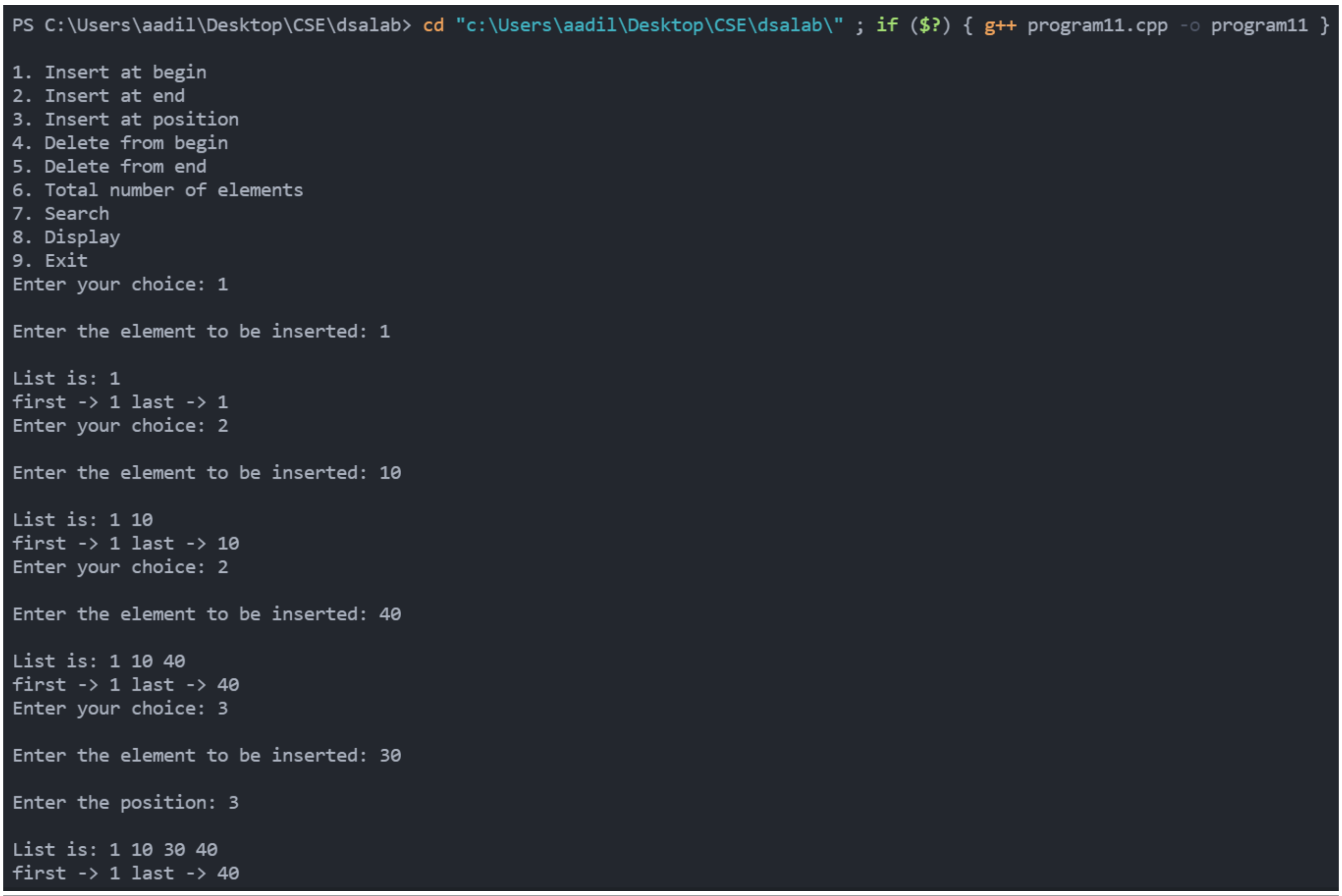
```
#include <iostream>
using namespace std;
struct node
{
    int data;
    node *next;
} *first = NULL, *last = NULL;

int count = 0;
void insertatbegin(int x)
{
    node *temp = new node;
    temp->data = x;
    temp->next = NULL;
    if (first == NULL)
    {
        first = temp;
        last = temp;
        count++;
    }
    else
    {
        temp->next = first;
        first = temp;
        count++;
    }
}
void insertatend(int x)
{
    node *temp = new node;
    temp->data = x;
    temp->next = NULL;
    if (first == NULL)
    {
        first = temp;
        last = temp;
        count++;
    }
    else
    {
        last->next = temp;
        last = temp;
        count++;
    }
}
```

```cpp
void insertatposition(int x, int pos)
{
    if (pos == 1)
    {
        insertatbegin(x);
    }
    else if (pos == count + 1)
    {
        insertatend(x);
    }
    else if (pos <= count)
    {
        node *temp = new node;
        temp->data = x;
        temp->next = NULL;
        node *p = first;
        for (int i = 1; i < pos - 1; i++)
        {
            p = p->next;
        }
        temp->next = p->next;
        p->next = temp;
        count++;
    }
    else
    {
        cout << "\nInvalid position";
    }
}
int deletefrombegin()
{
    if (first == NULL)
    {
        cout << "\nList is empty";
        return -1;
    }
    else
    {
        node *temp = first;
        first = first->next;
        int x = temp->data;
        delete temp;
        count--;
        return x;
    }
}
```

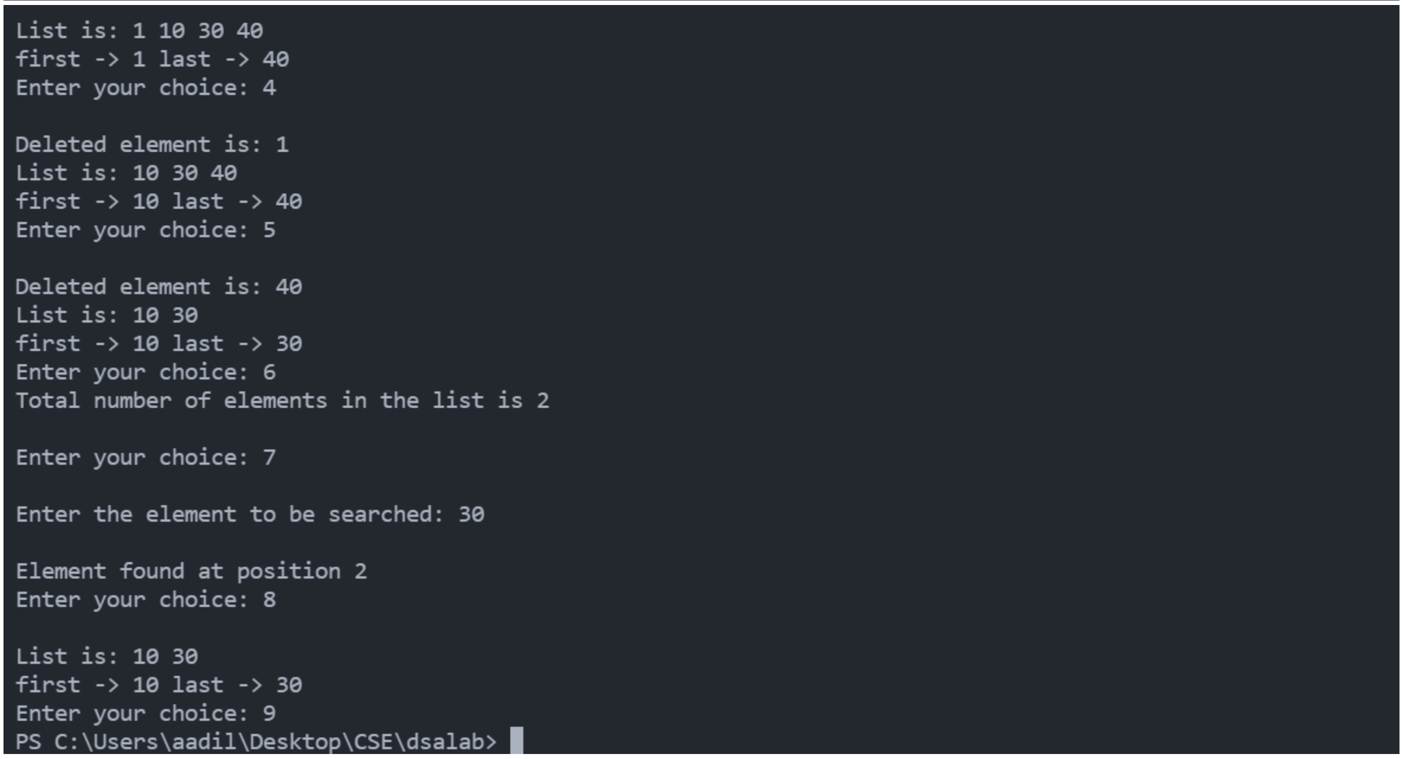
```
int deletefromend()
{
    if (first == NULL)
    {
        cout << "\nList is empty";
        return -1;
    }
    else
    {
        node *temp = first;
        while (temp->next != last)
        {
