Contact Book Application

The Contact Book app is created to give users a strong resource for organising their contacts efficiently. Its main goal is to provide capabilities like saving, listing, removing, finding, and sorting contacts. These abilities are designed to improve the ease of access to contact details.

Execution Instructions

1. Setup: Before running the Contact Book application, ensure Python is installed on your system. Download or clone the application code from the provided repository.
2. Running the Application:
    a. Clone the GitHub repository.
    b. Open a Jupyter Notebook environment on your local machine or a cloud-based platform like Google Colab, Anaconda Cloud etc.
    c. Import the Jupyter Notebook.
    d. Execute the notebook cells to interact with the Contact Book.

Design of Data Structure and Algorithms

Data Structures

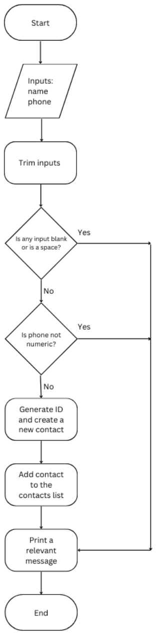
1. Contact Information: Each contact is stored as a dictionary with the following keys:

    a. ID: A string representing the contact's unique identifier.
    b. name: A string representing the contact's name.
    c. phone: A string representing the contact's phone number.

2. Contact Book: The contact book itself is a list of dictionaries, where each dictionary represents a contact.

Algorithms

1. **Insertion**:
    a. Data Structure: A new contact is added to the list of contacts.
    b. Algorithm:
        i. Input: name, phone
        ii. Trim inputs and check if none of the inputs are blank or are a space. If yes, print a relevant message.
        iii. Check if the phone is numeric. If not, print a relevant message.
        iv. Generate an ID and create a new dictionary with the contact information.
        v. Append the dictionary to the contacts list. Print a relevant message.
    c. Flowchart:


Start
Inputs:
name
phone
Trim inputs
Is any input blank or is a space?
Yes
No
Is phone not numeric?
Yes
No
Generate ID and create a new contact
Add contact to the contacts list
Print a relevant message
End

2. **Listing**:

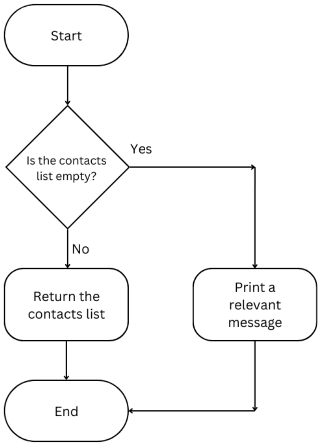
    a. Data Structure: List all contacts.

    b. Algorithm:

        i. If the contacts list is empty, print a relevant message.

        ii. Otherwise, print all contact details.

    c. Flowchart:



3. **Deletion**:

   a. Data Structure: Remove a contact from the list based on the ID.

   b. Algorithm:

      i. Input: ID

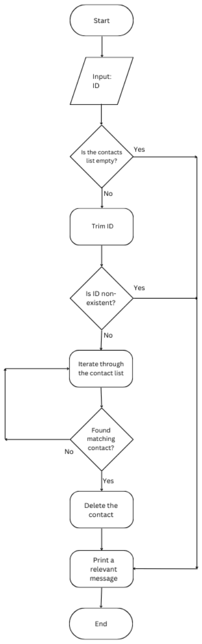
      ii. Check if the contacts list is empty. If it is, print a relevant message.

      iii. Trim ID and check if it is existent. If not, print a relevant message.

      iv. Iterate through the contacts list.

      v. If a contact with the matching ID is found, remove it from the list. Print a relevant message.

   c. Flowchart:



4. **Searching**:

    a. Data Structure: Search for a contact by ID, name or phone.

    b. Algorithm:

        i. Input: query

        ii. Check if the contacts list is empty. If it is, print a relevant message.

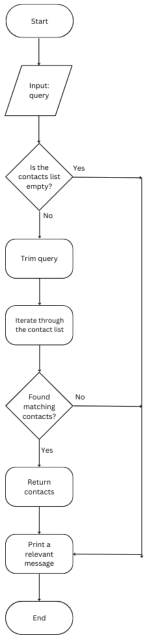
        iii. Trim query.

        iv. Iterate through the contacts list.

        v. If the query matches the contact ID or name or phone, return the contact information.

        vi. If no contacts are found, print a relevant message.

    c. Flowchart:



5. **Sorting**:

    a. Data Structure: The list of contacts will be sorted based on the contact name.

    b. Algorithm:

        i. Input: order

        ii. Check if the contacts list is empty. If it is, print a relevant message.

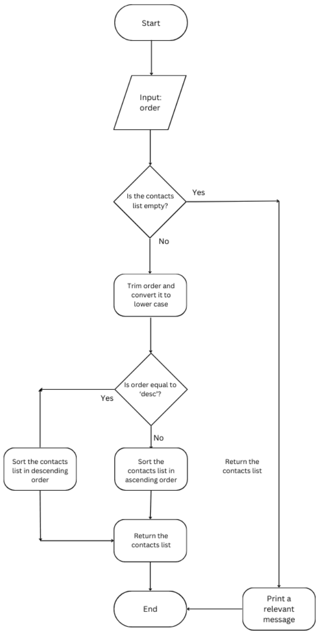
        iii. Trim the order and convert it to lowercase.

        iv. Check if the order is equal to ‘desc’. If it is, sort the contacts list by descending order of the contact name.

        v. Otherwise, sort the contacts list by ascending order of the contact name.

        vi. Print the sorted contacts list.

c. Flowchart:


Start
Input: order
Is the contacts list empty?
Yes
No
Trim order and convert it to lower case
Is order equal to 'desc'?
Yes
No
Sort the contacts list in descending order
Sort the contacts list in ascending order
Return the contacts list
Return the contacts list
End
Print a relevant message

Test Plan

Checkpoints

1. **Insertion**:
    a. Test adding a new contact with name and phone.
    b. Test adding a new contact with an invalid name and/or phone.
2. **Listing**:
    i. Test listing all contacts.
    ii. Test listing when there are no contacts.
3. **Deletion**:

   a. Test deleting a contact by ID.
   b. Test deleting a non-existent contact.
   c. Test deleting a contact by ID when the contacts list is empty.

4. **Searching**:
   a. Test searching for an existing contact by ID, name, and phone.
   b. Test searching for a non-existent contact.
   c. Test searching when the contacts list is empty.
5. **Sorting**:
   a. Test sorting the contacts list by default order.
   b. Test sorting the contacts list by descending order.
   c. Test sorting when the contacts list is empty.

Test Cases

ID	**Description**	**Input Data**	**Expected Results**

01	Verify contact creation with a name and phone.	name='Joey'
		phone='9595959595'
			A new contact should be added and a relevant message should be printed.

02	Verify contact creation with a blank name and blank phone.	name=''
		phone=''
			A relevant message should be printed.

