Practical 05

Abhishek Gupta 2019450017

5.1 Singly Linked List_17.cpp

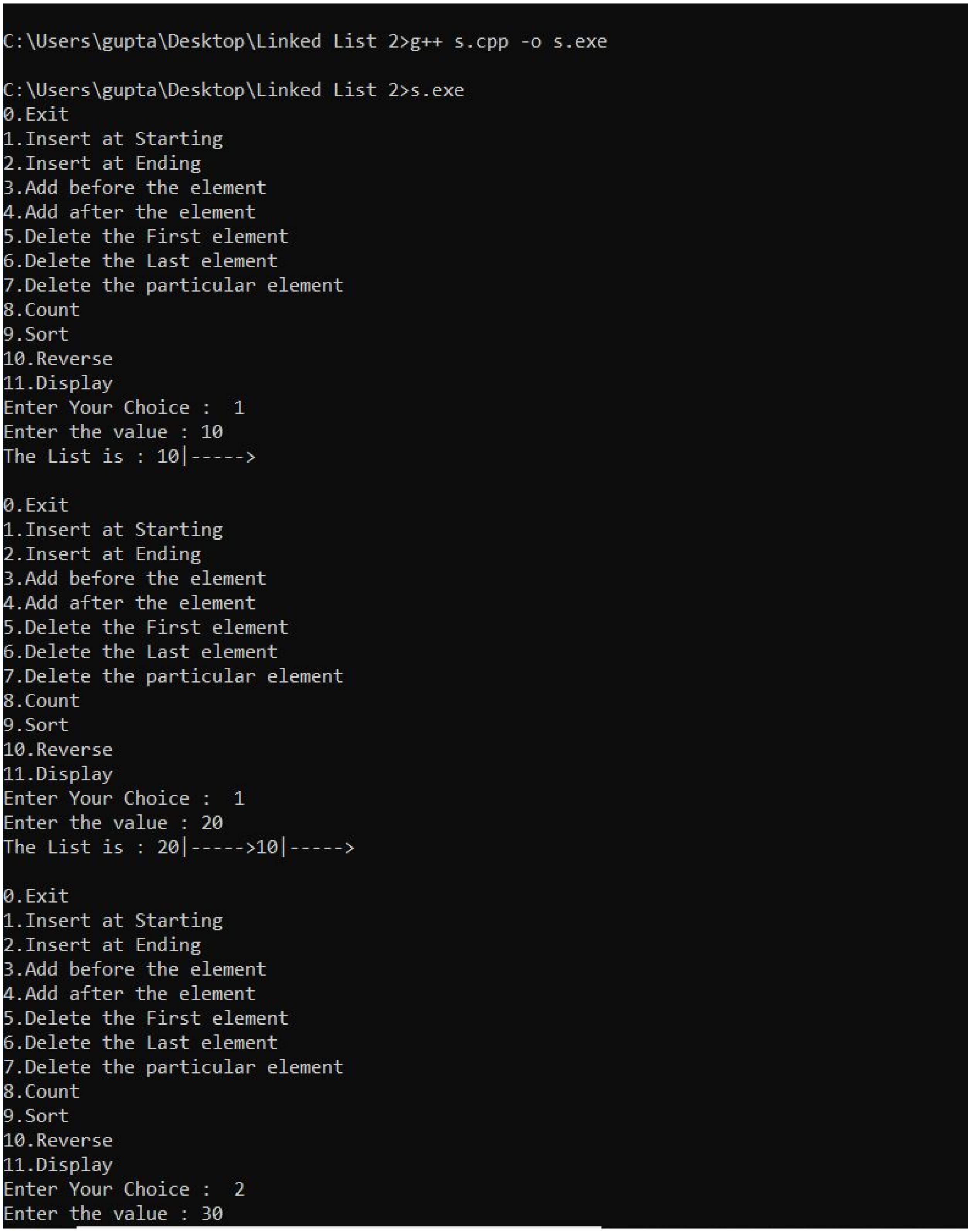
```
#include<iostream>
#include<stdlib.h>
using namespace std;

struct node
{
    int data;
    struct node *next;
}
*list=NULL,*p,*s,*q,*r,*temp;  //*p is used for new node

class SingleLinkList
{
public:
int choice,value;

void get()
{
    do
            {
                cout<<"0.Exit\n1.Insert at Starting\n2.Insert at
Ending\n3.Add before the element\n4.Add after the
element\n5.Delete the First element\n6.Delete the Last
element\n7.Delete the particular
element\n8.Count\n9.Sort\n10.Reverse\n11.Display\n";
                cout<<"Enter Your Choice : "<<" ";
                cin>>choice;
                switch(choice)
                {
                    case 0:
                    break;

                    case 1:
                        insert_start();
                        break;

                    case 2:
                        insert_end();
                        break;

                    case 3:
                        before_add();
                        break;

                    case 4:
                        after_add();
                        break;

                    case 5:
                        delete_start();
                        break;

                    case 6:
                        delete_end();
                        break;

                    case 7:
                        delete_ele();
                        break;
                    case 8:
                        count_ele();
                        break;

                    case 9:
                        sort_ele();
                        break;

                    case 10:
                        reverse_ele();
                        break;

                    case 11:
                        display();
                        break;

                    default:
                        cout<<"invalid input"<<endl<<endl;
                }
            }while(choice!=0);

}

void insert_start()
{
            cout<<"Enter the value : ";
            cin>>value;
            p=(struct node*)malloc(sizeof(node));
            p->data=value;
            if(list == NULL)
            {
                    p->next=NULL;
                    list=p;
                    display();
            }
            else
            {
                p->next=list;
                list=p;
                display();
            }
}

void insert_end()
{
            cout<<"Enter the value : ";
            cin>>value;
            p=(struct node*)malloc(sizeof(node));
            p->data=value;
            if(list == NULL)
            {
                p->next=NULL;
                list=p;
                display();
            }
            else
            {
                q=list;
                while(q->next != NULL)
                {
                    q=q->next;
                }
                q->next=p;
                p->next=NULL;
                display();
            }
}

void before_add()
{
            int before,count=0;
            cout<<"Enter Before Value : ";
            cin>>before;
            if(list==NULL)
            {
                cout<<"The Number is Not Present";
            }
            else
            {
                q=(struct node*)malloc(sizeof(node));
                cout<<"Enter Value : ";
                cin>>value;
                q->data=value;
                p=list;
                while(p != NULL)
                {
                    if(p->data == before)
                    break;

                    r=p;
                    p=p->next;
                    count++;
                }
                if(count ==0)
                {
                    q->next=p;
                    list=q;
                }
                else
                {
                    r->next=q;
                q->next=p;
                }
                display();
            }
}

void after_add()
{
            int after;
            cout<<"Enter After Value : ";
            cin>>after;
            if(list==NULL)
            {
                cout<<"The Number is Not Present";
            }
            else
            {
                q=(struct node*)malloc(sizeof(node));
                cout<<"Enter Value : ";
                cin>>value;
                q->data=value;
                p=list;
                while(p != NULL)
                {
                    if(p->data == after)
                    break;

                    p=p->next;
                }
                r=p->next;
                p->next=q;
                q->next=r;
                display();
            }
}

void delete_start()
{
        cout<<"Delete Fisrt element "<<endl;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else if (list->next == NULL)
        {
            list=NULL;
        }
        else
        {
            p=list;
            list=list->next;
            delete p;
        }
        display();
```

```
}

void delete_end()
{

        cout<<"Delete Last element "<<endl;
        p=list;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else if (list->next == NULL)
        {
            list=NULL;
        }
        else
        {
            while (p->next->next != NULL)
            p = p->next;
            delete (p->next);
            p->next = NULL;
        }
        display();
}

void delete_ele()
{
        int del;
        cout<<"Enter Element to be deleted : ";
        cin>>del;
        p=list;
        if(list == NULL)
        {
            cout<<"Empty List";
        }
        else if (list->data == del)
        {
            q=list;
            list=list->next;
            delete q;
        }
        else
        {
            while(p != NULL)
            {
                if(p->data == del)
                break;

                r=p;
                p=p->next;
            }
            q=p;
            p=p->next;
            r->next=p;
            delete q;
        }
        display();
}

void count_ele()
{
    int c=0;
    p=list;
    while(p != NULL)
    {
        p=p->next;
        c++;
    }
    cout<<"The Number of Elements is : "<<c<<endl<<endl;
}

void sort_ele()
{
        cout<<"Sorted List "<<endl;
        q=list;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else
        {
            while(q!= NULL)
            {
                r=q->next;
                while(r!= NULL)
                {
                    if(r->data < q->data)
                    swap(r->data,q->data);

                    r=r->next;
                }
                q=q->next;
            }
            display();
        }

}

  void reverse_ele()
    {
        q=p=list;
        temp=NULL;
        while(q!=NULL)
        {
            q=p->next;
            p->next=temp;
            temp=p;
            p=q;
        }
        list=temp;
        display();
    }

void display()
{
    if(list==NULL)
            {
                cout<<endl<<"List is Empty "<<endl<<endl;
            }
            else
            {
                cout<<"The List is : ";
                q=list;
                while(q !=NULL)
                {
                    cout<<q->data<<"|----->";
                    q=q->next;
                }
                cout<<endl<<endl;
            }
}
};

int main()
{
    SingleLinkList s;
    s.get();
    return 0;
}
```

Output :