            temp = temp->next;
        }
        int x = last->data;
        last = temp;
        delete last->next;
        last->next = NULL;
        count--;
        return x;
    }
}
void search(int x)
{
    node *p = first;
    int pos = 1;
    while (p != NULL)
    {
        if (p->data == x)
        {
            cout << "\nElement found at position " << pos;
            return;
        }
        p = p->next;
        pos++;
    }
    cout << "\nElement not found";
}
void display()
{
    node *p = first;
    if (first == NULL)
    {
        cout << "\nList is empty";
    }
```

```cpp
    else
    {
        cout << "\nList is: ";
        while (p != NULL)
        {
            cout << p->data << " ";
            p = p->next;
        }
        cout << "\nfirst -> " << first->data << " last -> " << last->data;
    }
}
int main()
{
    int choice, x, pos;
    cout << "\n1. Insert at begin"
         << "\n2. Insert at end"
         << "\n3. Insert at position"
         << "\n4. Delete from begin"
         << "\n5. Delete from end"
         << "\n6. Total number of elements"
         << "\n7. Search"
         << "\n8. Display"
         << "\n9. Exit";
    while (1)
    {
        cout << "\nEnter your choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            insertatbegin(x);
            display();
            break;
        case 2:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            insertatend(x);
            display();
            break;
        case 3:
            cout << "\nEnter the element to be inserted: ";
            cin >> x;
            cout << "\nEnter the position: ";
            cin >> pos;
            insertatposition(x, pos);
            display();
            break;
        case 4:
            cout << "\nDeleted element is: " << deletefrombegin();