03	Verify contact creation with a blank name.	name=”
		phone=’8080808080’
			A relevant message should be printed.

04	Verify contact creation with a blank phone.	name=’Sam’
		phone=”
			A relevant message should be printed.

05	Verify contact creation by passing a space in name and phone.	name=’ ’
		phone=’ ’
			A relevant message should be printed.

06	Verify contact creation by passing a space in the name.	name=’ ’
		phone=’7070707070’
			A relevant message should be printed.

07	Verify contact creation by passing a space in the phone.	name=’Ron’
		phone=’ ’
			A relevant message should be printed.

08	Verify contact creation with a non-numeric phone.	name=’Jack’
		phone=’3!30303e30’
			A relevant message should be printed.

09	Verify the listing of contacts when the contacts list is not empty.	-	The contacts list should be returned.
10	Verify the listing of contacts when the contacts list is empty.	-	A relevant message should be printed.

11	Verify contact deletion with an ID.	Valid ID	A contact with a matching ID should be deleted and a relevant message should be printed.
12	Verify contact deletion with a non-existent ID.	Non-existent ID	A relevant message should be printed.
13	Verify contact deletion when the contacts list is empty.	Sample ID	A relevant message should be printed.
14	Verify the search with a query matching a single contact.	Query matching a single contact	Matching contact should be returned.
15	Verify the search with a query matching several contacts.	Query matching multiple contacts	Matching contacts should be returned.
16	Verify the search with a query matching no contacts.	Query matching no contacts	A relevant message should be printed.
17	Verify the search when the contacts list is empty.	Sample query	A relevant message should be printed.
18	Verify sorting with default order.	order=’’	The contacts list sorted in ascending order of the name

			should be returned.
19	Verify the sorting in descending order.	order=’desc’	The contacts list sorted in descending order of the name should be returned.
20	Verify the sorting when the contacts list is empty.	order=’’	A relevant message should be printed.

Application Functionality

1. **Main Menu**
    a. Functionality:
        i. Displays a user-friendly main menu with options to view, add, search, delete, and sort contacts and quit the application.
        ii. Controls the flow of the application based on user input.
        iii. Allows users to quit the application gracefully.
    b. Implementation:
        i. \`main_menu()\` function uses \`time.sleep(1)\` to introduce a delay of 1 second before displaying the menu.
        ii. \`start_contact_book()\` function initiates an infinite loop to keep the menu active until the user chooses to quit.
        iii. Handles user input to execute corresponding functions based on the selected menu option.
        iv. Allows users to quit by entering 'X' or 'x', which breaks the loop and exits the application gracefully.
        v. Provides error handling for invalid menu choices, prompting the user to select again.
2. **Listing Contacts**
    a. Functionality:
        i. Integrated with the main menu to display contacts when selected.
        ii. Displays all contacts stored in the contacts list.
        iii. If the list is empty, it notifies the user with an appropriate message.

b. Implementation:

    i. ` list_contacts()` function implements listing of contacts.

    ii. Iterates through `contacts_list` and prints each contact's details including ID, name, and phone number.

    iii. Uses `enumerate` to display contacts with an indexed format.

    iv. Handles the case where the `contacts_list` is empty by printing a message indicating no contacts are present.

3. **Adding a Contact**

    a. Functionality:

        i. Integrated with the main menu for user interaction.

        ii. Allows users to add a new contact by providing a name and phone number.

        iii. Performs validations to ensure both fields are not empty or spaces and that the phone number is numeric.

        iv. Generates a unique ID for each contact before adding it to the contacts list.

    b. Implementation:

        i. `add_contact(name, phone)` function implements addition of a contact.

        ii. Validates input fields (`name` and `phone`) to ensure they are not empty or just spaces.

        iii. Checks if the phone is numeric using the `isnumeric()` method.

iv. Generates a unique ID for the new contact using the `generate_unique_id()` function from the `uuid` module.

v. Creates a dictionary with keys 'id', 'name', and 'phone', then appends it to `contacts_list`.

vi. Provides feedback to the user on successful contact addition or validation errors.

4. **Searching Contacts**

a. Functionality:

i. Integrated with the main menu for user interaction.

ii. Enables searching for a contact by ID, name, or phone number.

iii. Returns search results matching the query.

iv. Notifies the user if no contacts match the query.

b. Implementation:

i. `search_contacts(query)` function implements searching of contacts.

ii. Checks if `contacts_list` is empty and exits with a message if no contacts exist.

iii. Removes leading and trailing white spaces from `query`.

iv. Initializes `search_results` list to store matching contacts.

v. Iterates through `contacts_list` to find matches for the query against 'id', 'name', or 'phone' fields.

vi. Uses `enumerate` to display matching contacts with an indexed format or notifies the user if no matches are found.

5. **Sorting Contacts**

   a. Functionality:

      i. Integrated with the main menu for user interaction.

      ii. Sorts contacts alphabetically by name in either ascending or descending order based on user input.

      iii. Displays sorted contacts list with ID, name, and phone number.

   b. Implementation:

      i. `sort_contacts(order)` function implements sorting of contacts.

      ii. Checks if `contacts_list` is empty and exits with a message if no contacts exist.

      iii. Removes leading and trailing white spaces from `order` and converts it to lowercase.

      iv. Uses `sorted()` function to sort `contacts_list` based on the 'name' field.

      v. Sorts contacts in ascending order by default and switches to descending if order equals 'desc'.

      vi. Uses `enumerate` to display sorted contacts with an indexed format.

      vii. Provides clear feedback to the user about the sorting order applied.

6. **Deleting a Contact**

   a. Functionality:

      i. Integrated with the main menu for user interaction.

      ii. Allows users to delete a contact by providing its unique ID.

    iii. Confirms deletion with appropriate messages.
    iv. Notifies the user if the contact ID is not found.

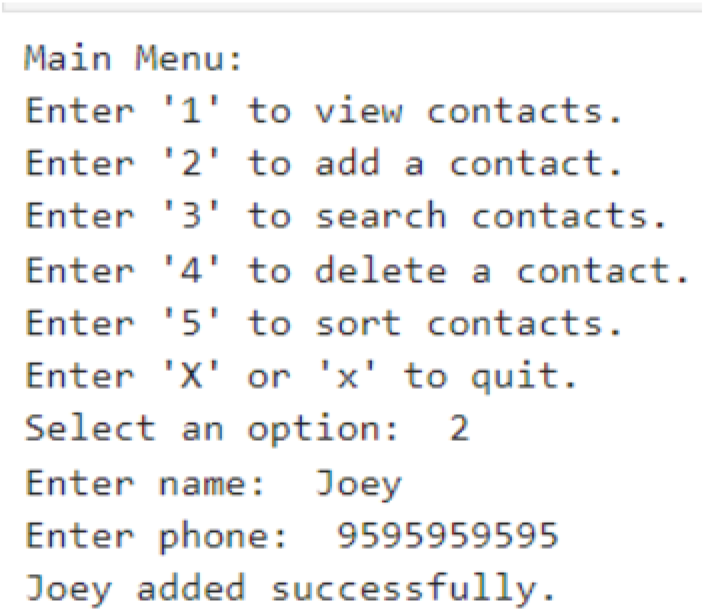
  b. Implementation:
    i. `delete_contact(contact_id)` function implements deletion of a contact.
    ii. Checks if `contacts_list` is empty and exits with a message if no contacts exist.
    iii. Removes leading and trailing white spaces from `contact_id`.
    iv. Iterates through `contacts_list` to find a match for `contact_id`.
    v. Deletes the contact upon finding a match and provides confirmation.
    vi. Prints a message if the contact ID does not exist in the `contacts_list`.