Practical 05

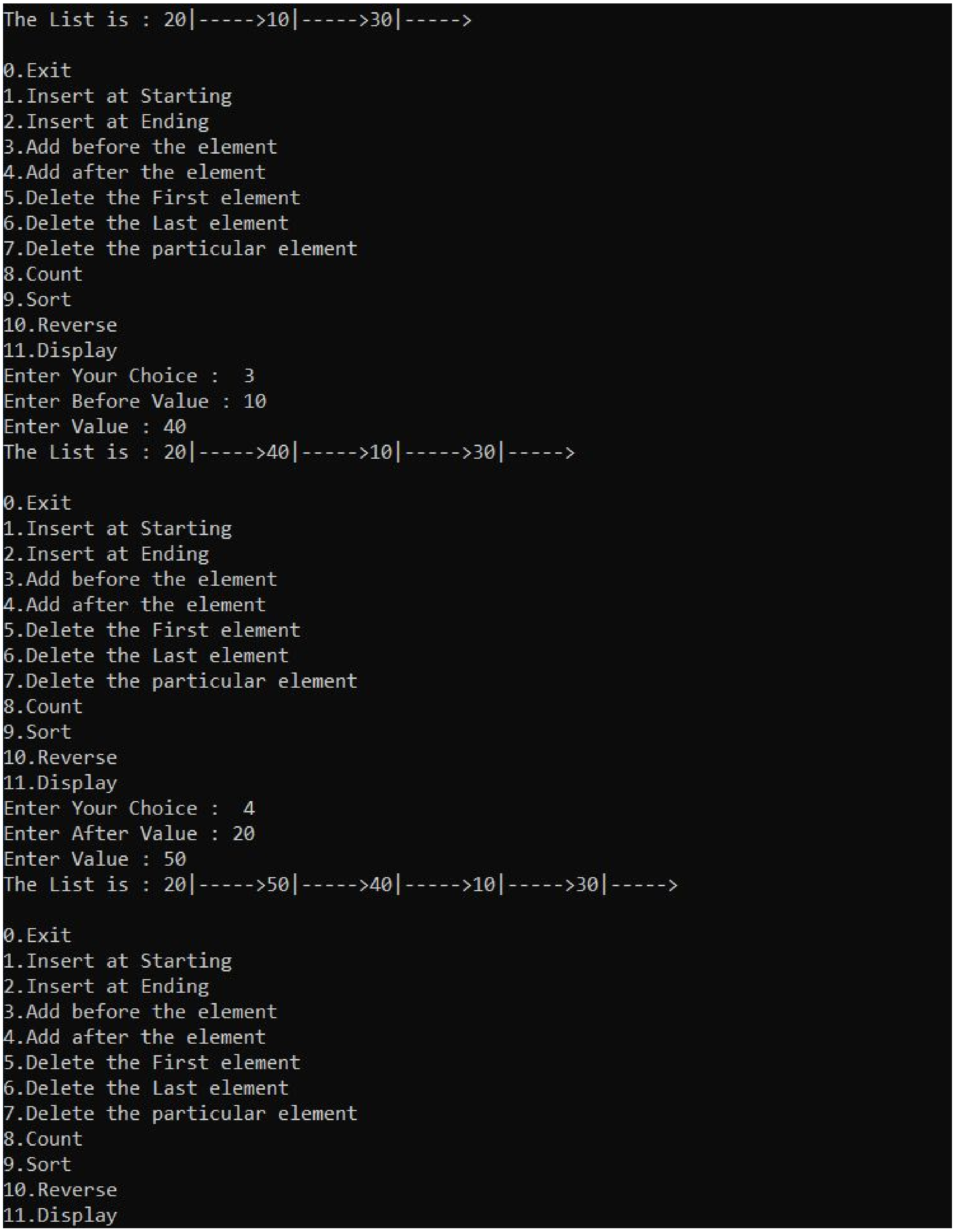
Abhishek Gupta **2019450017**



```
C:\Users\gupta\Desktop\Linked List 2>g++ s.cpp -o s.exe

C:\Users\gupta\Desktop\Linked List 2>s.exe
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  1
Enter the value : 10
The List is : 10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  1
Enter the value : 20
The List is : 20|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  2
Enter the value : 30
```

Practical 05

Abhishek Gupta 2019450017

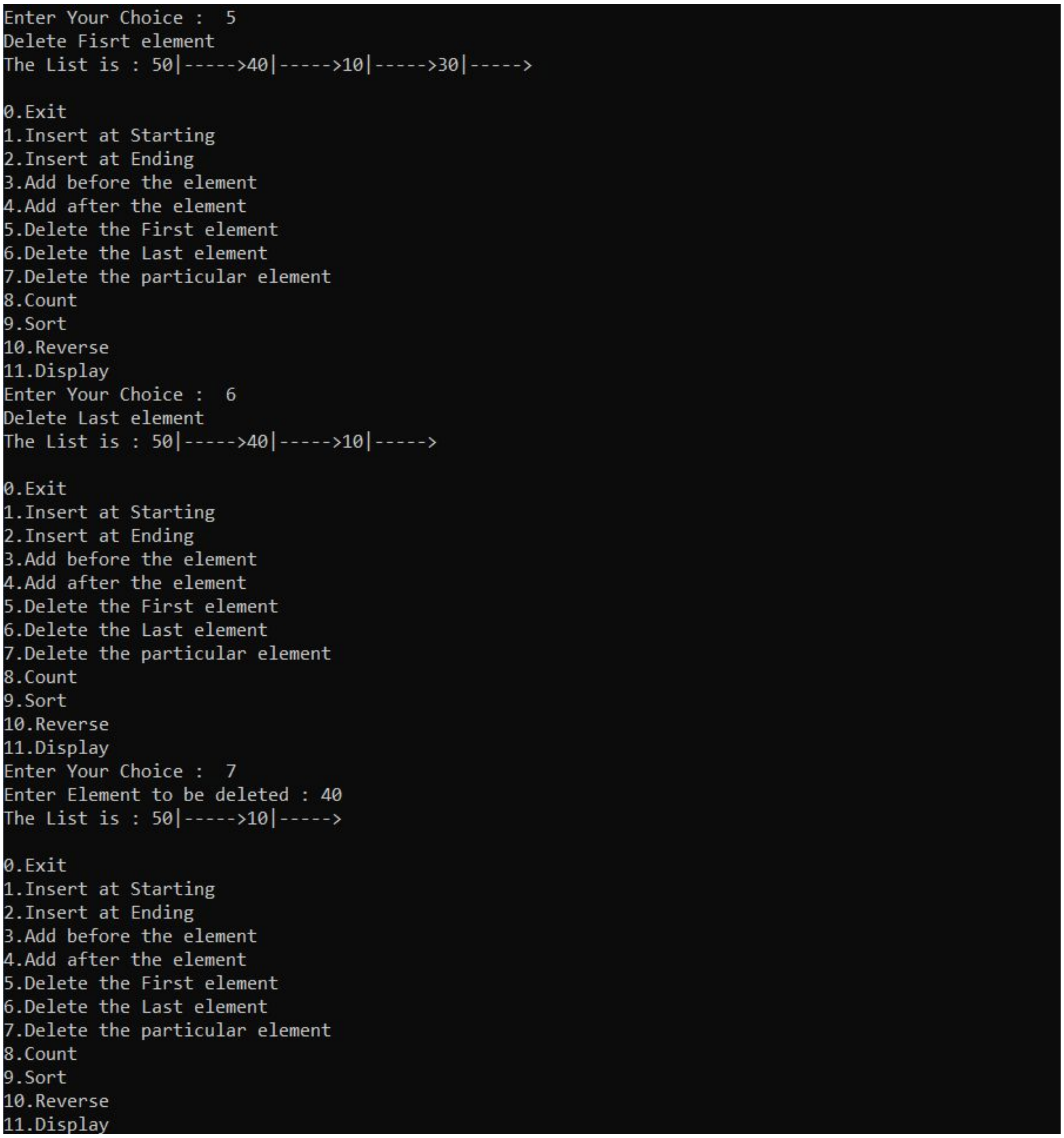


```
The List is : 20|----->10|----->30|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  3
Enter Before Value : 10
Enter Value : 40
The List is : 20|----->40|----->10|----->30|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  4
Enter After Value : 20
Enter Value : 50
The List is : 20|----->50|----->40|----->10|----->30|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
```

