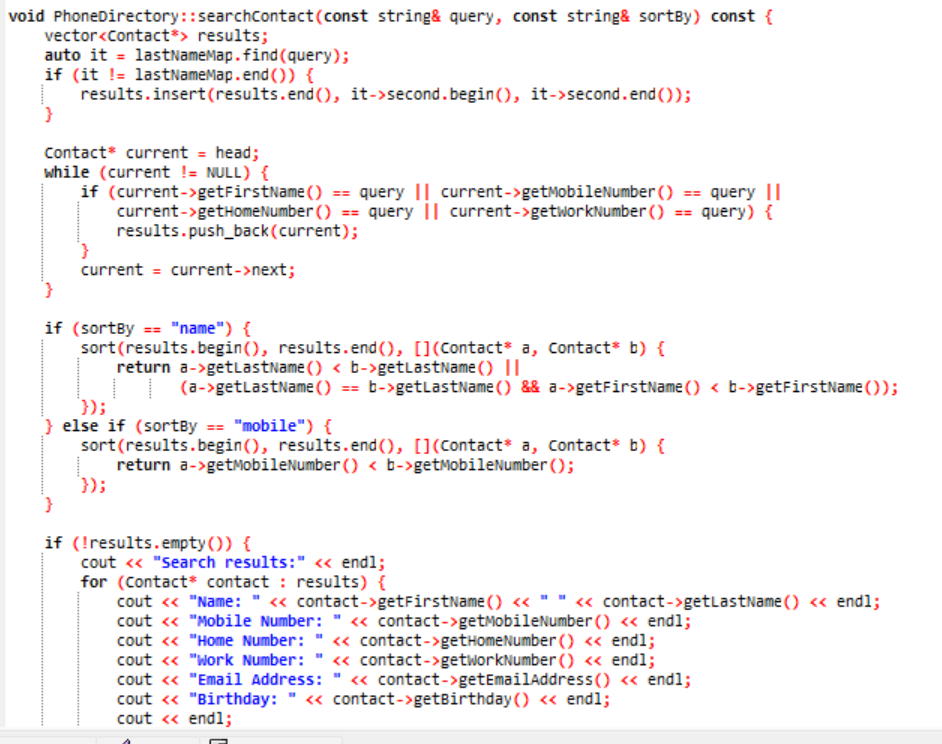
**Delete Contact Method:**

* This method deletes a contact from the phone directory based on the first name and last name.



**Search Contact Method:**

* This method searches for a contact in the phone directory based on the query string and optionally sorts the results.

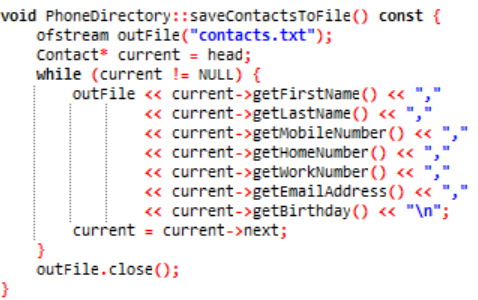


**Save Contacts to File Method:**

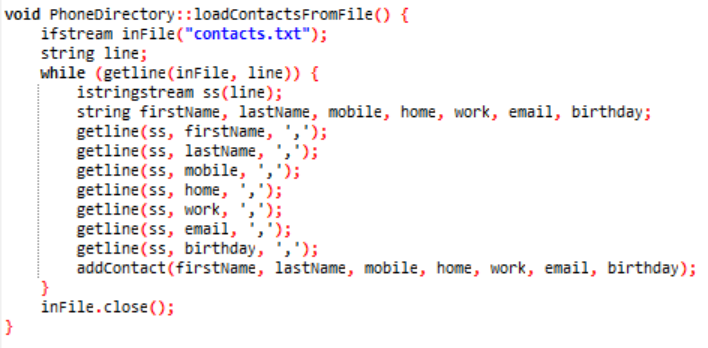
* This method saves all contacts in the phone directory to a file.

Load Contacts from File Method:

This method loads contacts from a file into the phone directory.



