Phonebook Management System

Brief introduction:

Here we have used **doubly link list** to arrange sort and store all the contacts provided by the user and **circular queue** to store and display recently deleted contacts.

The Advantages of using doubly link list: -

- Allows us to iterate in both directions.
- We can delete a node easily only by getting the node we want to delete no need of previous or next node.
- Reversing is easy.

The Advantages of using circular queue: -

- In the circular queue, elements can be inserted easily if there are vacant locations until it is not fully occupied, whereas in the case of a linear queue insertion is not possible once the rear reaches the last index even if there are empty locations present in the queue.
- The main advantage we got for using circular queue is that it never got full, whenever it got full it removed element from front and inserted new data there.

Features of the project:

There are various features we have used in this project mentioned below:

1. Inserting a contact in phone book
2. Sorting contacts in ascending order
3. Display all contacts
4. Delete any contact by name
5. Search any contact by name or by number
6. Update name or number of any contact
7. Display recently deleted latest 10 contacts

- Inserting a contact in Phone Book:

  We ask user for name and number and store it in the doubly link list.

- Sorting contacts in Phone Book:

  After inserting contact we have automatically called sorting function which sorts the entire list. Sorting technique we have used is selection sort.

- Display all contacts in Phone Book:

  If we want to see what contacts we have inserted we can display them.

- Delete any contact by name:

  If we want to delete any contact we just need to know its name and we can delete it.

- Search any contact by name or number:

  If we want to check the details of any contact or if we want to know if a particular contact is in our phonebook or not then we just need to know its name or number and we can easily view the details of that contact.

- Update name or number of any contact:

  If we want to update a contact we just need to know its name and we can easily update name or number.

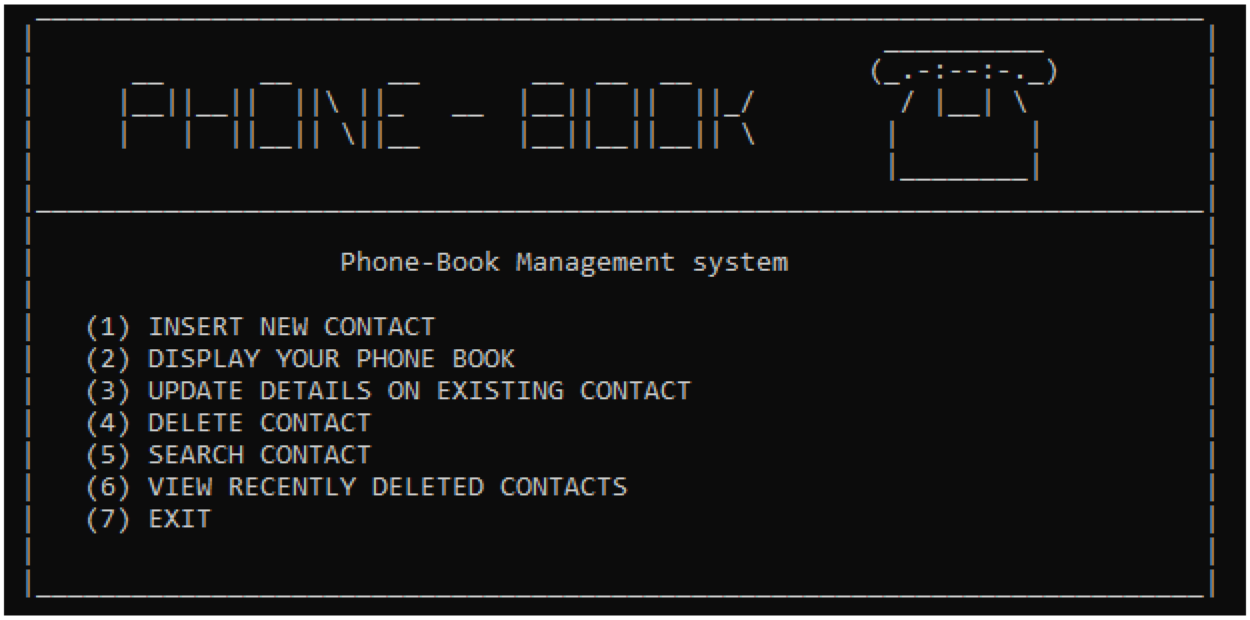
- Display recently deleted contacts:

  If we want to view which contacts we deleted recently we can easily view that and it will only store the latest 10 deleted items.