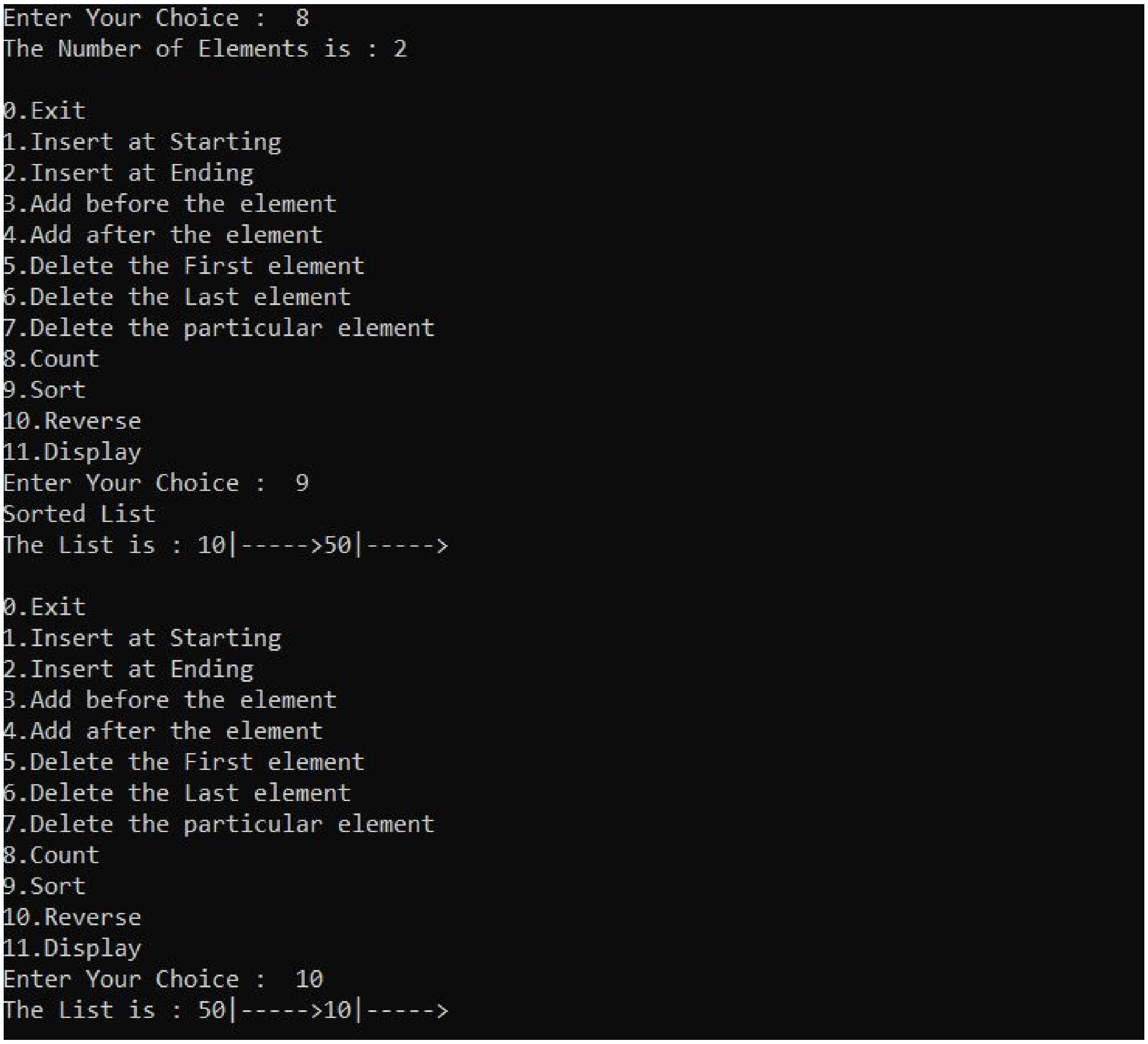
Linked List

Practical 05

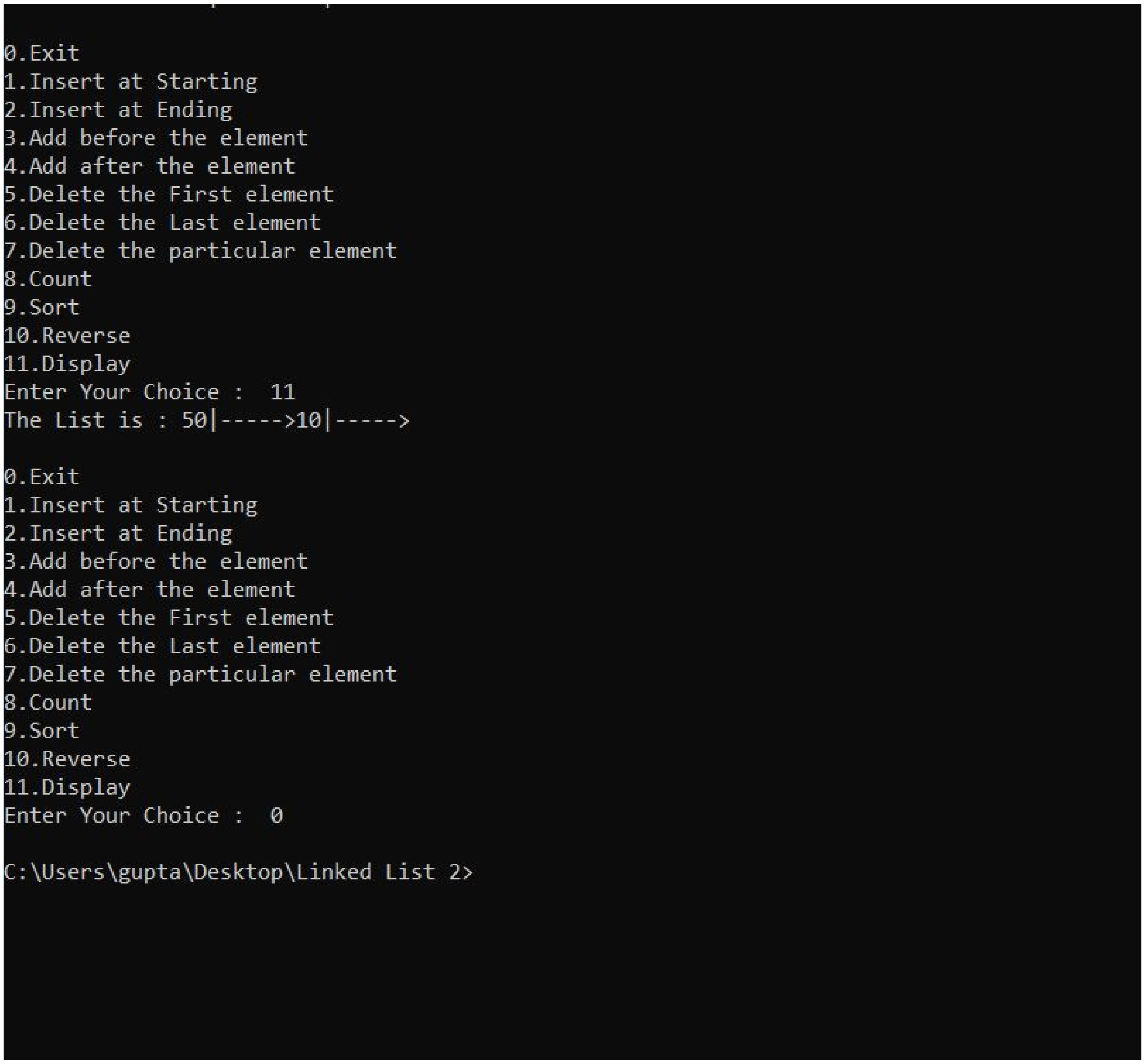
Abhishek Gupta **2019450017**



```
Enter Your Choice :  5
Delete Fisrt element
The List is : 50|----->40|----->10|----->30|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  6
Delete Last element
The List is : 50|----->40|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  7
Enter Element to be deleted : 40
The List is : 50|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
```



```
Enter Your Choice :  8
The Number of Elements is : 2

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  9
Sorted List
The List is : 10|----->50|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  10
The List is : 50|----->10|----->
```



```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  11
The List is : 50|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  0

C:\Users\gupta\Desktop\Linked List 2>
```

5.2 Doubly Linked List_17.cpp

```
#include<iostream>
#include<stdlib.h>
using namespace std;

struct node
{
    int data;
    struct node *lptr;
    struct node *rptr;
}
*list=NULL,*p,*s,*q,*r,*temp;

class DoubleLinkList
{
public:
int choice,value;

void get()
{
    do
            {
                cout<<"0.Exit\n1.Insert at Starting\n2.Insert at 
Ending\n3.Add before the element\n4.Add after the 
element\n5.Delete the First element\n6.Delete the Last 
element\n7.Delete the particular 
element\n8.Count\n9.Sort\n10.Reverse\n11.Display\n";
                cout<<"Enter Your Choice : "<<" ";
                cin>>choice;
                switch(choice)
                {
```

```
        case 0:
        break;

        case 1:
            insert_start();
            break;

        case 2:
            insert_end();
            break;

        case 3:
            before_add();
            break;

        case 4:
            after_add();
            break;

        case 5:
            delete_start();
            break;

        case 6:
            delete_end();
            break;

        case 7:
            delete_ele();
            break;
```

