FAIZAN CHOUDHARY

20BCS021

DSA LAB

14th December 2021

CODE: (code pasted in this format for readability)

```
#include <iostream>
using namespace std;

struct list
{
    int info;
    struct list *next;
    struct list *prev;
};
struct list *ptr, *front=NULL, *rear=NULL, *p, *temp;

void create_node (int x)
{
    ptr=(struct list *) malloc (sizeof(struct list));
    if (ptr==NULL)
    {
        cout<<"\nMemory could not be allocated!\n";
        return;
    }
    ptr->info = x;
    ptr->next = NULL;
    ptr->prev = NULL;
}

int isEmpty ()
{
    if (front==NULL || rear==NULL)
        return 1;
    else
        return 0;
}

int size ()
{
    if (isEmpty()==1)
     return 0;
    else
    {
        int count=1;
        for (p=front; p!=rear; p=p->next)
         count++;
        return count;
    }
}

void display ()
{
    if (isEmpty()==1)
     cout<<"\nList is empty! Nothing to display\n";
    else
    {
        p=front;
        cout<<endl<<"NULL <- ";
        while (p->next != NULL)
        {
            cout<<p->info<<" <-> ";
            p=p->next;
        }
        cout<<rear->info;
        cout<<" -> NULL"<<endl;
    }
}

void display_rev ()
{
    if (isEmpty()==1)
     cout<<"\nList is empty! Nothing to display\n";
    else
    {
        p=rear;
        cout<<"NULL <- ";
        while (p->prev != NULL)
        {
            cout<<p->info<<" <-> ";
            p=p->prev;
        }
        cout<<front->info;
        cout<<" -> NULL"<<endl;
    }
}

void insert_beg (int n)
{
    create_node(n);
    if (front==NULL)
    {
        front=rear=ptr;
    }
    else
    {
        ptr->next = front;
        front->prev = ptr;
        ptr->prev = NULL;
        front = ptr;
    }
    display();
```

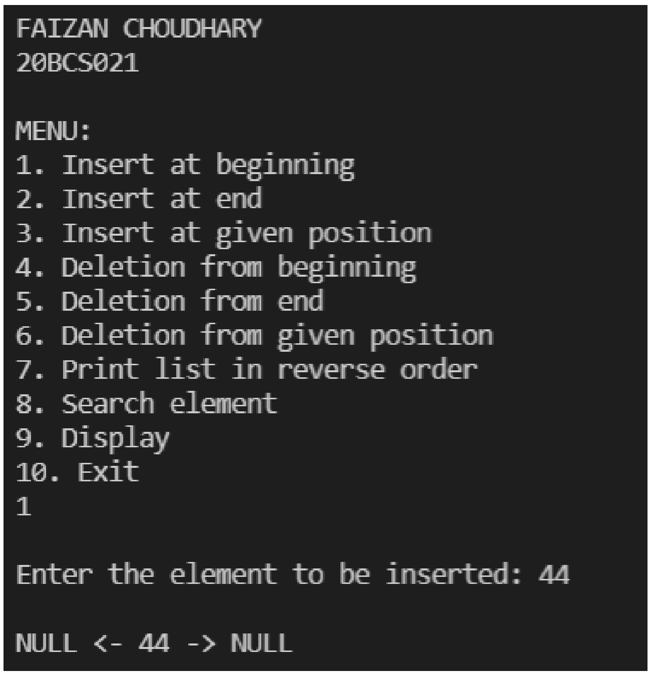
```
}

void insert_end (int n)
{
    create_node(n);
    if (front==NULL)
    {
        front=rear=ptr;
    }
    else
    {
        ptr->prev = rear;
        rear->next = ptr;
        ptr->next = NULL;
        rear = ptr;
    }
    display();
}

void insert_pos (int n, int k)
{
    if (k == 1)
    {
        insert_beg(n);
        return ;
    }
    else if (k > size())
    {
        insert_end(n);
        return ;
    }
    else
    {
        if (size()==0)
        {
            cout<<"\nList is empty, inserting at first position.\n";
            insert_beg(n);
            return ;
        }
        create_node(n);
        p = front;
        k--;
        while (k--)
            p = p->next;
        temp = p->prev;

        temp->next = ptr;
        ptr->prev = temp;
        p->prev = ptr;
        ptr->next = p;
    }
    display();
}
```

```
void del_beg ()
{
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    ptr = front;
    front = front->next;
    if (front != NULL)
        front->prev = NULL;
    cout<<"\nDeleting element: "<<ptr->info<<endl;
    delete ptr;
    display();
}

void del_end ()
{
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    ptr = rear;
    rear = ptr->prev;
    if (rear != NULL)
        rear->next = NULL;
    cout<<"\nDeleting element: "<<ptr->info<<endl;
    delete ptr;
    display();
}

void del_pos (int k) {
    int i=k;
    if (isEmpty()==1)
    {
        cout<<"\nList is empty! Nothing to delete\n";
        return ;
    }
    else if (k == 1) {
        del_beg();
        return ;
    }
    else if (k == size()) {
        del_end();
        return ;
    }
    ptr = front;
    i--;
    while (i-- && ptr != NULL)
        ptr = ptr->next;
    temp = ptr->prev;
    p = ptr->next;
    temp->next = p;
```