Testing and Validation

The Contact Book application is thoroughly tested with positive and negative test cases to ensure robust functionality (Gore, 2024).

Test Cases:

1. Verify contact creation with a name and phone.
    a. Input Data: name='Joey', phone='9595959595'
    b. Expected Results: A new contact should be added and a relevant message should be printed.
    c. Actual Results: A new contact is added and a relevant message is printed.



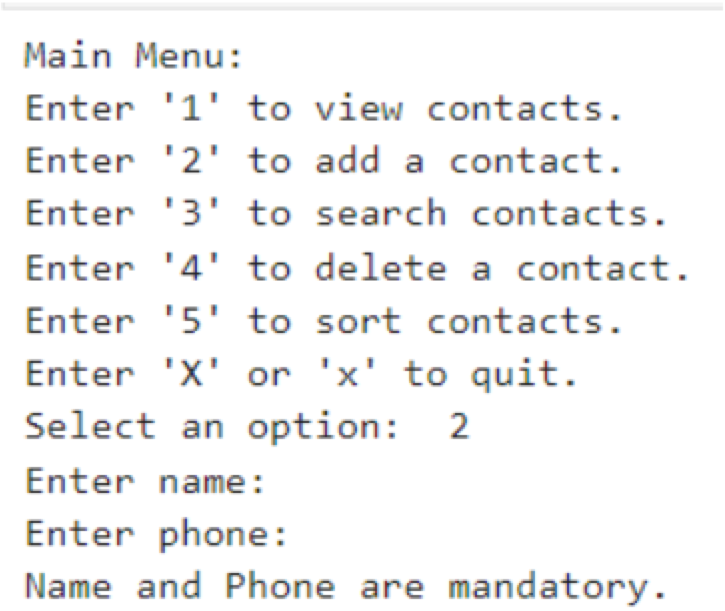
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:  Joey
Enter phone:  9595959595
Joey added successfully.
```

2. Verify contact creation with a blank name and blank phone.

    a. Input Data: name=’’, phone=’’

    b. Expected Results: A relevant message should be printed.

    c. Actual Results: A relevant message is printed.



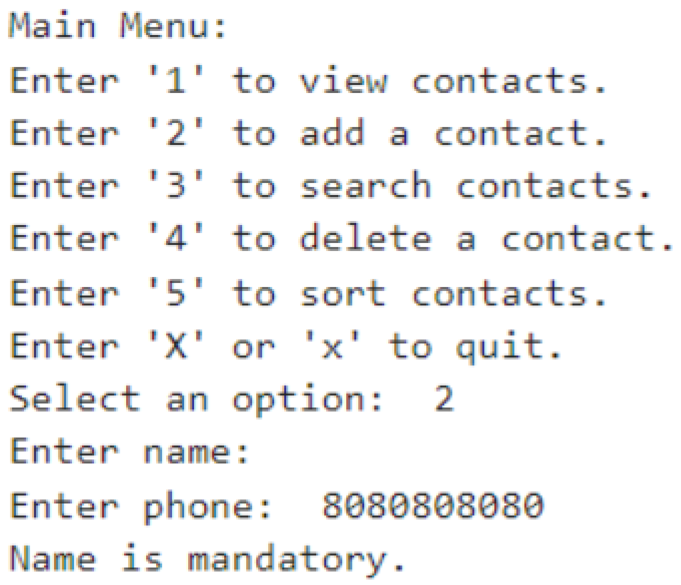
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:
Enter phone:
Name and Phone are mandatory.
```

3. Verify contact creation with a blank name.

    a. Input Data: name=’’, phone=’8080808080’

b. Expected Results: A relevant message should be printed.

c. Actual Results: A relevant message is printed.



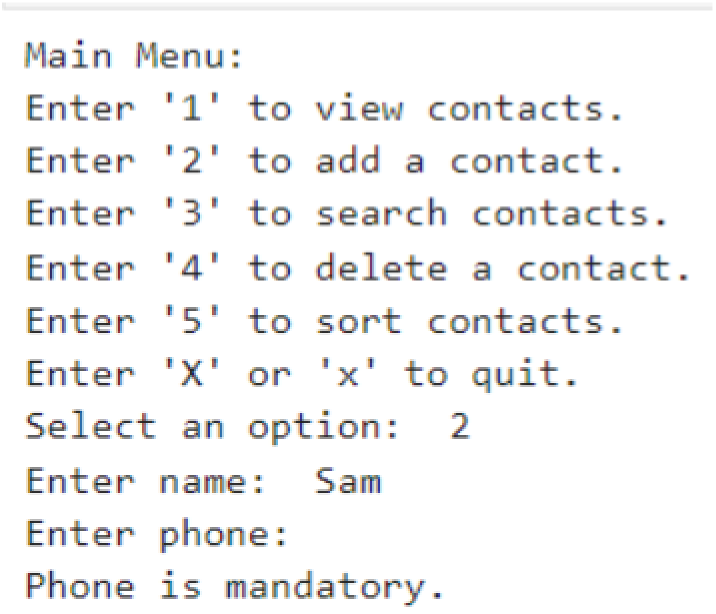
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:
Enter phone:  8080808080
Name is mandatory.
```

4. Verify contact creation with a blank phone.

    a. Input Data: name=’Sam’, phone=’’

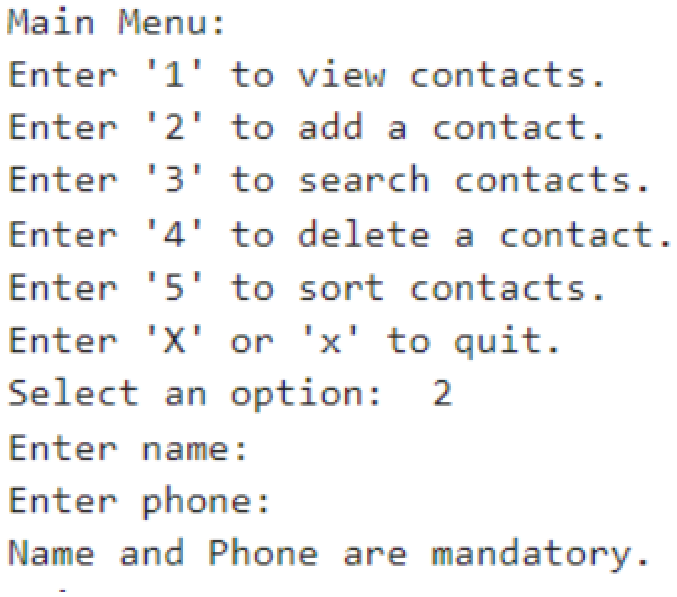
    b. Expected Results: A relevant message should be printed.

    c. Actual Results: A relevant message is printed.



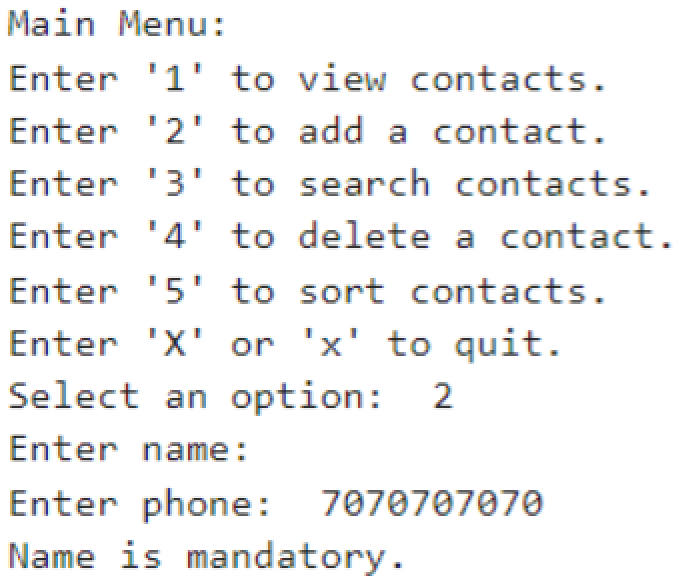
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:  Sam
Enter phone:
Phone is mandatory.
```