Methodology:

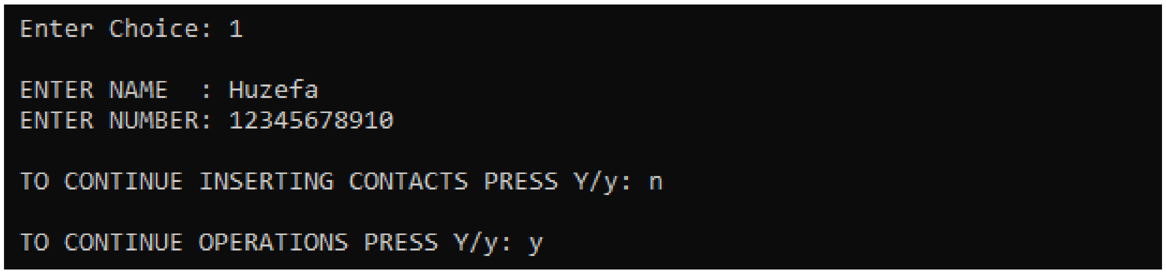
Now we will discuss each and every feature in detail.

First of all there will be a menu with different operations that we can perform on them and we can select from the menu



```
                   Phone-Book Management system

    (1) INSERT NEW CONTACT
    (2) DISPLAY YOUR PHONE BOOK
    (3) UPDATE DETAILS ON EXISTING CONTACT
    (4) DELETE CONTACT
    (5) SEARCH CONTACT
    (6) VIEW RECENTLY DELETED CONTACTS
    (7) EXIT
```

- **Insert new contact**

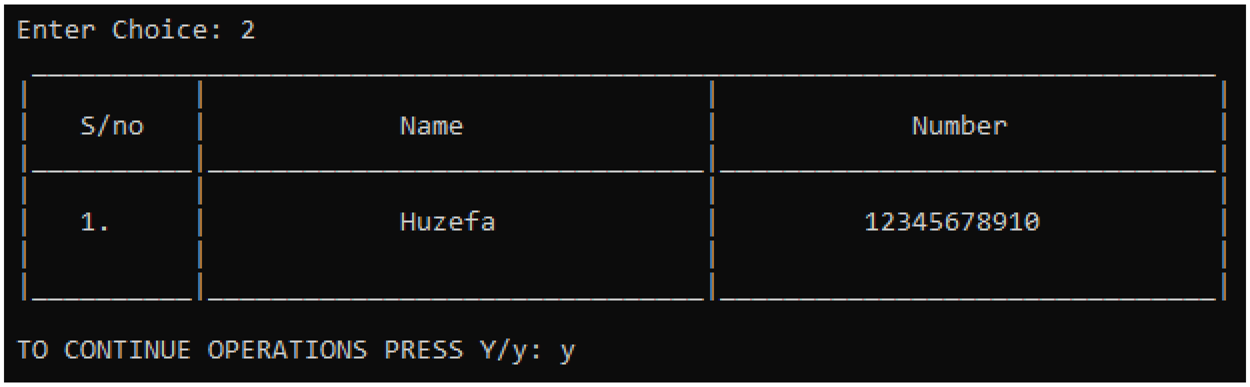


```
Enter Choice: 1

ENTER NAME  : Huzefa
ENTER NUMBER: 12345678910

TO CONTINUE INSERTING CONTACTS PRESS Y/y: n

TO CONTINUE OPERATIONS PRESS Y/y: y
```

Here when we selected insert new option then it asked user for name and number and after inputting name and number it inserts them at the head of list to reduce time complexity as we don't have to traverse the whole list till the end then after inserting contact in the list it automatically sorts the list using selection sort and then asks us to enter another contact, if we don't want to enter another contact then it just asks us to go back to menu and perform other operations.

```
void InsertContacts()
{
	string option;
	do
	{
		struct Node* newNode = new Node();
		newNode->next = Head; //Intitializing newnode -> next part to head because 
we are inserting it at head to reduce time complexity
		newNode->prev = NULL; //We know that Head node prev part is always NULL
		cout << "\n ENTER NAME  : ";
		cin >> newNode->name;
		cout << " ENTER NUMBER: ";
		cin >> newNode->number;

		while (newNode->number.length() != 11)//comparing if the entered number is 
equal to 11 or not
		{
			cout << " ENTER VALID NUMBER: ";
			cin >> newNode->number;
		}
		if (Head == NULL) //if the node inserted is first node then initialize Head 
= newnode
		{
			Head = newNode;
		}
		else //If the node inserted is not the first node then update prev pointer 
of head node to newnode
		{
			Head->prev = newNode;
			Head = newNode; //Set newnode as head
		}

		cout << "\n TO CONTINUE INSERTING CONTACTS PRESS Y/y: ";
		cin >> option;
	} while (option == "y" || option == "Y");

}

void SortingContact()
{
	Node* i, * j;
	for (i = Head; i->next != NULL; i = i->next)//start from head till second last 
node
	{
		for (j = i->next; j != NULL; j = j->next) //start from current node pointed 
by i till last node
		{
			if (i->name > j->name) //comparing names to sort in ascending order
			{
				swap(i->name, j->name);
				swap(i->number, j->number);
			}
		}
	}
}
```