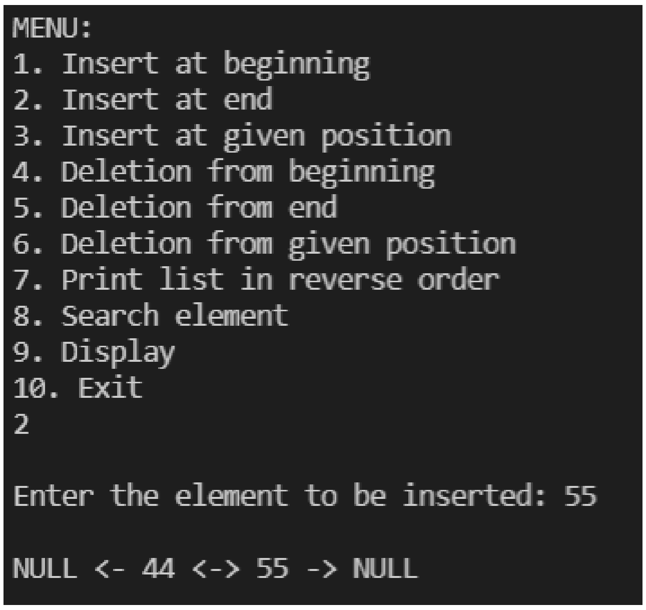
```cpp
    p->prev = temp;
    cout<<"\nDeleting element: "<<ptr->info<<" at position "<<k<<endl;
    delete ptr;
    display();
}

void search (int key)
{
    if (isEmpty()==1)
    {
        cout<<"\nList empty!\n";
        return ;
    }
    bool flag = false;
    int k = 0;
    display();
    p = front;
    while (p != NULL) {
        k++;
        if (p->info == key) {
            flag = true;
            cout<<"\nElement found at position: "<<k<<" !\n";
        }
        p = p->next;
    }
    if (!flag)
        cout<<"\nElement not present in list!\n";
}

int main()
{
    cout<<"\nFAIZAN CHOUDHARY\n20BCS021\n";
    int ch,n,k;

    while (true)
    {
        A:
        cout<<"\nMENU:\n1. Insert at beginning\n2. Insert at end\n3. Insert at given
position\n4. Deletion from beginning\n5. Deletion from end\n6. Deletion from given
position\n7. Print list in reverse order\n8. Search element\n9. Display\n10. Exit\n";
        cin>>ch;
        switch (ch)
        {
        case 1: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                insert_beg(n);
                break;
        case 2: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                insert_end(n);
                break;
        case 3: cout<<"\nEnter the element to be inserted: ";
                cin>>n;
                C:
```

```
                cout<<"\nEnter position: ";
                cin>>k;
                if (k<=0)
                {
                    cout<<"\nEnter valid position!\n";
                    goto C;
                }
                insert_pos(n,k);
                break;
        case 4: del_beg();
                break;
        case 5: del_end();
                break;
        case 6: B:
                cout<<"\nEnter the position to be deleted: ";
                cin>>k;
                if (k<=0)
                {
                    cout<<"\nEnter valid position!\n";
                    goto B;
                }
                del_pos(k);
                break;
        case 7: cout<<"\nList elements: "<<endl;
                display();
                cout<<"\nList elements in reverse order: "<<endl;
                display_rev();
                break;
        case 8: cout<<"\nEnter element to be searched for: ";
                cin>>n;
                search (n);
                break;
        case 9: cout<<"\nList elements: "<<endl;
                display();
                break;
        case 10: exit(0);
        default: cout<<"\nWrong choice! Enter again...\n";
                 goto A;
        }
    }
}
```