5. Verify contact creation by passing a space in name and phone.
    a. Input Data: name=’ ’, phone=’ ’
    b. Expected Results: A relevant message should be printed.
    c. Actual Results: A relevant message is printed.



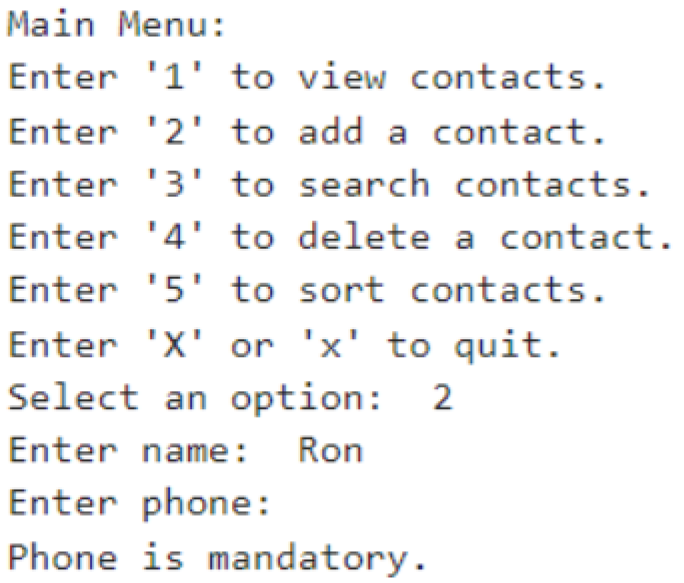
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:
Enter phone:
Name and Phone are mandatory.
```

6. Verify contact creation by passing a space in the name.
    a. Input Data: name=’ ’, phone=’7070707070’
    b. Expected Results: A relevant message should be printed.
    c. Actual Results: A relevant message is printed.



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:
Enter phone:  7070707070
Name is mandatory.
```

7. Verify contact creation by passing a space in the phone.
    a. Input Data: name='Ron', phone=' '
    b. Expected Results: A relevant message should be printed.
    c. Actual Results: A relevant message is printed.

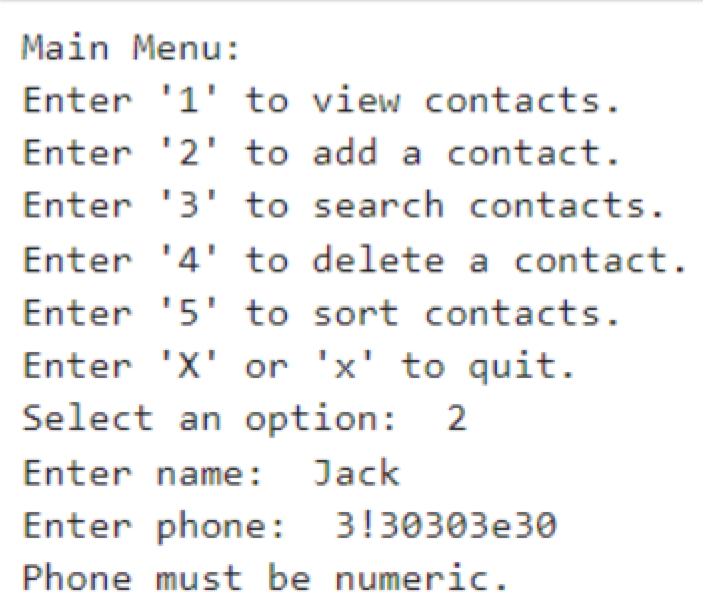


```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:  Ron
Enter phone:
Phone is mandatory.
```

8. Verify contact creation with a non-numeric phone.
    a. Input Data: name='Jack', phone='3!30303e30'

b. Expected Results: A relevant message should be printed.

c. Actual Results: A relevant message is printed.