- **Display your Phone Book**

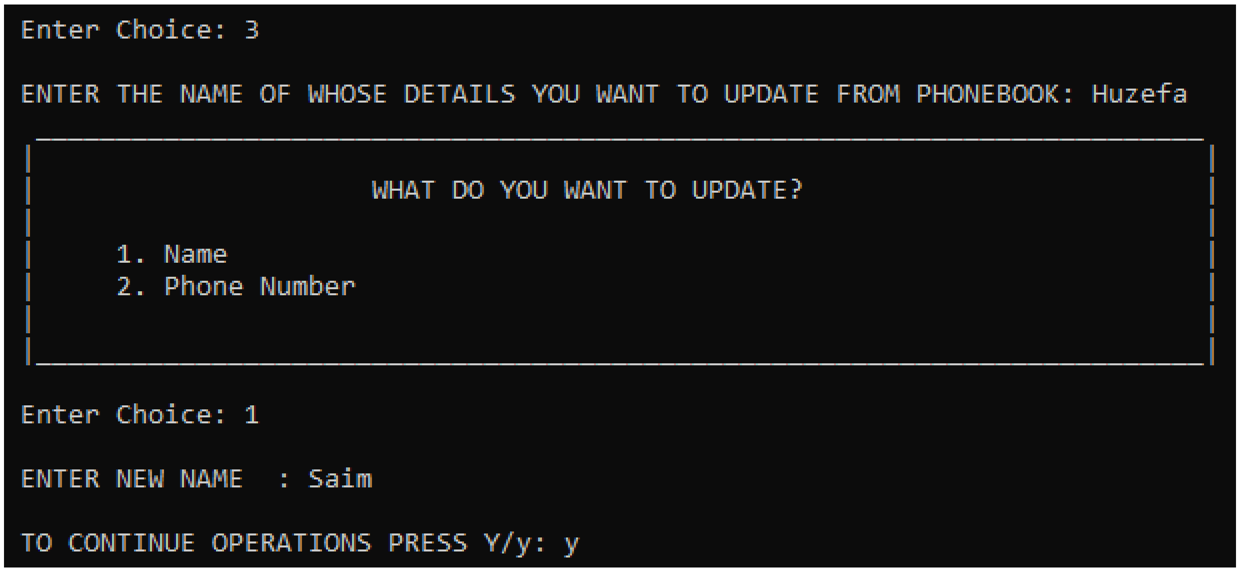


```
Enter Choice: 2
 _______________________________________________________________________________
|           |                                  |                                 |
|    S/no   |              Name                |              Number             |
|___________|__________________________________|_________________________________|
|           |                                  |                                 |
|    1.     |              Huzefa              |           12345678910           |
|           |                                  |                                 |
|___________|__________________________________|_________________________________|

TO CONTINUE OPERATIONS PRESS Y/y: y
```

Here when we selected display your phone book it displays all the contacts that are stored in the list starting from head till the last node and then asks us if we to do more operations or exit.

```
void DisplayContacts()
{
       int i = 0; //for serial no
       struct Node* ptr;
       ptr = Head;    //starting point of the list to traverse it
       cout << "
___________________________________________________________________________" << endl;
       cout << " |          |                                |
|" << endl;
       cout << " |   S/no   |            Name                |            Number
|" << endl;
       cout << "
|__________|________________________________|________________________________| " << endl;
       cout << " |          |                                |
|" << endl;
       while (ptr != NULL)//traverse till last
       {
              i++;
              if (ptr->name.length() > 6) // comparing length of name to display it
inside box
              {
                     cout << " |   " << i << ".     |\t\t " << ptr->name << "\t    |\t
" << ptr->number << "\t    |" << endl;
              }
              else
              {
                     cout << " |   " << i << ".     |\t\t " << ptr->name << "\t\t    |\t
" << ptr->number << "\t    |" << endl;
              }
              ptr = ptr->next; //move to next node
       }
       cout << " |          |                                |
|" << endl;
       cout << "
|__________|________________________________|________________________________|" << endl;

   }
```