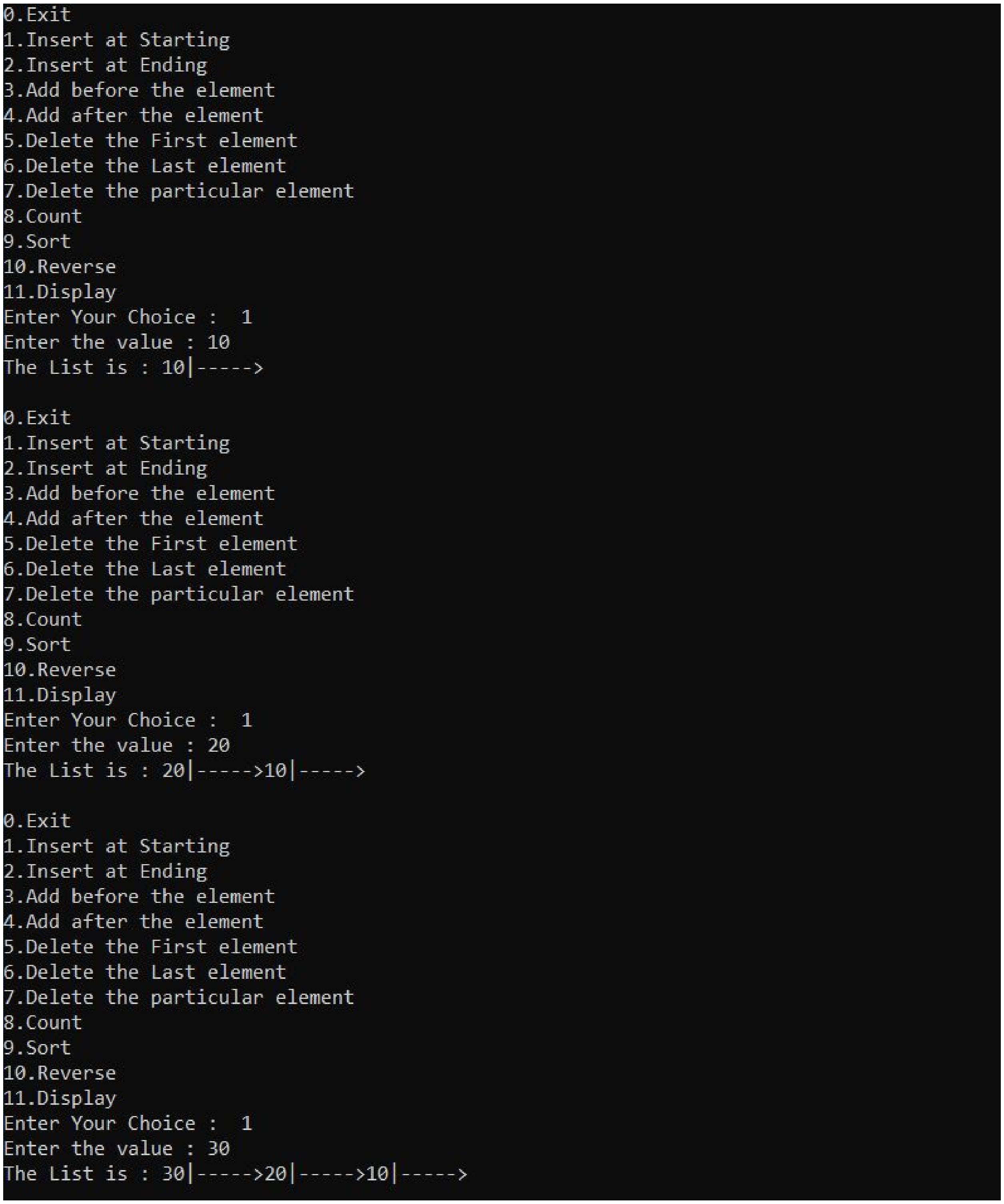
```
                    case 8:
                        count_ele();
                        break;

                    case 9:
                        sort_ele();
                        break;

                    case 10:
                        reverse_ele();
                        break;

                    case 11:
                        display();
                        break;

                    default:
                        cout<<"invalid input"<<endl<<endl;
                }
            }while(choice!=0);

}

void insert_start()
{
            cout<<"Enter the value : ";
            cin>>value;
            p=(struct node*)malloc(sizeof(node));
            p->data=value;
            if(list == NULL)
            {
                    p->lptr=NULL;
                    p->rptr=NULL;
                    list=p;
                    display();
            }
            else
            {
                q=list;
                p->lptr=NULL;
                p->rptr=list;
                q->lptr=p;
                list=p;
                display();
            }
}

void insert_end()
{
            cout<<"Enter the value : ";
            cin>>value;
            p=(struct node*)malloc(sizeof(node));
            p->data=value;
            if(list == NULL)
            {
                    p->lptr=NULL;
                    p->lptr=NULL;
                    list=p;
                    display();
            }
            else
            {
                q=list;
                while(q->rptr != NULL)
                {
                    q=q->rptr;
                }
                q->rptr=p;
                p->lptr=q;
                p->rptr=NULL;
                display();
            }
}

void before_add()
{
            int before,count=0;
            cout<<"Enter Before Value : ";
            cin>>before;
            if(list==NULL)
            {
                cout<<"The Number is Not Present";
            }
            else
            {
                q=(struct node*)malloc(sizeof(node));
                cout<<"Enter Value : ";
                cin>>value;
                q->data=value;
                p=list;
                while(p != NULL)
                {
                    if(p->data == before)
                    break;

                    r=p;
                    p=p->rptr;

                    count++;
                }
                if(count ==0)
                {
                    q->lptr=NULL;
                    q->rptr=list;
                    list=q;
                }
                else
                {
                q->lptr=r;
                r->rptr=q;
                q->rptr=p;
                p->lptr=q;
                }
                display();
            }
}

void after_add()
{
            int after;
            cout<<"Enter After Value : ";
            cin>>after;
            if(list==NULL)
            {
                cout<<"The Number is Not Present";
            }
            else
            {
               q=(struct node*)malloc(sizeof(node));
                cout<<"Enter Value : ";
                cin>>value;
                q->data=value;
                p=list;
                while(p != NULL)
                {
                    if(p->data == after)
                    break;
                    p=p->rptr;
                }
                if(p->rptr == NULL)
                {
                    p->rptr=q;
                    q->lptr=p;
                    q->rptr=NULL;
                }
                else
                {
                   s=p->rptr;
                   p->rptr=q;
                   q->rptr=s;
                   q->lptr=p;
                   s->lptr=q;
                }
                display();
            }
```

```
}

void delete_start()
{
        cout<<"Delete Fisrt element "<<endl;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else if (list->rptr == NULL)
        {
            list=NULL;
        }
        else
        {
            p=list;
            list=list->rptr;
            list->lptr=NULL;
            delete p;
        }
        display();
}

void delete_end()
{
        cout<<"Delete Last element "<<endl;
        p=list;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else if (list->lptr == NULL && list->rptr == NULL)
        {
            list=NULL;
        }
        else
        {
        while(p->rptr != NULL)
        {
            r=p;
            p=p->rptr;
        }
        delete(r->rptr);
        r->rptr=NULL;
        }
        display();
}

void delete_ele()
{
        int del;
        cout<<"Enter Element to be deleted : ";
        cin>>del;
        p=list;
        if(list == NULL)
        {
            cout<<"Empty List";
        }
        else if (p->data == del)
        {
            q=list;
            list=list->rptr;
            list->lptr=NULL;
            delete q;
        }
        else
        {
            while(p->data !=del)
            {
                q=p;
                p=p->rptr;
            }
            if(p->rptr == NULL)
            {
                delete p;
                q->rptr=NULL;
            }
            else
            {
                s=p->rptr;
                q->rptr=s;
                s->lptr=q;
            }
        }
            display();
}

int count_ele()
{
    int c=0;
    p=list;
    while(p != NULL)
    {
        p=p->rptr;
        c++;
    }
    cout<<"The Number of Elements is : "<<c<<endl<<endl;
    return c;
}

void sort_ele()
{
        cout<<"Sorted List "<<endl;
        q=list;
        if(list == NULL)
        {
            cout<<"Empty List"<<endl<<endl;
        }
        else
        {
            while(q!= NULL)
            {
                r=q->rptr;
                while(r!= NULL)
                {
                    if(r->data < q->data)
                    swap(r->data,q->data);

                    r=r->rptr;
                }
                q=q->rptr;
            }
        }
        display();
}

void reverse_ele()
{
    q=list;
        while(q!=NULL)
        {
            r=q;
            temp=q->rptr;
            q->rptr=q->lptr;
            q->lptr=temp;
            q=temp;
        }
        list=r;
        display();
}

void display()
{
    if(list==NULL)
            {
                cout<<endl<<"List is Empty "<<endl<<endl;
            }
            else
            {
                cout<<"The List is : ";
                q=list;
                while(q !=NULL)
                {
                    cout<<q->data<<"|----->";
                    q=q->rptr;
                }
                cout<<endl<<endl;
            }
}
};

int main()
{
    DoubleLinkList d;
    d.get();
    return 0;
}
```

Output :