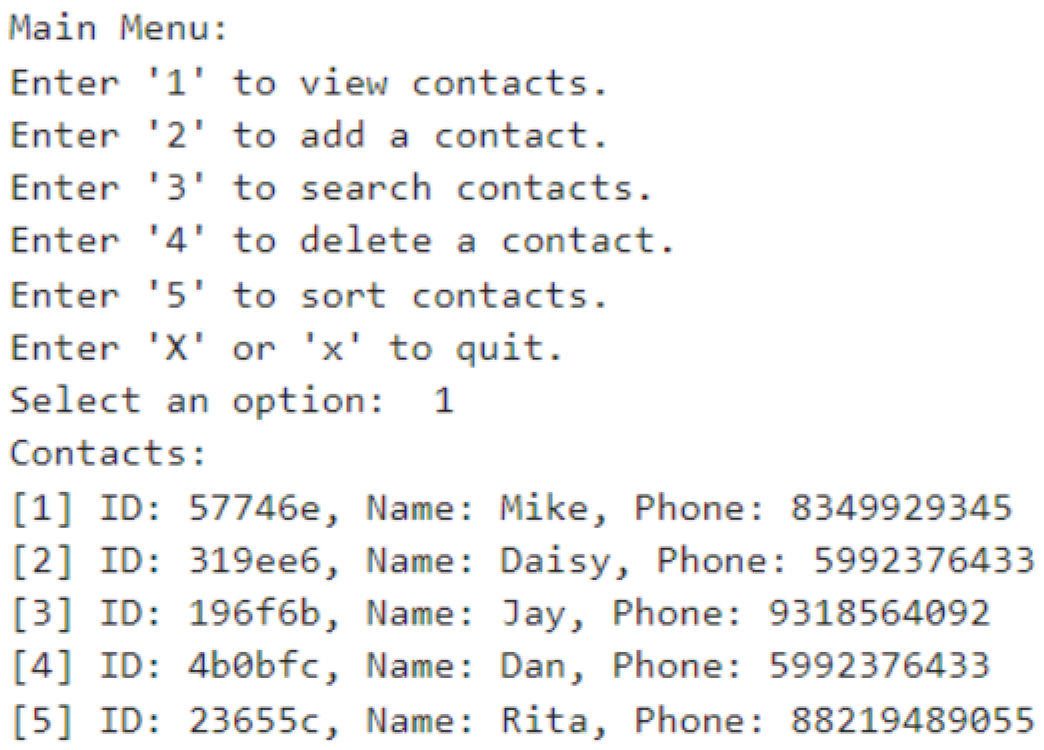
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  2
Enter name:  Jack
Enter phone:  3!30303e30
Phone must be numeric.
```

9. Verify the listing of contacts when the contacts list is not empty.

    a. Input Data: -

    b. Expected Results: The contacts list should be returned.

    c. Actual Results: The contacts list is returned.



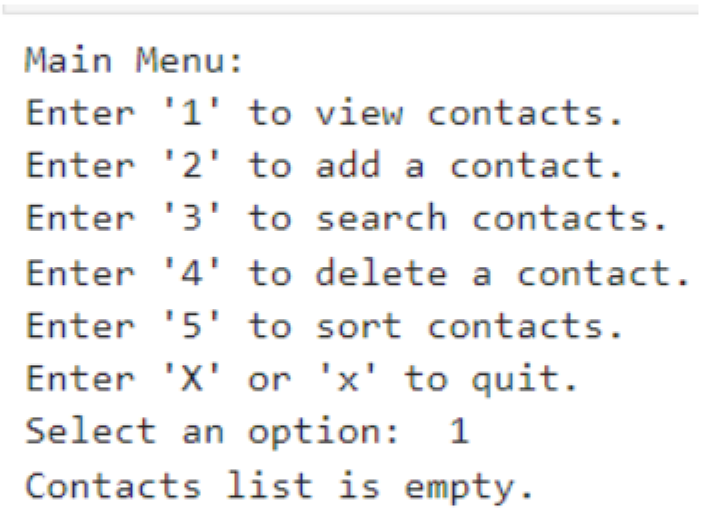
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  1
Contacts:
[1] ID: 57746e, Name: Mike, Phone: 8349929345
[2] ID: 319ee6, Name: Daisy, Phone: 5992376433
[3] ID: 196f6b, Name: Jay, Phone: 9318564092
[4] ID: 4b0bfc, Name: Dan, Phone: 5992376433
[5] ID: 23655c, Name: Rita, Phone: 88219489055
```

10. Verify the listing of contacts when the contacts list is empty.

    a. Input Data: -

    b. Expected Results: A relevant message should be printed.

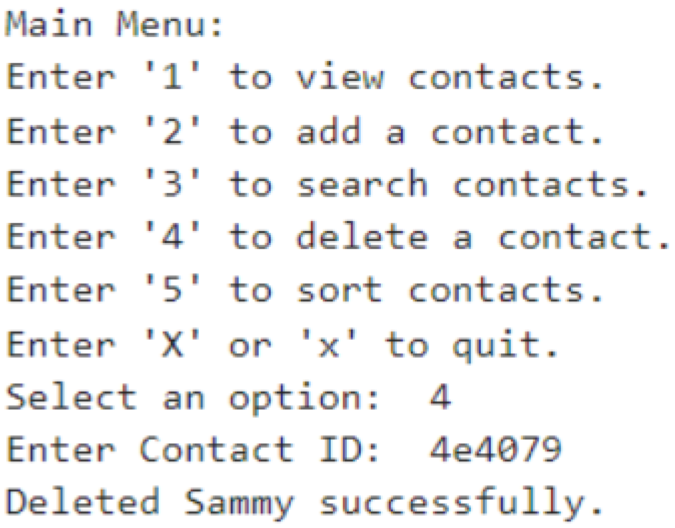
    c. Actual Results: A relevant message is printed.



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  1
Contacts list is empty.
```

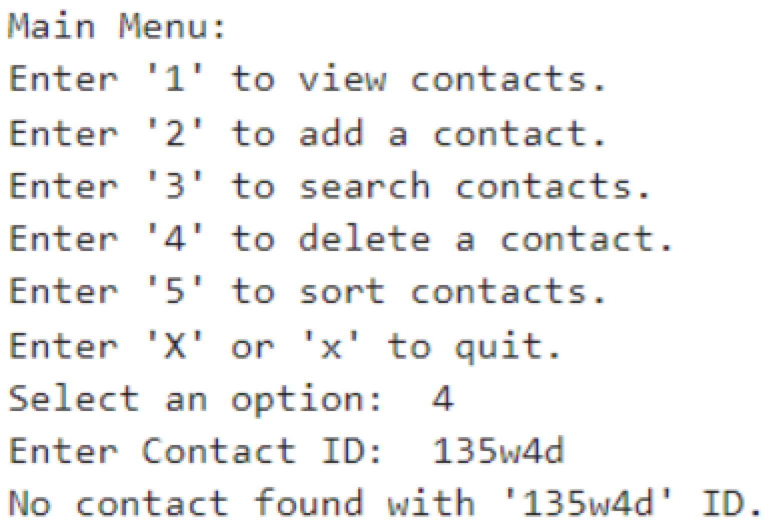
11. Verify contact deletion with an ID.

   a. Input Data: Valid ID
   b. Expected Results: A contact with a matching ID should be deleted and a relevant message should be printed.
   c. Actual Results: A contact with a matching ID is deleted and a relevant message is printed.



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  4
Enter Contact ID:  4e4079
Deleted Sammy successfully.
```

12. Verify contact deletion with a non-existent ID.
    a. Input Data: Non-existent ID
    b. Expected Results: A relevant message should be printed.
    c. Actual Results: A relevant message is printed.