- **Update an existing contact**



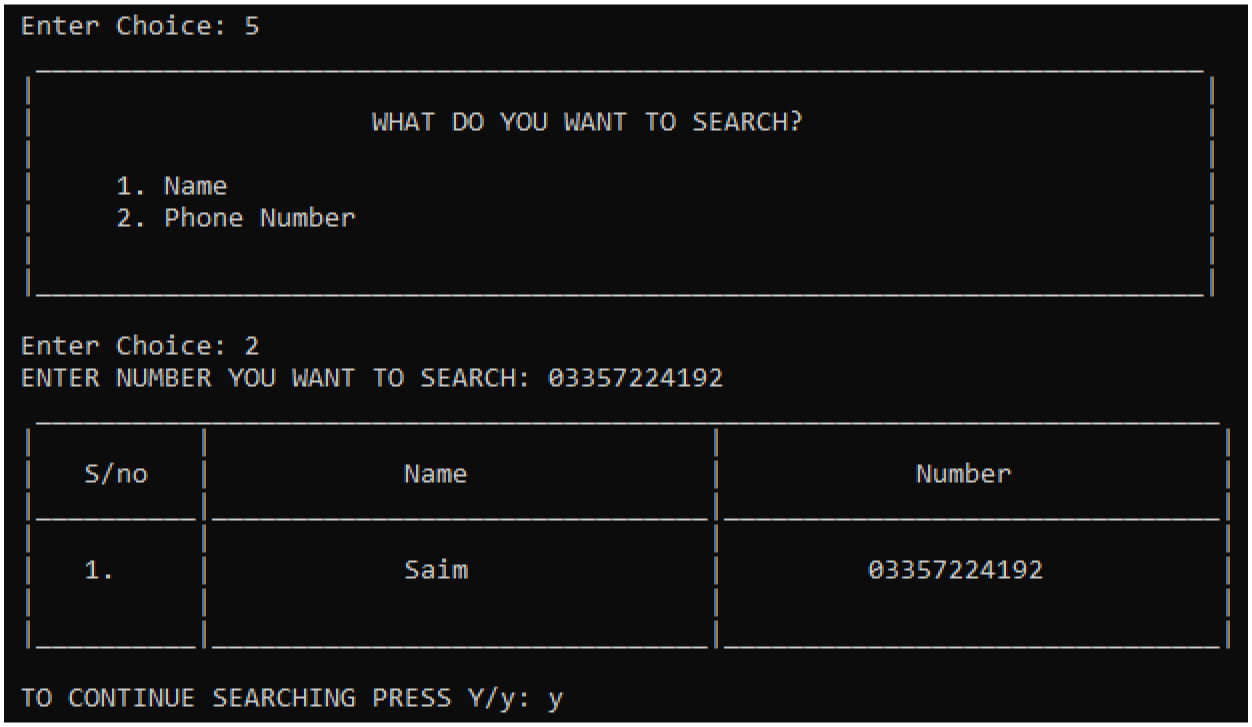
```
Enter Choice: 3

ENTER THE NAME OF WHOSE DETAILS YOU WANT TO UPDATE FROM PHONEBOOK: Huzefa
 ___________________________________________________________________________
|                                                                           |
|                     WHAT DO YOU WANT TO UPDATE?                           |
|                                                                           |
|     1. Name                                                               |
|     2. Phone Number                                                       |
|                                                                           |
|___________________________________________________________________________|

Enter Choice: 1

ENTER NEW NAME  : Saim

TO CONTINUE OPERATIONS PRESS Y/y: y
```

If we want to update any contact then it will ask us for the name of the contact we want to update and if that name is present in the contacts then it will ask us what we want to update and if we select name then it will update the name and perform sorting again on it to sort the name in order and if we update the number it updates it and then it asks us for if we want to exit or perform other operations.

```
void UpdateContact(string checkname)
{
	int choose;
	struct Node* updatenode;
	updatenode = Head;
	while (updatenode != NULL) //check until all nodes traversed
	{
		if (checkname == updatenode->name)//compared entered name by user with data in linked list
		{
			cout << " ___________________________________________________________________________ " << endl;
			cout << " |                                                                           |" << endl;
			cout << " |                     WHAT DO YOU WANT TO UPDATE?                           |" << endl;
			cout << " |                                                                           |" << endl;
			cout << " |     1. Name                                                               |" << endl;
			cout << " |     2. Phone Number                                                       |" << endl;
			cout << " |                                                                           |" << endl;
			cout << " |___________________________________________________________________________|" << endl;
			cout << "\n Enter Choice: ";
```

```
            cin >> choose;
            switch (choose)
            {
            case 1:
                cout << "\n ENTER NEW NAME  : ";
                cin >> updatenode->name;
                SortingContact();
                break;
            case 2:
                cout << " ENTER NEW NUMBER: ";
                cin >> updatenode->number;
                while (updatenode->number.length() != 11)//comparing length of
number if it equal to 11 or not
                {
                    cout << " ENTER VALID NUMBER: ";
                    cin >> updatenode->number;
                }
                break;
            default:
                cout << "
_________________________________________________________________________ " << endl;
                cout << " |
|" << endl;
                cout << " |                      WRONG INPUT GIVEN....
|" << endl;
                cout << "
|_________________________________________________________________________|" << endl;
            }
            return;
        }
        updatenode = updatenode->next; //move to next node
    }
    cout << "
_________________________________________________________________________ " << endl;
    cout << " |
|" << endl;
    cout << " |              The name doesnt exist in the current data.
|" << endl;
    cout << "
|_________________________________________________________________________|" << endl;
}
```