            display();
            break;
        case 5:
            cout << "\nDeleted element is: " << deletefromend();
            display();
            break;
        case 6:
            cout << "Total number of elements in the list is " << count <<
endl;
            break;
        case 7:
            cout << "\nEnter the element to be searched: ";
            cin >> x;
            search(x);
            break;
        case 8:
            display();
            break;
        case 9:
            return 0;
        default:
            cout << "\nInvalid choice";
        }
    }
    return 0;
}
```

OUTPUT:



```
PS C:\Users\aadil\Desktop\CSE\dsalab> cd "c:\Users\aadil\Desktop\CSE\dsalab\" ; if ($?) { g++ program11.cpp -o program11 }

1. Insert at begin
2. Insert at end
3. Insert at position
4. Delete from begin
5. Delete from end
6. Total number of elements
7. Search
8. Display
9. Exit
Enter your choice: 1

Enter the element to be inserted: 1

List is: 1
first -> 1 last -> 1
Enter your choice: 2

Enter the element to be inserted: 10

List is: 1 10
first -> 1 last -> 10
Enter your choice: 2

Enter the element to be inserted: 40

List is: 1 10 40
first -> 1 last -> 40
Enter your choice: 3

Enter the element to be inserted: 30

Enter the position: 3

List is: 1 10 30 40
first -> 1 last -> 40
```



```
List is: 1 10 30 40
first -> 1 last -> 40
Enter your choice: 4

Deleted element is: 1
List is: 10 30 40
first -> 10 last -> 40
Enter your choice: 5

Deleted element is: 40
List is: 10 30
first -> 10 last -> 30
Enter your choice: 6
Total number of elements in the list is 2

Enter your choice: 7

Enter the element to be searched: 30

Element found at position 2
Enter your choice: 8

List is: 10 30
first -> 10 last -> 30
Enter your choice: 9
PS C:\Users\aadil\Desktop\CSE\dsalab>
```