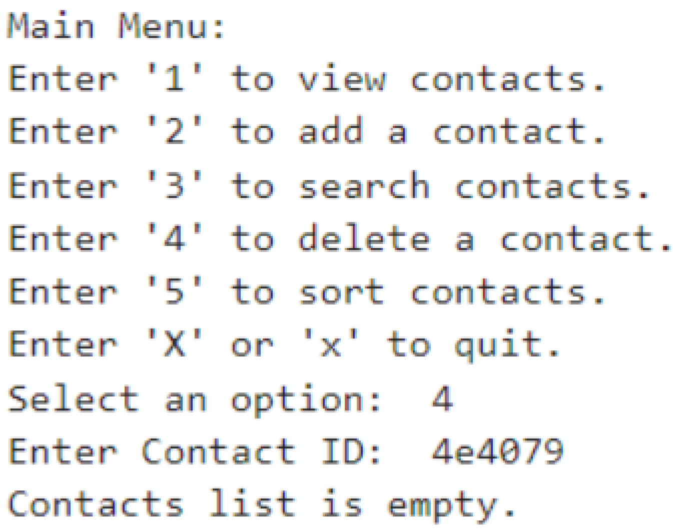
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  4
Enter Contact ID:  135w4d
No contact found with '135w4d' ID.
```

13. Verify contact deletion when the contacts list is empty.

    a. Input Data: Sample ID

    b. Expected Results: A relevant message should be printed.

    c. Actual Results: A relevant message is printed.



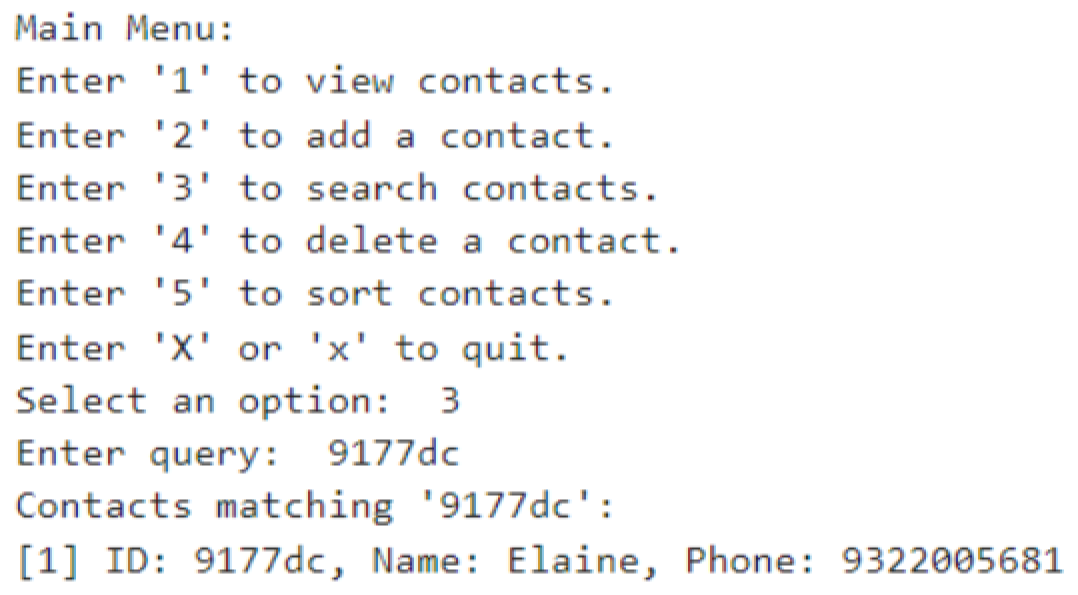
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  4
Enter Contact ID:  4e4079
Contacts list is empty.
```

14. Verify the search with a query matching a single contact.

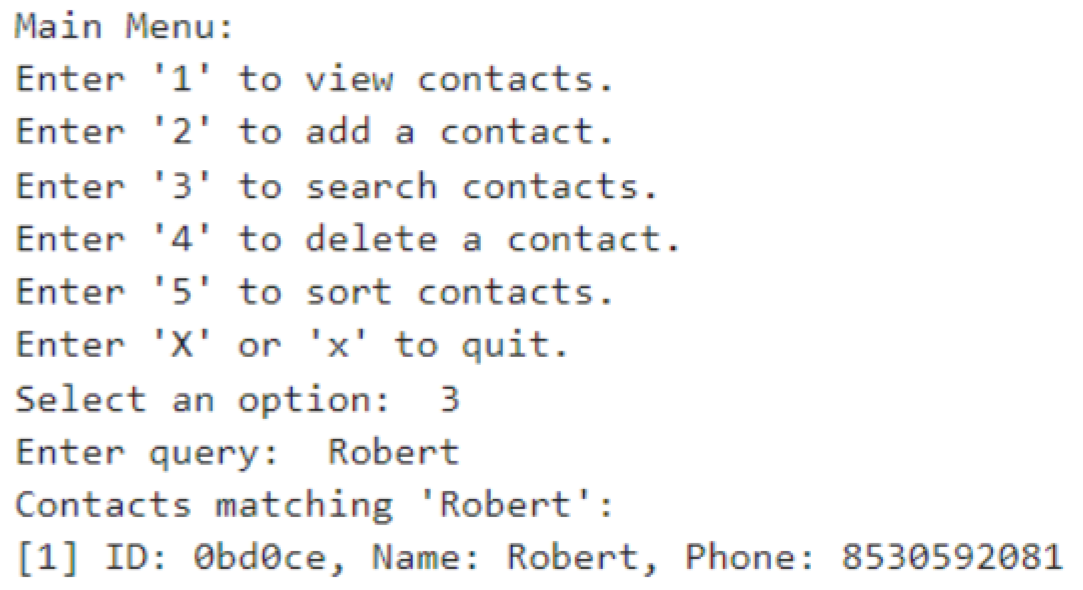
    a. Input Data: Query matching a single contact

    b. Expected Results: Matching contact should be returned.

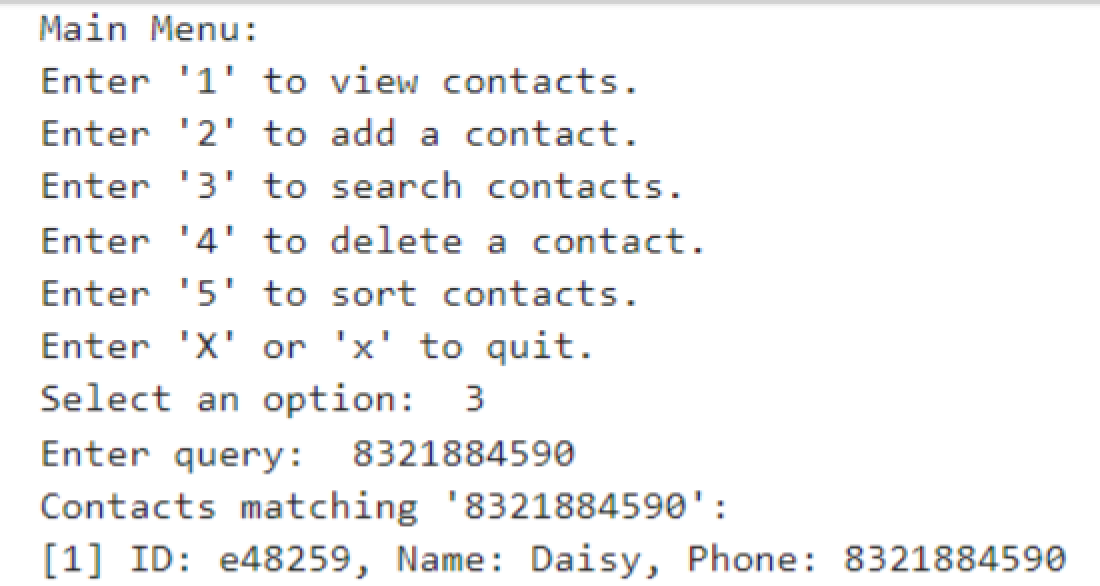
    c. Actual Results: Matching contact is returned.