- **Search contact**



```
Enter Choice: 5
 ________________________________________________________________________
|                                                                        |
|                     WHAT DO YOU WANT TO SEARCH?                        |
|                                                                        |
|     1. Name                                                            |
|     2. Phone Number                                                    |
|                                                                        |
|________________________________________________________________________|

Enter Choice: 2
ENTER NUMBER YOU WANT TO SEARCH: 03357224192
 __________________________________________________________________________
|          |                               |                               |
|   S/no   |             Name              |            Number             |
|__________|_______________________________|_______________________________|
|          |                               |                               |
|   1.     |             Saim              |         03357224192           |
|          |                               |                               |
|__________|_______________________________|_______________________________|

TO CONTINUE SEARCHING PRESS Y/y: y
```

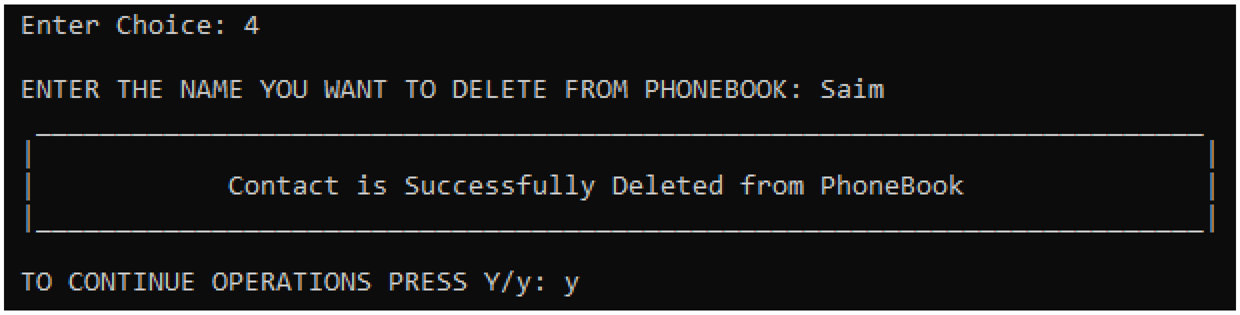
If we select search operation then it will give us two options either we want to search by name or by phone number, after searching that data it will display the details of that contact and then it will ask us if we want to continue searching for more contacts if we don’t want to search contacts then it will ask us if we want to exit or continue doing more operations.

```
void SearchContact()
{
	int choose;
	string num;
	string name;
	string option;
	do
	{
		cout << " 
______________________________________________________________________________ " << endl;
		cout << " |                                                                          
|" << endl;
		cout << " |                     WHAT DO YOU WANT TO SEARCH?                          
|" << endl;
		cout << " |                                                                          
|" << endl;
		cout << " |     1. Name                                                              
|" << endl;
		cout << " |     2. Phone Number                                                      
|" << endl;
		cout << " |                                                                          
|" << endl;
		cout << " 
|______________________________________________________________________________|" << endl;
		cout << "\n Enter Choice: ";
		cin >> choose;
```

```
                switch (choose)
                {
                case 1:
                        cout << " ENTER NAME YOU WANT TO SEARCH: ";
                        cin >> name;
                        searchname(name);
                        break;
                case 2:
                        cout << " ENTER NUMBER YOU WANT TO SEARCH: ";
                        cin >> num;
                        searchnum(num);
                        break;
                default:
                        cout << "
__________________________________________________________________________  " << endl;
                        cout << " |
|" << endl;
                        cout << " |                          WRONG INPUT GIVEN....
|" << endl;
                        cout << "
|_________________________________________________________________________|" << endl;
                        break;
                }
                cout << "\n TO CONTINUE SEARCHING PRESS Y/y: ";
                cin >> option;
        } while (option == "y" || option == "Y");
    }

void searchname(string searchnamecheck)
{
        int i = 1;
        struct Node* searchnode;
        searchnode = Head;
        while (searchnode != NULL)//traversing till last node
        {
                if (searchnamecheck == searchnode->name)
                {
                        cout << "
___________________________________________________________________________" << endl;
                        cout << " |          |                                  |
|" << endl;
                        cout << " |   S/no   |             Name                 |
Number               |" << endl;
                        cout << "
|__________|________________________________|________________________________| " << endl;
                        cout << " |          |                                  |
|" << endl;
                        if (searchnode->name.length() > 6) // comparing length of name to
display it inside box
                        {
                                cout << " |    " << i << ".      |\t\t " << searchnode->name <<
"\t    |\t      " << searchnode->number << "\t    |" << endl;
                        }
                        else
                        {
                                cout << " |    " << i << ".      |\t\t " << searchnode->name <<
"\t\t    |\t      " << searchnode->number << "\t    |" << endl;
                        }
                        cout << " |          |                               |
|" << endl;
                        cout << "
|__________|______________________________|______________________________|" << endl;
                        return;
                }
                searchnode = searchnode->next;
        }
        cout << "
______________________________________________________________________ " << endl;
        cout << " |
|" << endl;
        cout << " |             Entered name is not present in the PhoneBook
|" << endl;
        cout << "
|______________________________________________________________________|" << endl;
}

void searchnum(string searchnumbercheck)
{
        int i = 1;
        struct Node* searchnode;
        searchnode = Head;
        while (searchnode != NULL)//traversing till last node
        {
                if (searchnumbercheck == searchnode->number)
                {
                        cout << "
______________________________________________________________________" << endl;
                        cout << " |          |                               |
|" << endl;
                        cout << " |   S/no   |             Name              |
Number              |" << endl;
                        cout << "
|__________|______________________________|______________________________| " << endl;
                        cout << " |          |                               |
|" << endl;
                        if (searchnode->name.length() > 6) // comparing length of name to
display it inside box
                        {
                                cout << " |    " << i << ".      |\t\t " << searchnode->name <<
"\t    |\t      " << searchnode->number << "\t    |" << endl;
                        }
                        else
                        {
                                cout << " |    " << i << ".      |\t\t " << searchnode->name <<
"\t\t    |\t      " << searchnode->number << "\t    |" << endl;
                        }
                        cout << " |          |                               |
|" << endl;
                        cout << "
|__________|______________________________|______________________________|" << endl;
                        return;
                }
                searchnode = searchnode->next;
        }
        cout << "
______________________________________________________________________ " << endl;
```

```
        cout << " |
|" << endl;
        cout << " |             Entered number is not present in the PhoneBook
|" << endl;
        cout << "
|___________________________________________________________________________|" << endl;
    }
```