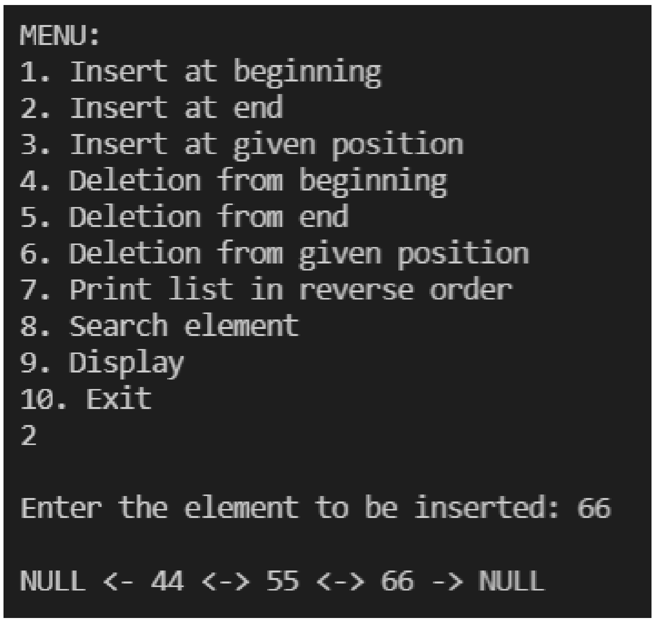
OUTPUT:



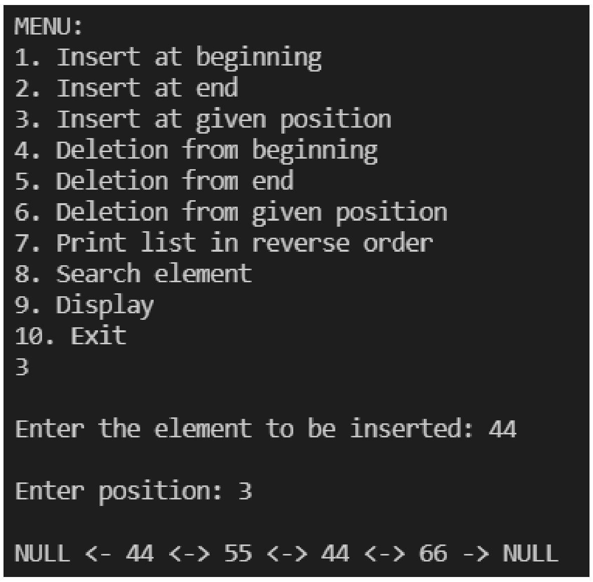
```
FAIZAN CHOUDHARY
20BCS021

MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
1

Enter the element to be inserted: 44

NULL <- 44 -> NULL
```