Practical 05

Abhishek Gupta 2019450017

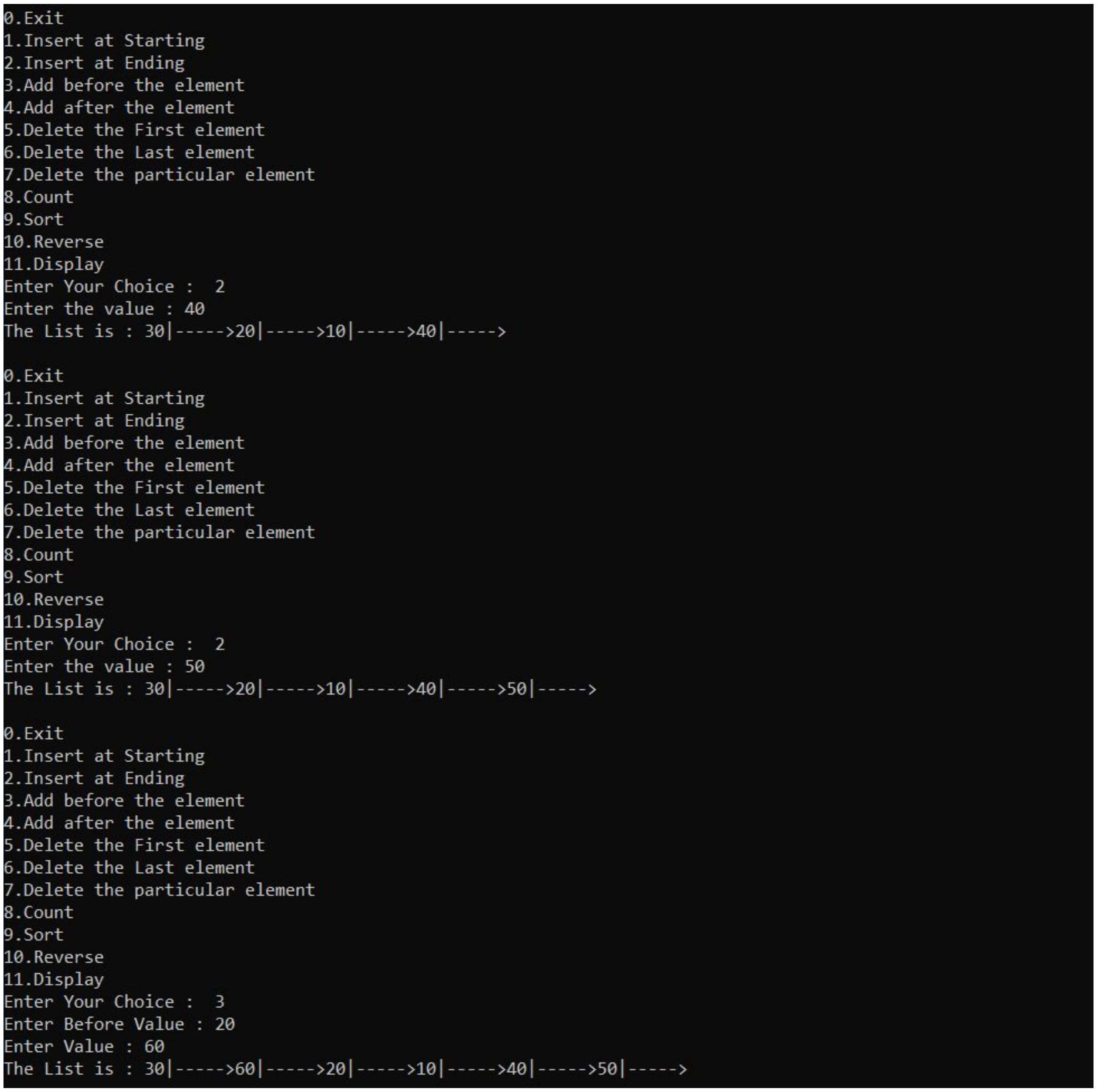


```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  1
Enter the value : 10
The List is : 10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  1
Enter the value : 20
The List is : 20|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  1
Enter the value : 30
The List is : 30|----->20|----->10|----->
```

Linked List

Practical 05

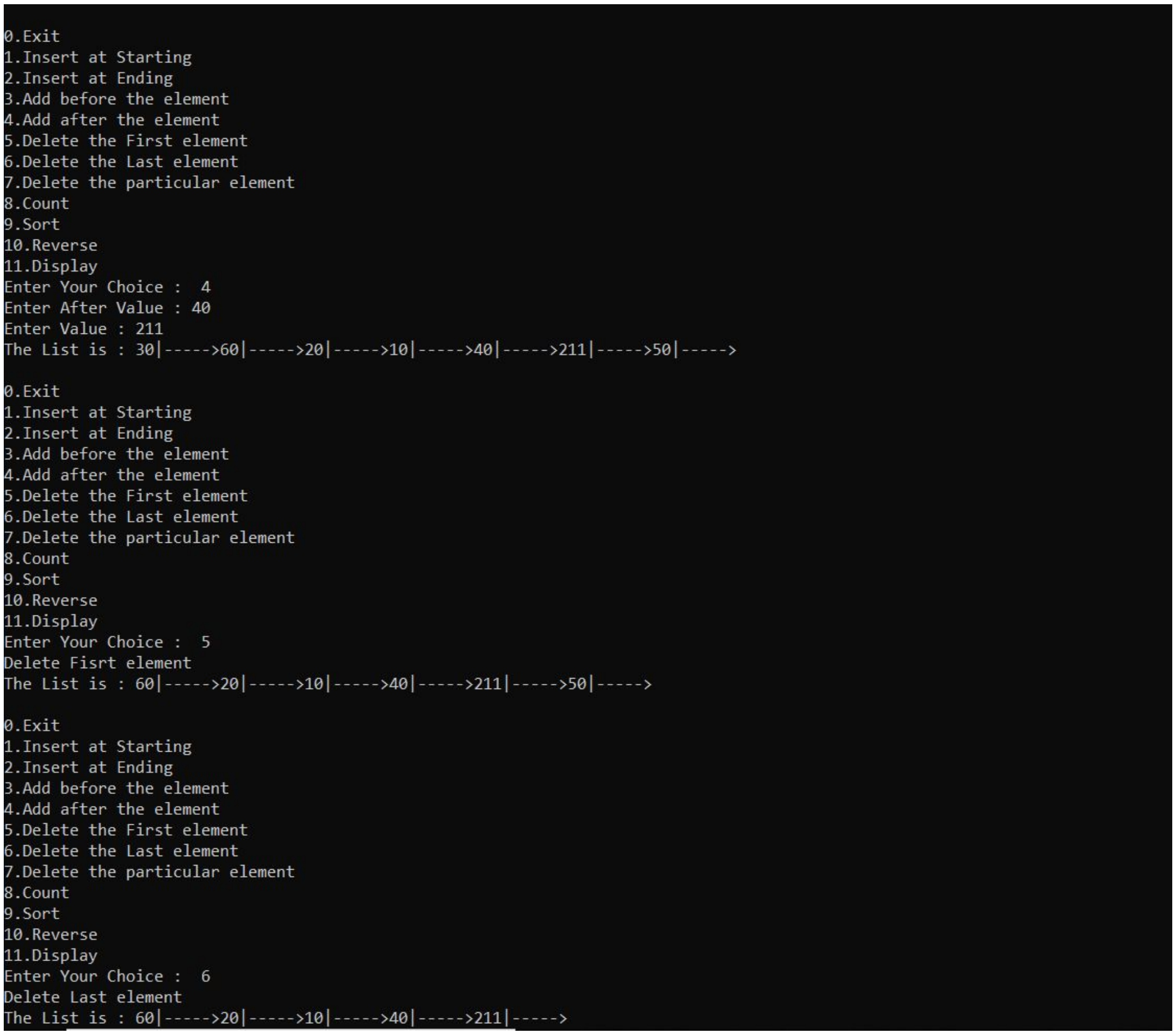
Abhishek Gupta **2019450017**



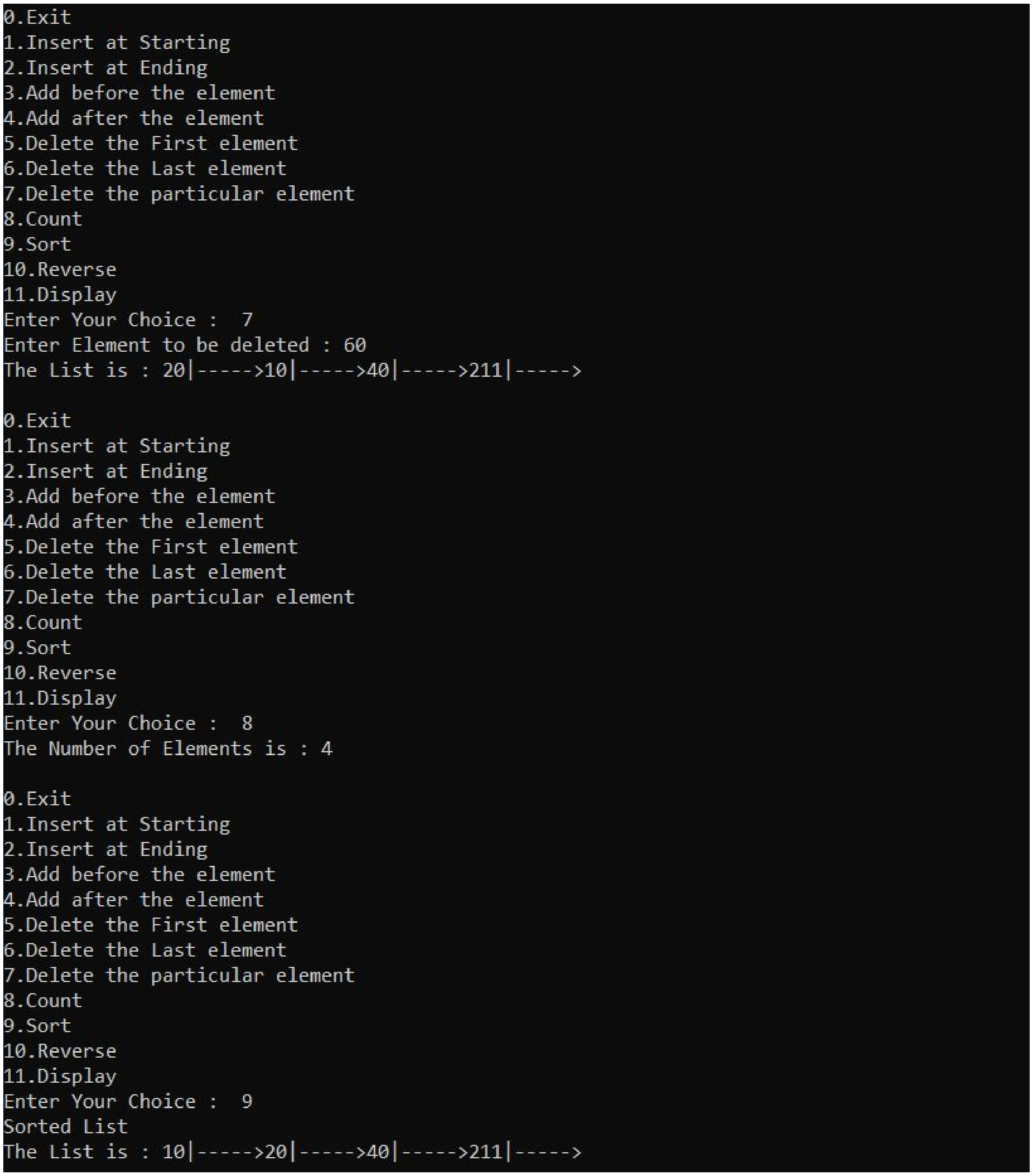
```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  2
Enter the value : 40
The List is : 30|----->20|----->10|----->40|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  2
Enter the value : 50
The List is : 30|----->20|----->10|----->40|----->50|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  3
Enter Before Value : 20
Enter Value : 60
The List is : 30|----->60|----->20|----->10|----->40|----->50|----->
```