- **Delete contact**



```
Enter Choice: 4

ENTER THE NAME YOU WANT TO DELETE FROM PHONEBOOK: Saim
 ___________________________________________________________________________
|                                                                           |
|            Contact is Successfully Deleted from PhoneBook                 |
|___________________________________________________________________________|

TO CONTINUE OPERATIONS PRESS Y/y: y
```

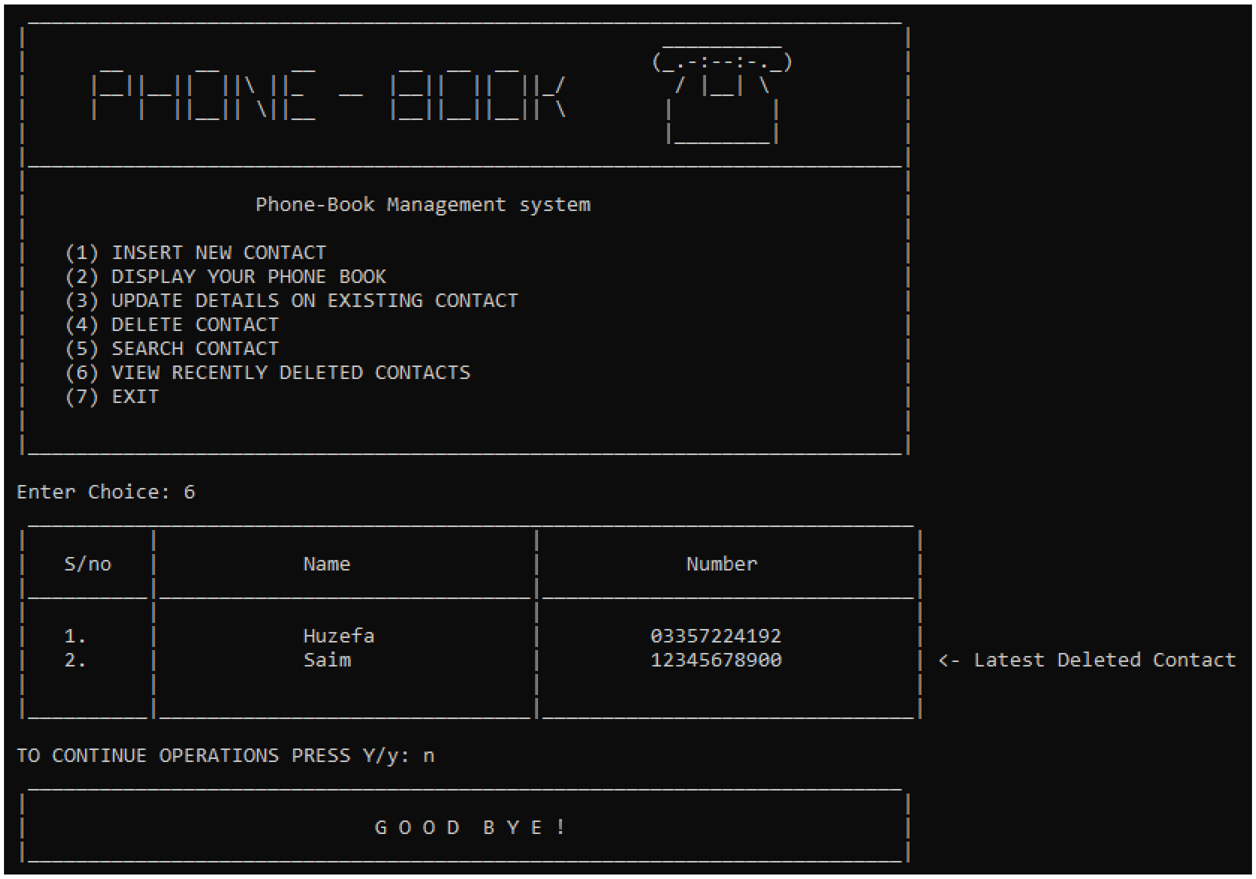
If we select delete operation then it will ask for name and it will search for it and before deleting it, it will push the name and number of that contact in circular queue so we can see the recently deleted contacts later on and then it will ask us if we want to exit or continue doing more operations.

```
void DeleteContact(string deletenamecheck)
{
        struct Node* delnode;
        delnode = Head;
        //traversing unless we get equal to user input
        while (delnode != NULL)
        {
                if (deletenamecheck == delnode->name)
                {
                        break;
                }
                delnode = delnode->next;
        }
        if (delnode == NULL)
        {
                cout << "
___________________________________________________________________________ " << endl;
                cout << " |
|" << endl;
                cout << " |             Entered name is not present in the PhoneBook
|" << endl;
                cout << "
|___________________________________________________________________________|" << endl;
                return;
        }
        //Deleting First Node and there is no node afterwards // single node
        if (delnode == Head && Head->next == NULL)
        {
```

```
			Insert(delnode->name, delnode->number);//pushing data in circular linked
list to store recently deleted data
			free(delnode); //delete the node
			cout << "
_____________________________________________________________________________ " << endl;
			cout << " |
|" << endl;
			cout << " |            Contact is Successfully Deleted from PhoneBook