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  9177dc
Contacts matching '9177dc':
[1] ID: 9177dc, Name: Elaine, Phone: 9322005681
```



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  Robert
Contacts matching 'Robert':
[1] ID: 0bd0ce, Name: Robert, Phone: 8530592081
```



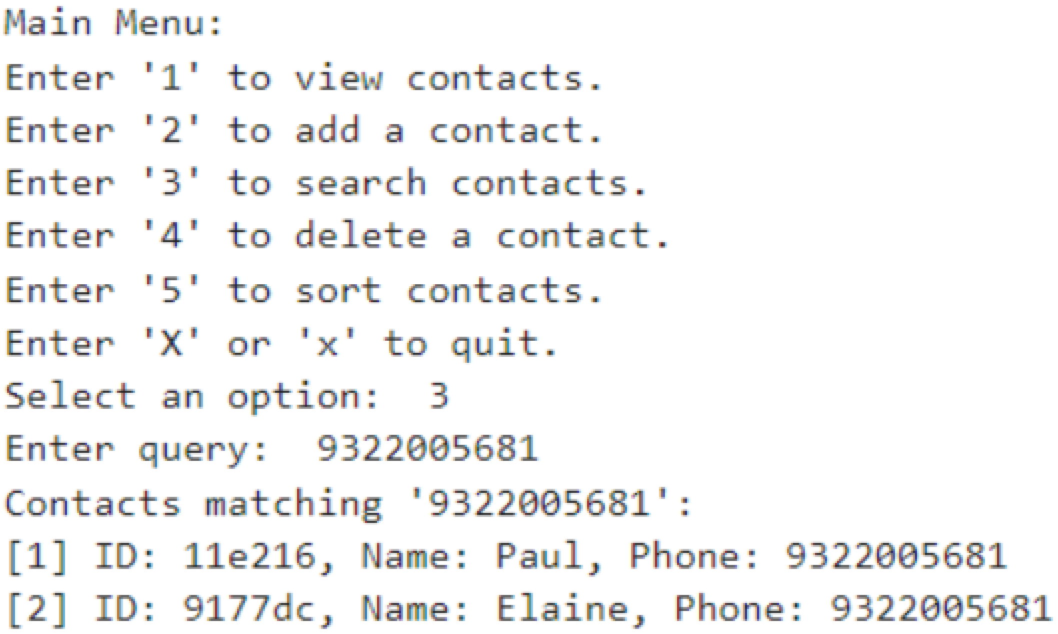
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  8321884590
Contacts matching '8321884590':
[1] ID: e48259, Name: Daisy, Phone: 8321884590
```

15. Verify the search with a query matching several contacts.

    a. Input Data: Query matching multiple contacts

    b. Expected Results: Matching contacts should be returned.

    c. Actual Results: Matching contacts is returned.



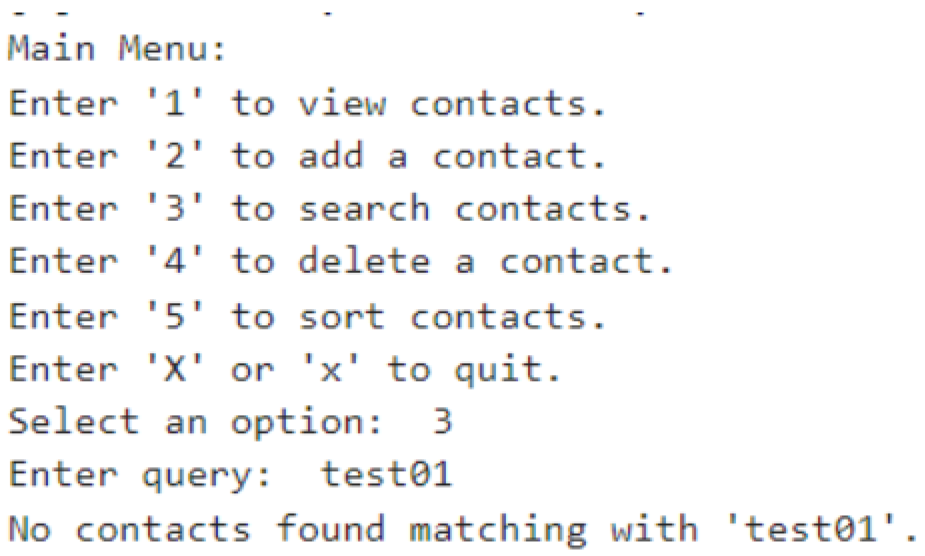
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  9322005681
Contacts matching '9322005681':
[1] ID: 11e216, Name: Paul, Phone: 9322005681
[2] ID: 9177dc, Name: Elaine, Phone: 9322005681
```

16. Verify the search with a query matching no contacts.

a. Input Data: Query matching no contacts

b. Expected Results: A relevant message should be printed.

c. Actual Results: A relevant message is printed.