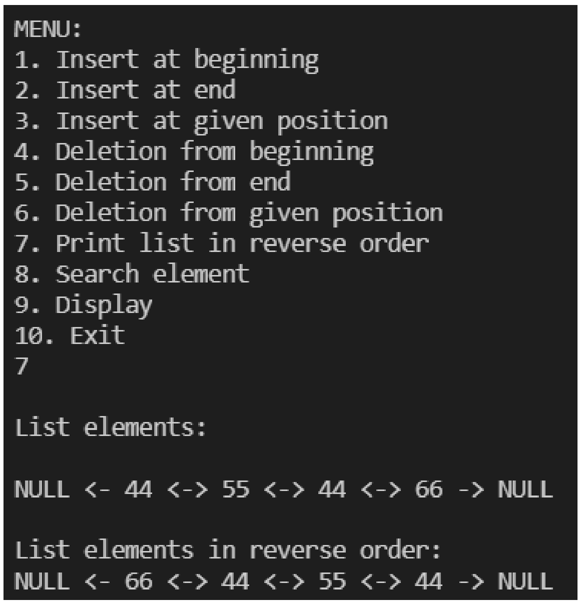
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
2

Enter the element to be inserted: 55

NULL <- 44 <-> 55 -> NULL
```



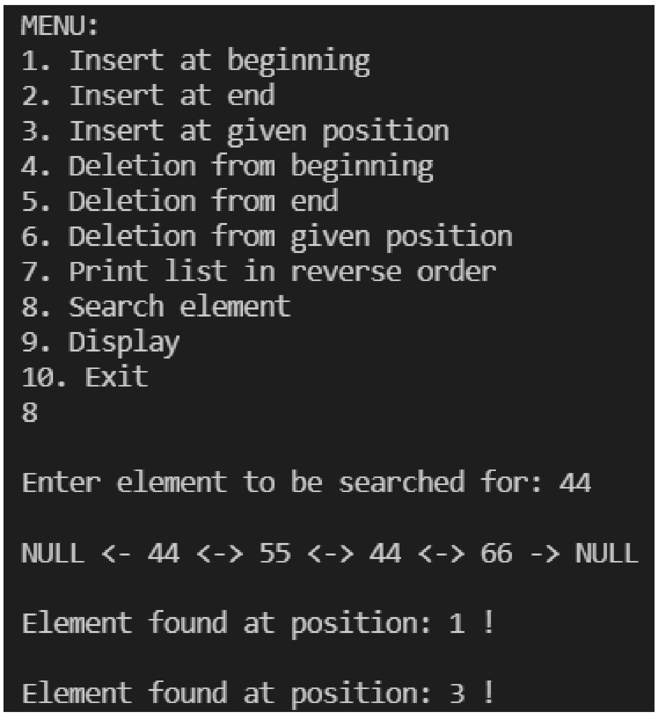
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
2

Enter the element to be inserted: 66

NULL <- 44 <-> 55 <-> 66 -> NULL
```



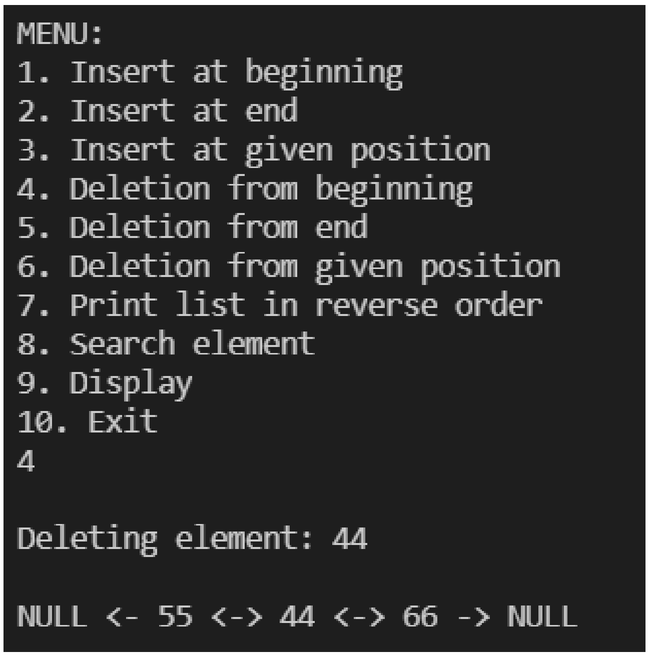
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
3