|" << endl;
			cout << "
|_____________________________________________________________________________|" << endl;
			Head = NULL; //point head to NULL because first and only node is deleted
		}
		//deleting first node
		else if (delnode == Head)
		{
			Head = Head->next; //point head pointer to next node
			Insert(delnode->name, delnode->number);//pushing data in circular linked
list to store recently deleted data
			free(delnode);
			cout << "
_____________________________________________________________________________ " << endl;
			cout << " |
|" << endl;
			cout << " |            Contact is Successfully Deleted from PhoneBook
|" << endl;
			cout << "
|_____________________________________________________________________________|" << endl;
		}
		//Deleting middle node
		else if (delnode != Head && delnode->next != NULL)
		{
			//conditions to connect left and right nodes before deleting middle node
			delnode->next->prev = delnode->prev;
			delnode->prev->next = delnode->next;
			Insert(delnode->name, delnode->number);//pushing data in circular linked
list to store recently deleted data
			free(delnode);
			cout << "
_____________________________________________________________________________ " << endl;
			cout << " |
|" << endl;
			cout << " |            Contact is Successfully Deleted from PhoneBook
|" << endl;
			cout << "
|_____________________________________________________________________________|" << endl;
		}
		//Deleting Last Node
		else if (delnode->next == NULL)
		{
			delnode->prev->next = NULL;//Making second last node the last node by
breaking the link
			Insert(delnode->name, delnode->number);//pushing data in circular linked
list to store recently deleted data
			free(delnode);
			cout << "
_____________________________________________________________________________ " << endl;
```

```
                cout << " |
|" << endl;
                cout << " |              Contact is Successfully Deleted from PhoneBook
|" << endl;
                cout << "
|____________________________________________________________________________|" << endl;
        }
}
```