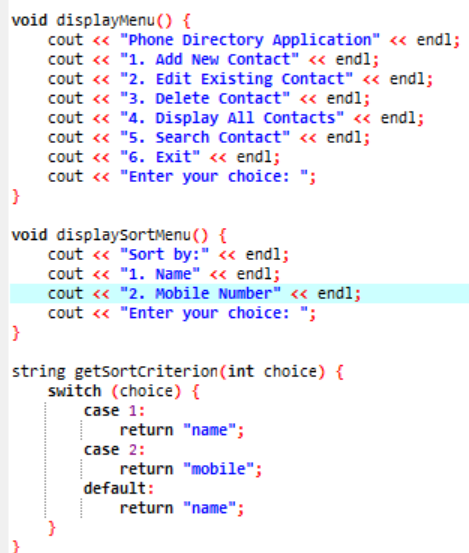
**Load Contact from file:**

****

**5.Main.cpp:**

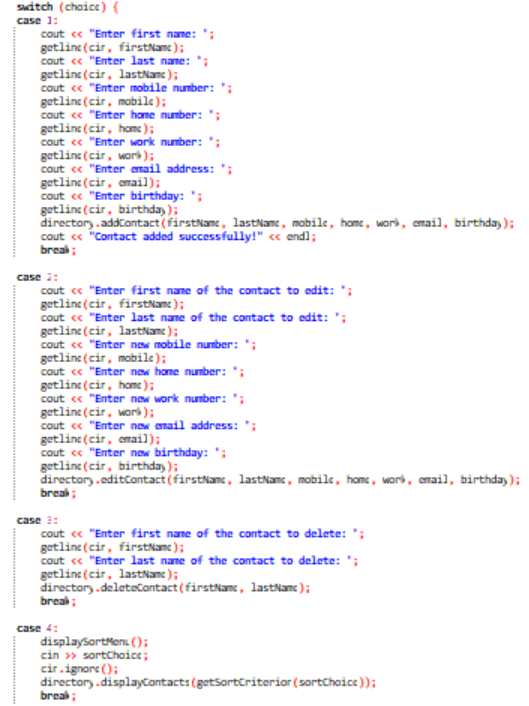
**Function Declarations:**

* displayMenu(): Displays the main menu options for the phone directory application.
* displaySortMenu(): Displays the sort options for displaying contacts.
* getSortCriterion(int choice): Returns the sort criterion based on the user's choice.



**switch (choice) :**

* The user's choice is processed using a switch statement, where each case corresponds to a menu option.



**Menu Options:**

* Add New Contact: Prompts the user to input contact details and adds a new contact to the phone directory.
* Edit Existing Contact: Allows the user to edit the details of an existing contact.
* Delete Contact: Enables the user to delete a contact from the phone directory.
* Display All Contacts: Displays all contacts in the phone directory, sorted based on the user's choice.
* Search Contact: Allows the user to search for a contact by name or phone number, displaying the results sorted according to the user's choice.
* Exit: Terminates the program.

**Input/Output:**

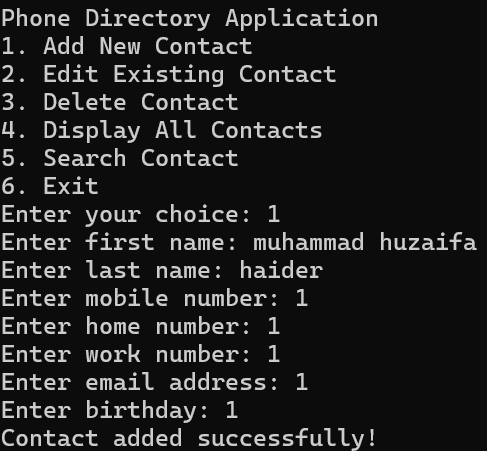
* The program uses cin and cout for user input and output, respectively.

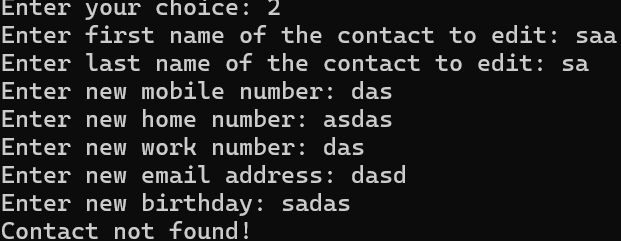
getline(cin, variable) is used to read input lines from the console.

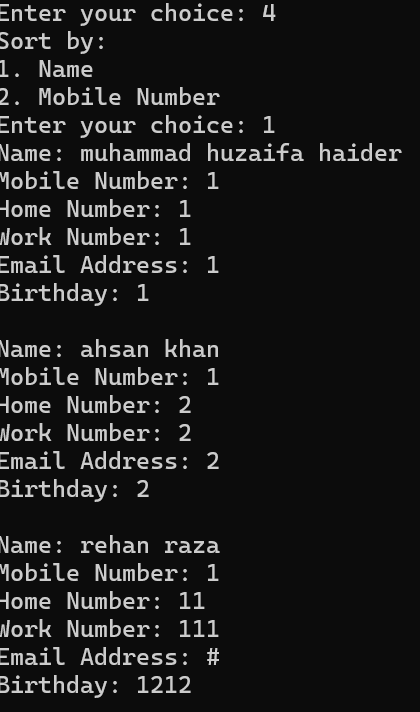
Function Calls:

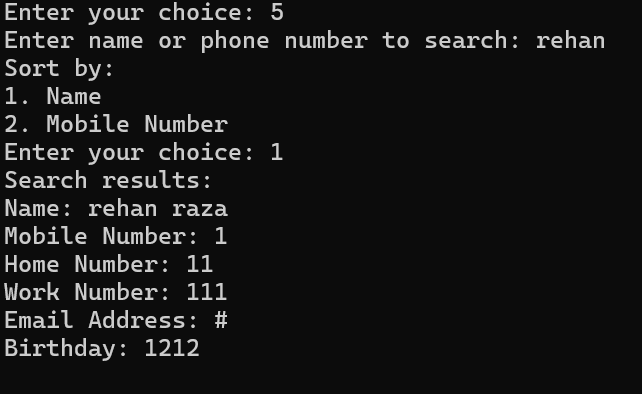
* Methods of the PhoneDirectory class are called to perform operations such as adding, editing, deleting, displaying, and searching contacts.

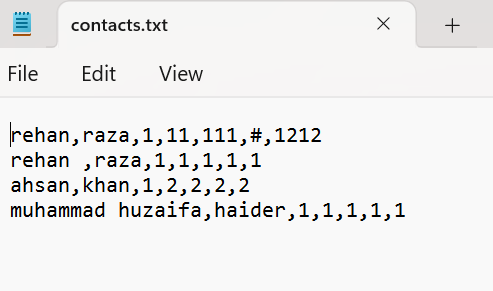
**Output:**

****

****

****

****

****