Enter the element to be inserted: 44

Enter position: 3

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL
```



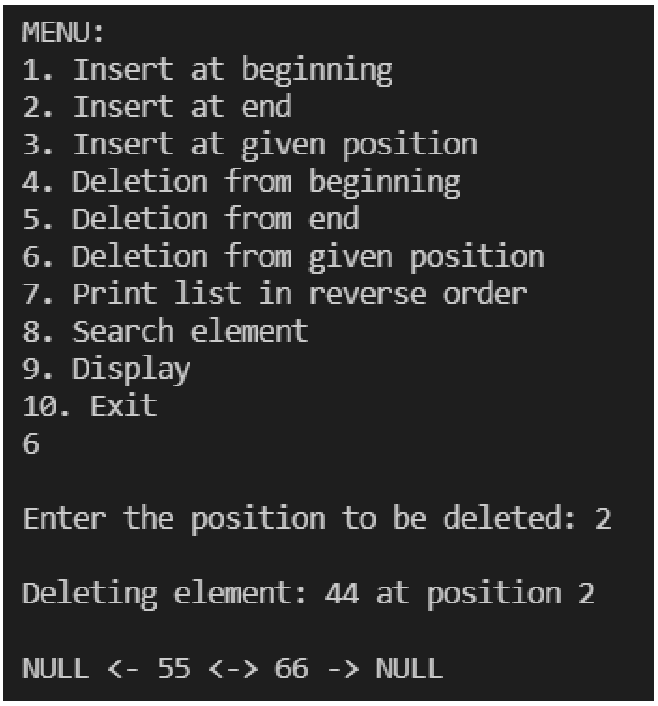
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
7

List elements:

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL

List elements in reverse order:
NULL <- 66 <-> 44 <-> 55 <-> 44 -> NULL
```



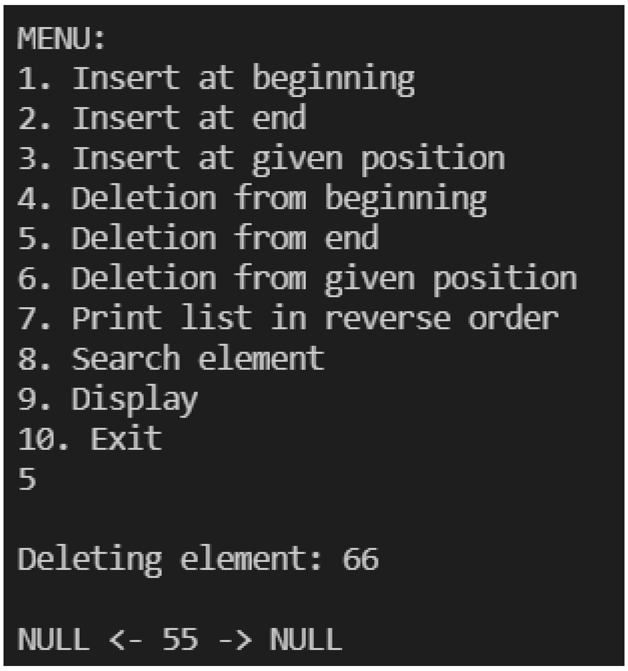
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
8

Enter element to be searched for: 44

NULL <- 44 <-> 55 <-> 44 <-> 66 -> NULL

Element found at position: 1 !