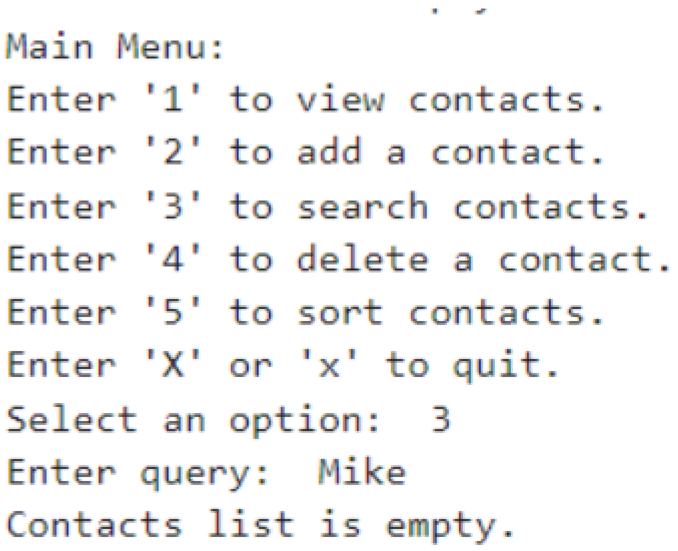
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  test01
No contacts found matching with 'test01'.
```

17. Verify the search when the contacts list is empty.

a. Input Data: Sample query

b. Expected Results: A relevant message should be printed.

c. Actual Results: A relevant message is printed.



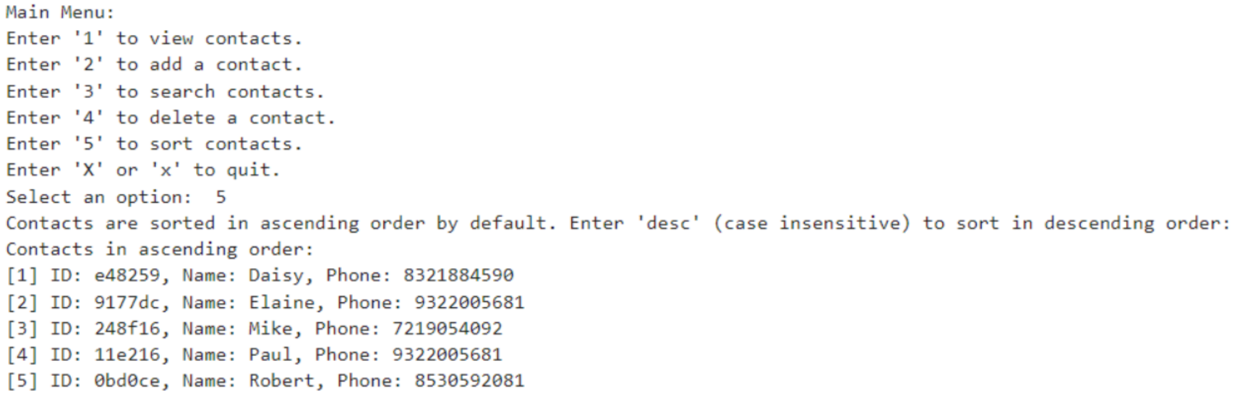
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  3
Enter query:  Mike
Contacts list is empty.
```

18. Verify sorting with default order.

a. Input Data: order=’’

b. Expected Results: The contacts list sorted in ascending order of the name should be returned.

c. Actual Results: The contacts list sorted in ascending order of the name is returned.



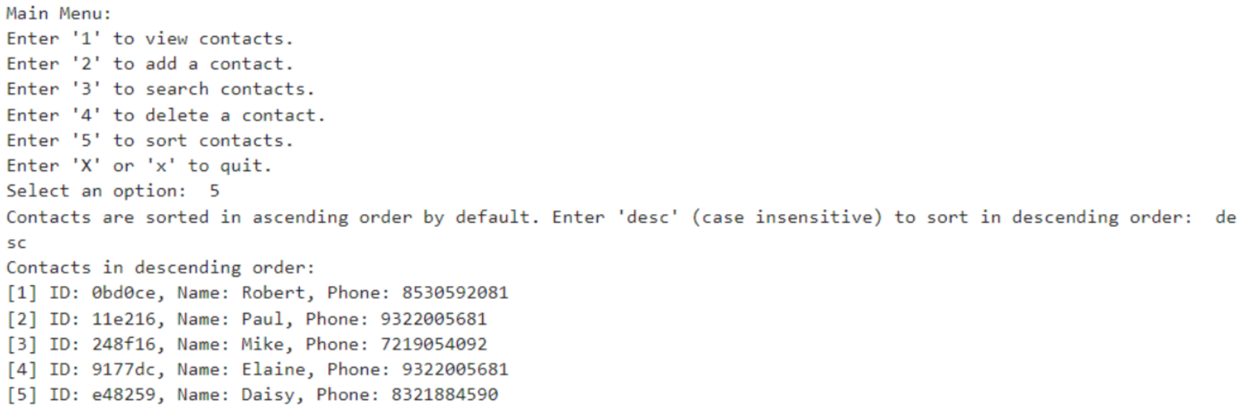
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  5
Contacts are sorted in ascending order by default. Enter 'desc' (case insensitive) to sort in descending order:
Contacts in ascending order:
[1] ID: e48259, Name: Daisy, Phone: 8321884590
[2] ID: 9177dc, Name: Elaine, Phone: 9322005681
[3] ID: 248f16, Name: Mike, Phone: 7219054092
[4] ID: 11e216, Name: Paul, Phone: 9322005681
[5] ID: 0bd0ce, Name: Robert, Phone: 8530592081
```

19. Verify the sorting in descending order.

a. Input Data: order=’desc’

b. Expected Results: The contacts list sorted in descending order of the name should be returned.

c. Actual Results: The contacts list sorted in descending order of the name is returned.



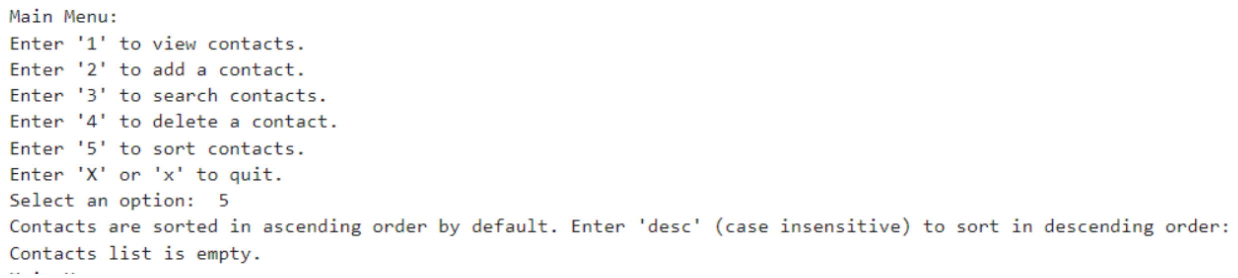
```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  5
Contacts are sorted in ascending order by default. Enter 'desc' (case insensitive) to sort in descending order:  de
sc
Contacts in descending order:
[1] ID: 0bd0ce, Name: Robert, Phone: 8530592081
[2] ID: 11e216, Name: Paul, Phone: 9322005681
[3] ID: 248f16, Name: Mike, Phone: 7219054092
[4] ID: 9177dc, Name: Elaine, Phone: 9322005681
[5] ID: e48259, Name: Daisy, Phone: 8321884590
```

20. Verify the sorting when the contacts list is empty.

   a. Input Data: order=’’

   b. Expected Results: A relevant message should be printed.

   c. Actual Results: A relevant message is printed.



```
Main Menu:
Enter '1' to view contacts.
Enter '2' to add a contact.
Enter '3' to search contacts.
Enter '4' to delete a contact.
Enter '5' to sort contacts.
Enter 'X' or 'x' to quit.
Select an option:  5
Contacts are sorted in ascending order by default. Enter 'desc' (case insensitive) to sort in descending order:
Contacts list is empty.
```

References

Gore, S. (2024) Contact Book Application. *LCS Launching into Computer Science April 2024*. Assignment submitted to the University of Essex Online.