Practical 05

Abhishek Gupta **2019450017**



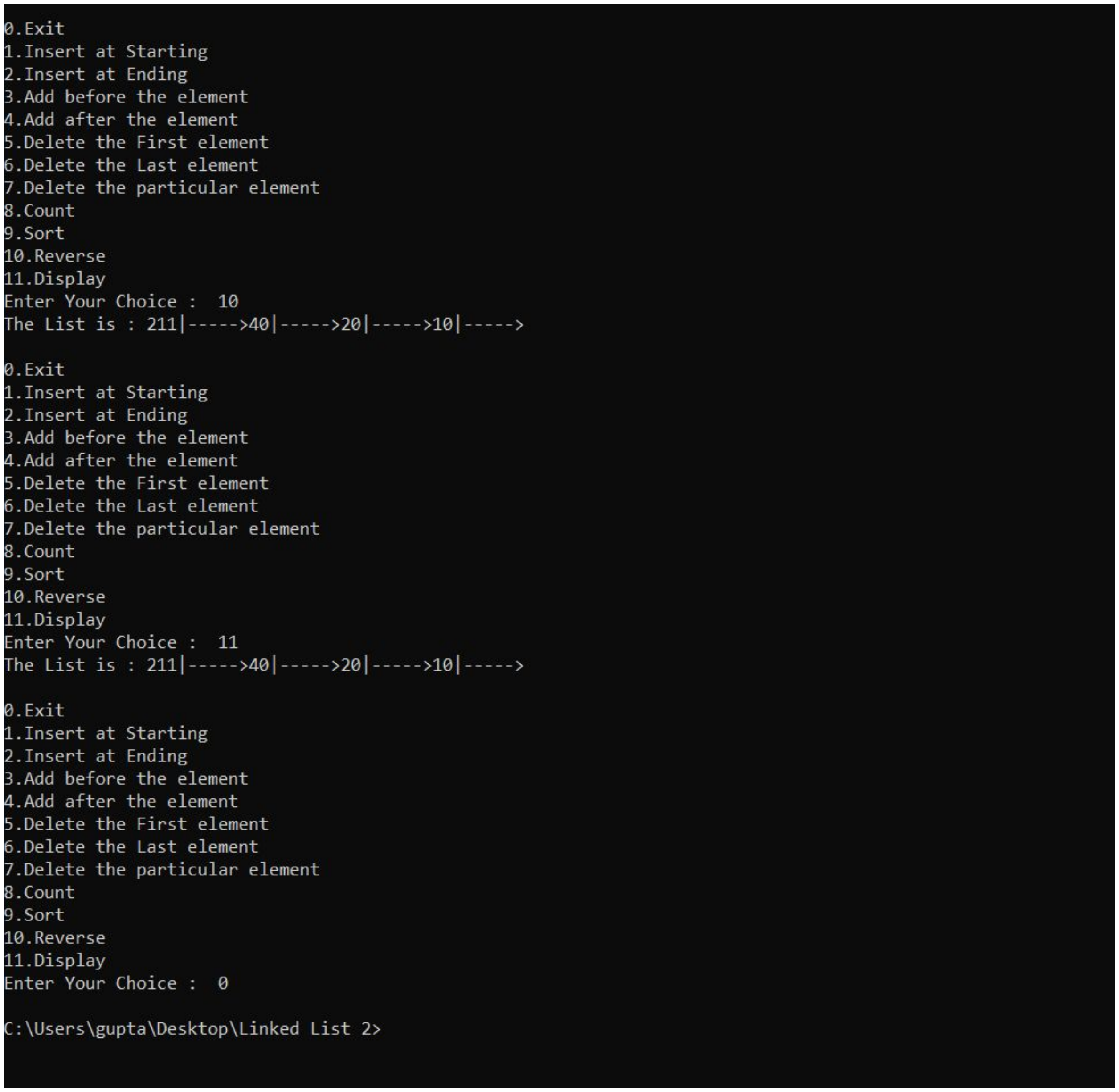
```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  4
Enter After Value : 40
Enter Value : 211
The List is : 30|----->60|----->20|----->10|----->40|----->211|----->50|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  5
Delete Fisrt element
The List is : 60|----->20|----->10|----->40|----->211|----->50|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  6
Delete Last element
The List is : 60|----->20|----->10|----->40|----->211|----->
```



```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  7
Enter Element to be deleted : 60
The List is : 20|----->10|----->40|----->211|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  8
The Number of Elements is : 4

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  9
Sorted List
The List is : 10|----->20|----->40|----->211|----->
```

Practical 05

Abhishek Gupta **2019450017**



```
0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  10
The List is : 211|----->40|----->20|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  11
The List is : 211|----->40|----->20|----->10|----->

0.Exit
1.Insert at Starting
2.Insert at Ending
3.Add before the element
4.Add after the element
5.Delete the First element
6.Delete the Last element
7.Delete the particular element
8.Count
9.Sort
10.Reverse
11.Display
Enter Your Choice :  0

C:\Users\gupta\Desktop\Linked List 2>
```

Practical 05

Abhishek Gupta **2019450017**

//Used Vishal Parab’s Help i was having troubles with the linking the program was going under unlimited loop.

5.3 Circular Linked List_17.cpp

```
#include<iostream>
#include<stdlib.h>
using namespace std;

class node
{
    public :
    int data;
    node *next;
};

class Linked_List
{
    public:
    node *list,*p,*q,*r,*temp;

    Linked_List()
    {
        list=NULL;
    }

    void Insert_start(int val)
    {
        p=(node*)malloc(sizeof(node));
        p->data=val;
        if(list==NULL)
        {
            p->next=p;
            list=p;
        }
        else
        {
            q=list;
            while(q->next!=list)
            {
                q=q->next;
            }
            q->next=p;
            p->next=list;
            list=p;
        }
    }

    void Insert_end(int val)
    {
        p=(node*)malloc(sizeof(node));
        p->data=val;
        if(list==NULL)
        {
            p->next=p;
            list=p;
        }
        else
        {
            q=list;
            while(q->next!=list)
            {
                q=q->next;
            }
            q->next=p;
            p->next=list;
        }
    }

    void after_add(int key,int val)
    {
        p=(node*)malloc(sizeof(node));
        p->data=val;
        if(list==NULL)
        {
            p->next=p;
            list=p;
        }
        else
        {
            bool exhaust=false;
            q=list;
            while(q->data!=key)
            {
                q=q->next;
                if(q==list)
                {
                    exhaust=true;
                    break;
                }
            }
            if(!exhaust)
            {
                r=q->next;
                q->next=p;
                p->next=r;
            }
            else
            {
                cout<<"\nThe element "<<key<<" doesnt exist in
the list!"<<endl;
            }
        }
    }

    void before_add(int key,int val)
    {
        bool exhaust=false;
        p=(node*)malloc(sizeof(node));
        p->data=val;
        if(list==NULL)
        {
            p->next=p;
            list=p;
        }
        else
        {
            q=list;
            if(q->data==key)
            {
                Insert_start(val);
            }
            else
            {
                while(q->data!=key)
                {
                    r=q;
                    q=q->next;
                    if(q==list)
                    {
                        exhaust=true;
                        break;
                    }
                }
                if(!exhaust)
                {
                    r->next=p;
                    p->next=q;
                }
                else
                {
                    cout<<"\nThe element "<<key<<" doesnt exist
in the list!"<<endl;
                }
            }
        }
    }



    void delete_start()
    {
        if(list==NULL)
        {
            cout<<"The list is empty!"<<endl;
        }
        else
        {
            q=list;
            if(q->next==list)
            {
                free(q);
                list=NULL;
                return;
            }
            list=list->next;
            q=list;
            while(q->next!=list)
            {
                r=q;
                q=q->next;
            }
            free(q);
            r->next=list;
        }
    }

    void delete_end()
    {
        if(list==NULL)
        {
            cout<<"The list is empty!"<<endl;
        }
        else
        {
            q=list;
            if(q->next==list)
```

```
            {
                free(q);
                list=NULL;
            }
            else
            {
                while(q->next!=list)
                {
                    r=q;
                    q=q->next;
                }
                r->next=list;
                free(q);
            }
        }
    }

    void reverse_ele()
    {
        q=p=list;
        temp=NULL;
        do
        {
            q=p->next;
            p->next=temp;
            temp=p;
            p=q;
        }while(q!=list);
        list=temp;
        q->next=list;
    }
```