Element found at position: 3 !
```



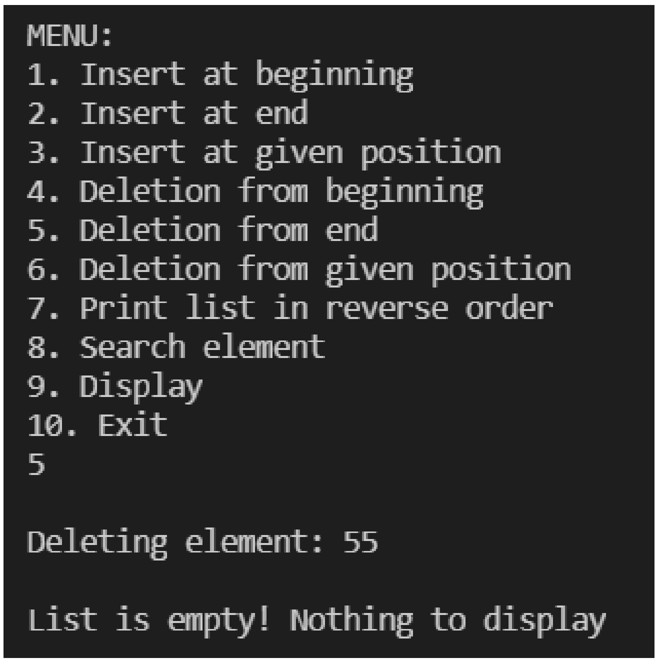
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
4

Deleting element: 44

NULL <- 55 <-> 44 <-> 66 -> NULL
```



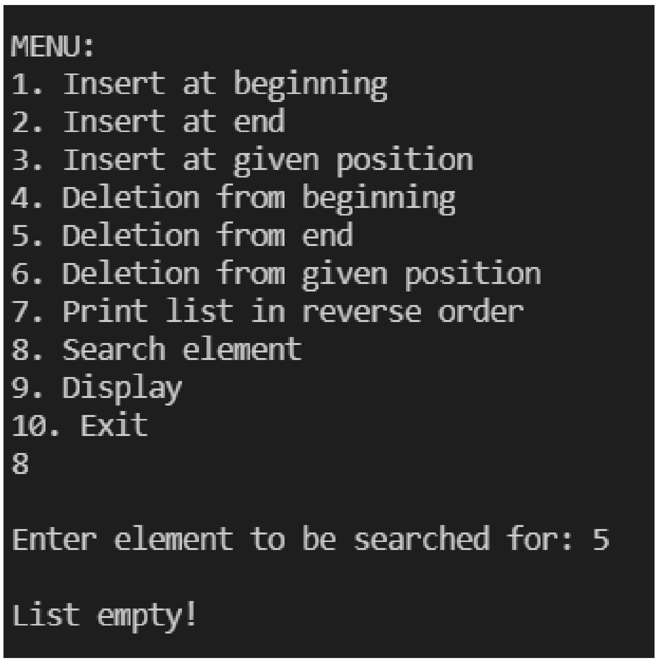
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
6

Enter the position to be deleted: 2

Deleting element: 44 at position 2

NULL <- 55 <-> 66 -> NULL
```



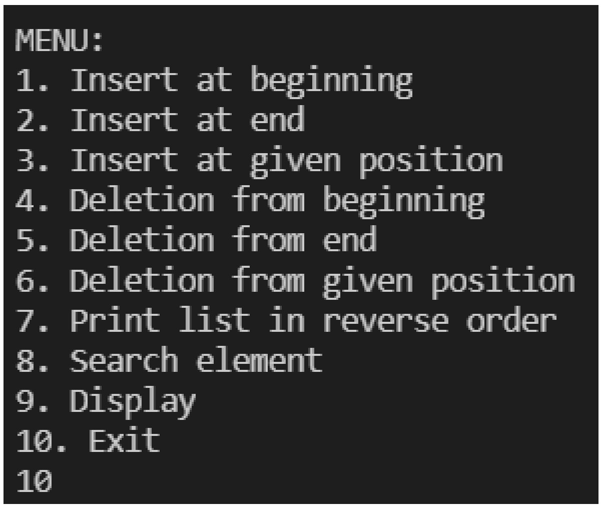
```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
5

Deleting element: 66

NULL <- 55 -> NULL
```



```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
5

Deleting element: 55

List is empty! Nothing to display
```



```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
8

Enter element to be searched for: 5

List empty!
```



```
MENU:
1. Insert at beginning
2. Insert at end
3. Insert at given position
4. Deletion from beginning
5. Deletion from end
6. Deletion from given position
7. Print list in reverse order
8. Search element
9. Display
10. Exit
10
```