- **View recently deleted contact**



```
 ___________________________________________________________________________
|                                                           _________       |
|      __    __   __    __   __   __                      (_.-:--:-._)     |
|     |__||__||  ||\ ||__  __  |__||  ||  ||_/             / |__| \        |
|     |   |  ||__|| \||__     |__||__||__|| \             |       |       |
|                                                          |_______|       |
|___________________________________________________________________________|
|                                                                           |
|                   Phone-Book Management system                            |
|                                                                           |
|   (1) INSERT NEW CONTACT                                                  |
|   (2) DISPLAY YOUR PHONE BOOK                                             |
|   (3) UPDATE DETAILS ON EXISTING CONTACT                                  |
|   (4) DELETE CONTACT                                                      |
|   (5) SEARCH CONTACT                                                      |
|   (6) VIEW RECENTLY DELETED CONTACTS                                      |
|   (7) EXIT                                                                |
|                                                                           |
|___________________________________________________________________________|

Enter Choice: 6
 ____________________________________________________________________________
|           |                                |                                |
|   S/no    |              Name              |             Number             |
|___________|________________________________|________________________________|
|           |                                |                                |
|   1.      |            Huzefa              |          03357224192           |
|   2.      |            Saim                |          12345678900           | <- Latest Deleted Contact
|           |                                |                                |
|___________|________________________________|________________________________|

TO CONTINUE OPERATIONS PRESS Y/y: n
 ___________________________________________________________________________
|                                                                           |
|                         G O O D   B Y E !                                 |
|___________________________________________________________________________|
```

Recently deleted contacts shows us the latest 10 contacts we have deleted and if the number exceeds 10 then it removes the oldest deleted contact from list and makes space for recently deleted contact. We have made it possible using a circular queue which dequeue’s automatically when it is full and makes space for new contact. The advantage we got using circular queue was that the queue never got full and whenever it got full it deletes the oldest item and makes space for the new item

We have named circular queue as Infinity Structure as it never gets full and deletes automatically.

```
string InfinityStructureName[10];
string InfinityStructureNumber[10];
```

```
int InfinityStructureSize = 10;
int front = -1;
int rear = -1;

void Insert(string element1, string element2)
{
       if ((rear + 1) % InfinityStructureSize == front)
              // check if structure is full suppose rear points at 9(last element) and
front points
              // at 0(first element) then 9 + 1 % 10 = 0 which is equal to front which
means structure is full
       {
              if (front == rear) //if structure has only one element then delete it and
point it at start
              {
                     front = -1;
                     rear = -1;
              }
              else //if structure have more than one element then suppose if front is
pointing at 9 then after
                     //calculation it will become 9 + 1 % 10 which is equal to 0 now
front will point at start again
              {
                     front = (front + 1) % InfinityStructureSize;
              }
       }
       if (front == -1 && rear == -1) //if structure is empty update front and rear
       {
              front = 0;
              rear = 0;
       }
       else //if structure is not empty then suppose if rear is 9 then after calculation
               //9 + 1 % 10 = 0 so now element will be inserted at start again
       {
              rear = (rear + 1) % InfinityStructureSize;
       }
       InfinityStructureNumber[rear] = element1;
       InfinityStructureName[rear] = element2;
}
void recentlydeleteddisplay()
{
       // Function to display status of Circular Queue
       int i;
       int sno = 1;
       if (front == -1) //if front is -1 which means queue is empty and no contacts are
deleted yet
       {
              cout << "
________________________________________________________________________ " << endl;
              cout << " |
|" << endl;
              cout << " |                         NO CONTACTS ARE DELETED
|" << endl;
              cout << "
|________________________________________________________________________|" << endl;
       }
       else
       {
```

```
		cout << " 
___________________________________________________________________________" << endl;
		cout << " |          |                                |
|" << endl;
		cout << " |   S/no   |            Name                |            Number
|" << endl;
		cout << " 
|__________|________________________________|________________________________| " << endl;
		cout << " |          |                                |
|" << endl;
		for (i = front; i != rear; i = (i + 1) % InfinityStructureSize) //
condition to update front as discussed earlier
		{
			if (InfinityStructureNumber[i].length() > 6)
			{
				cout << " |   " << sno << ".     |\t\t " <<
InfinityStructureNumber[i] << "\t    |\t      " << InfinityStructureName[i] << "\t    |"
<< endl;
			}
			else
			{
				cout << " |   " << sno << ".     |\t\t " <<
InfinityStructureNumber[i] << "\t\t    |\t      " << InfinityStructureName[i] << "\t
|" << endl;
			}
			sno++;
		}
		if (InfinityStructureNumber[i].length() > 6)
		{
			cout << " |   " << sno << ".     |\t\t " <<
InfinityStructureNumber[i] << "\t    |\t      " << InfinityStructureName[i] << "\t    |
<- Latest Deleted Contact" << endl;
		}
		else
		{
			cout << " |   " << sno << ".     |\t\t " <<
InfinityStructureNumber[i] << "\t\t    |\t      " << InfinityStructureName[i] << "\t    |
<- Latest Deleted Contact" << endl;
		}
		cout << " |          |                                |
|" << endl;
		cout << " 
|__________|________________________________|________________________________|" << endl;
	}
}
```