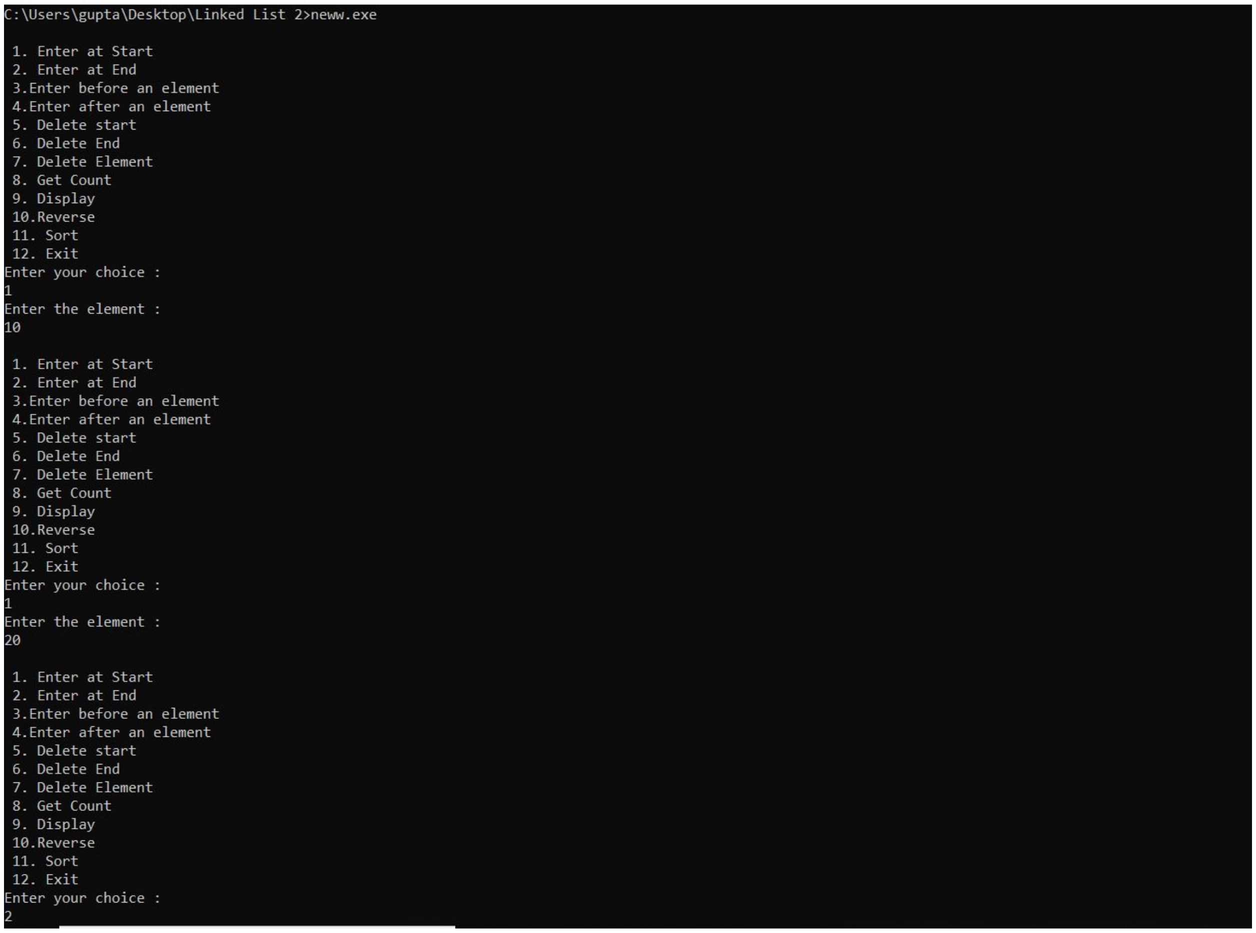
```
void sort_ele()
{
    for(int i=0;i<Count();i++)
    {
        q=list;
        while(q->next!=list)
        {
            r=q;
            q=q->next;
            if(r->data>q->data)
            {
                int temp=r->data;
                r->data=q->data;
                q->data=temp;
            }
        }
    }
}

void DeleteElement(int val)
{
    bool exhaust=false;
    if(list==NULL)
    {
        cout<<"The list is empty!"<<endl;
    }
    else
    {
        q=list;
```

```
            r=NULL;
            if(list->data==val)
            {
                delete_start();
                return;
            }
            do
            {
                r=q;
                q=q->next;
                if(q==list)
                {
                    exhaust=true;
                    break;
                }
            }while(q->data!=val);
            if(!exhaust)
            {
                temp=q->next;
                free(q);
                r->next=temp;
            }
            else
            {
                cout<<"\nThe element "<<val<<" doesnt exist in
the list!"<<endl;
            }
        }
    }

    int Count()
    {
        if(list==NULL)
        {
            return 0;
        }
        else
        {
            int c=0;
            q=list;
            do
            {
                c++;
                q=q->next;
            }while(q!=list);
            return c;
        }
    }

    void display()
    {
        q=list;
        if(list==NULL)
        {
            cout<<"\n List is Empty!"<<endl;
        }
        else
        {
            do
            {
                cout<<q->data<<"  --->  ";
                q=q->next;
            }while(q!=list);
        }
    }
};

int main()
{
    Linked_List l;
    int element,key;
    int choice;
    do
    {
        cout<<"\n 1. Enter at Start \n 2. Enter at End \n 
3.Enter before an element \n 4.Enter after an element \n 5. 
Delete start \n 6. Delete End \n 7. Delete Element \n 8. Get 
Count \n 9. Display \n 10.reverse_ele \n 11. sort_ele \n 12. 
Exit"<<endl;
        cout<<"Enter your choice : "<<endl;
        cin>>choice;
        switch (choice)
        {
            case 1:
            {
                cout<<"Enter the element : "<<endl;
                cin>>element;
                l.Insert_start(element);
                break;
            }

            case 2:
            {
                cout<<"Enter the element : "<<endl;
                cin>>element;
                l.Insert_end(element);
                break;
            }

            case 3:
            {
                cout<<"Enter the element to add: "<<endl;
                cin>>element;
                cout<<"Element should be added before :
"<<endl;

                cin>>key;
                l.before_add(key,element);
                break;
            }

            case 4:
            {
                cout<<"Enter the element to add: "<<endl;
                cin>>element;
                cout<<"Element should be added after :  "<<endl;
                cin>>key;
                l.after_add(key,element);
                break;
            }

            case 5:
            {
                l.delete_start();
                break;
            }

            case 6:
            {
                l.delete_end();
                break;
            }

            case 7:
            {
                cout<<"Enter the element to delete: "<<endl;
                cin>>element;
                l.DeleteElement(element);
                break;
            }

            case 8:
            {
                cout<<"\n The list contains "<<l.Count()<<"
elements"<<endl;
                break;
            }

            case 9:
            {
                l.display();
                break;
            }

            case 10:
            {
                l.reverse_ele();
                break;
            }

            case 11:
            {
                l.sort_ele();
                break;
            }

            default:
                break;
        }
    }
    while(choice!=12);
}
```

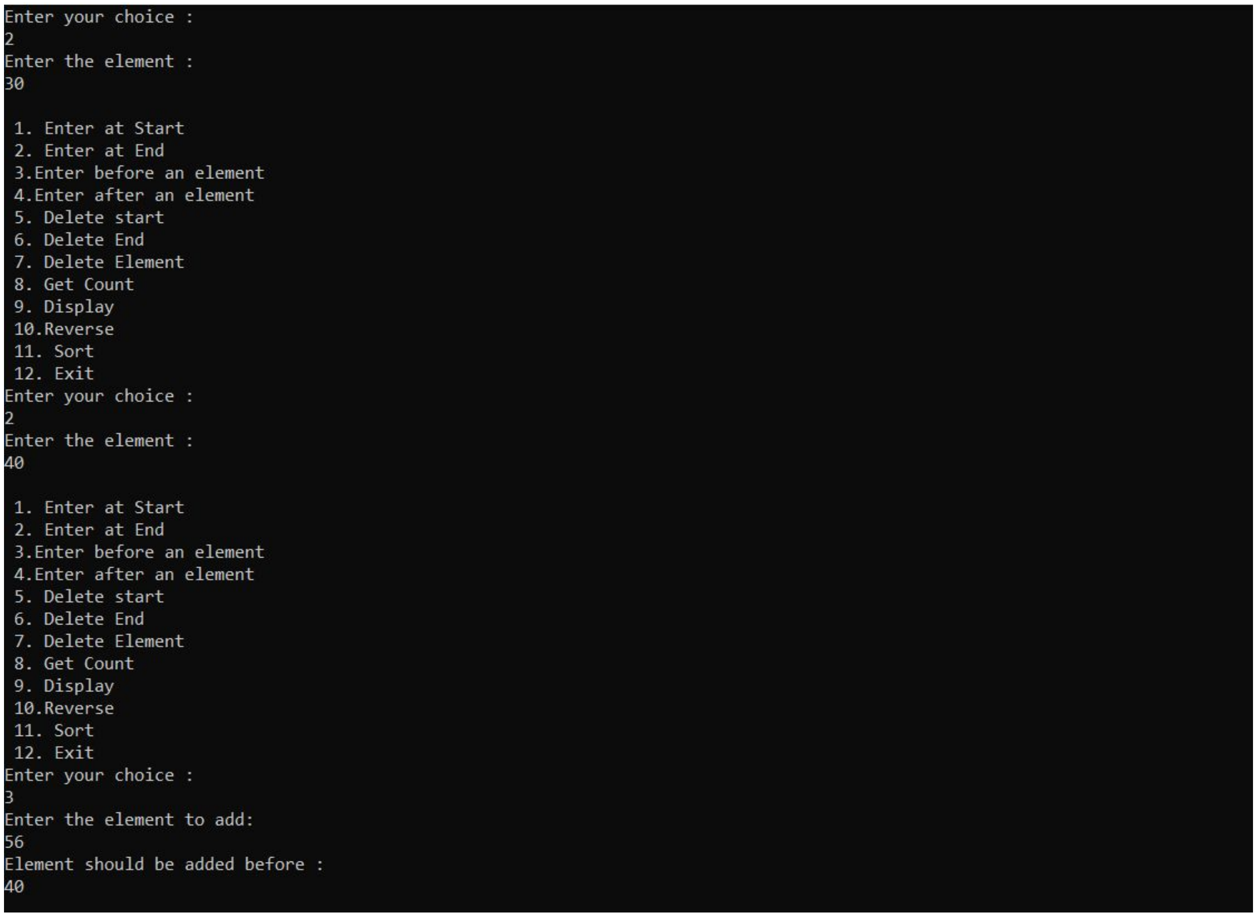
Output :

Practical 05

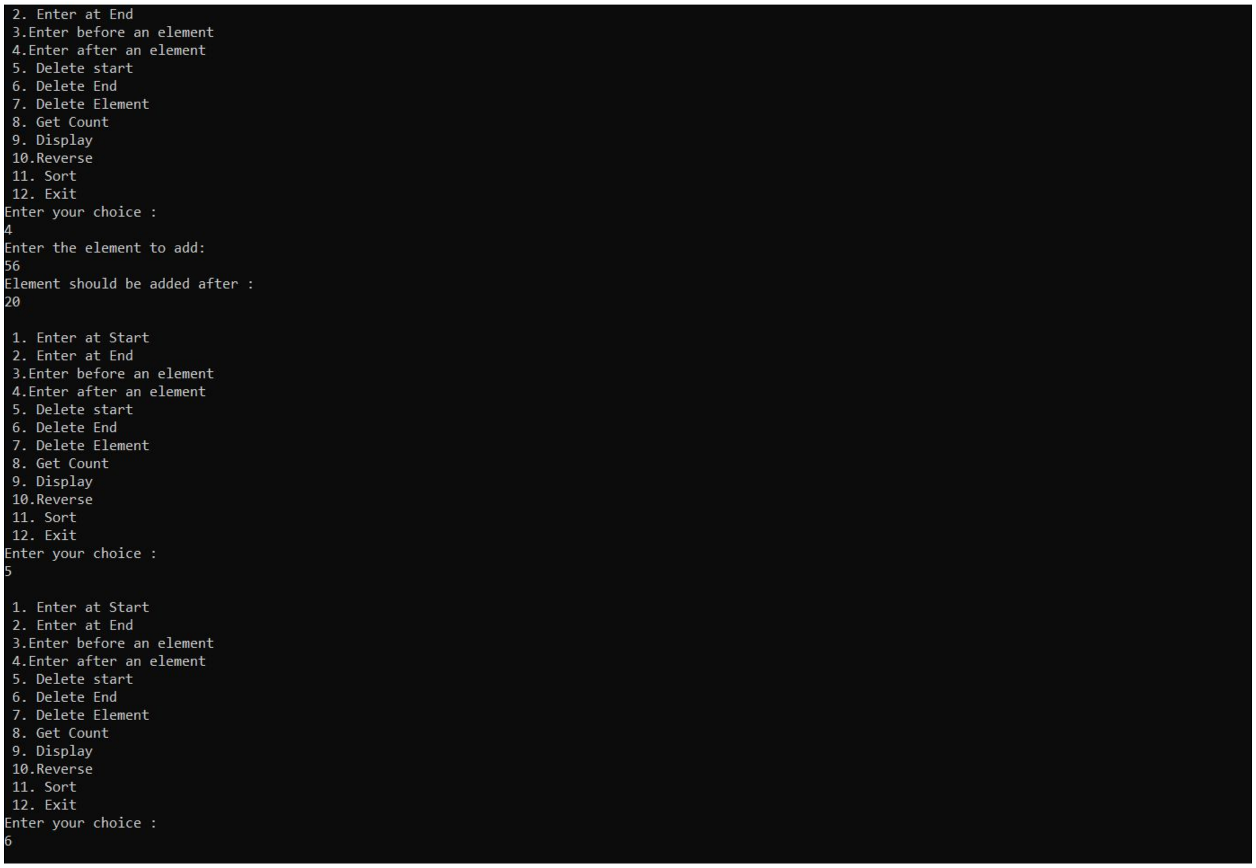
Abhishek Gupta **2019450017**



```
C:\Users\gupta\Desktop\Linked List 2>neww.exe

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
1
Enter the element :
10

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
1
Enter the element :
20

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
2
```



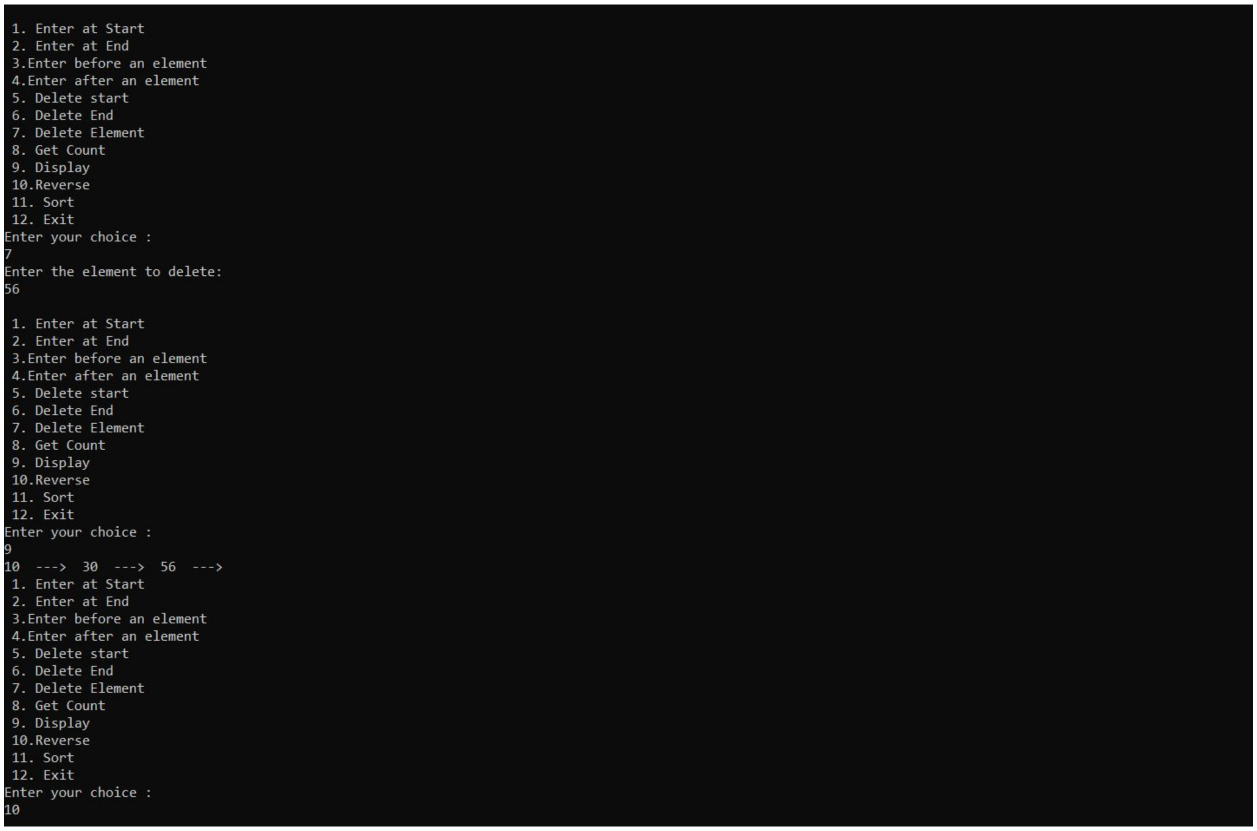
```
Enter your choice :
2
Enter the element :
30

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
2
Enter the element :
40

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
3
Enter the element to add:
56
Element should be added before :
40
```



```
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
4
Enter the element to add:
56
Element should be added after :
20

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
5

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
6
```

Practical 05

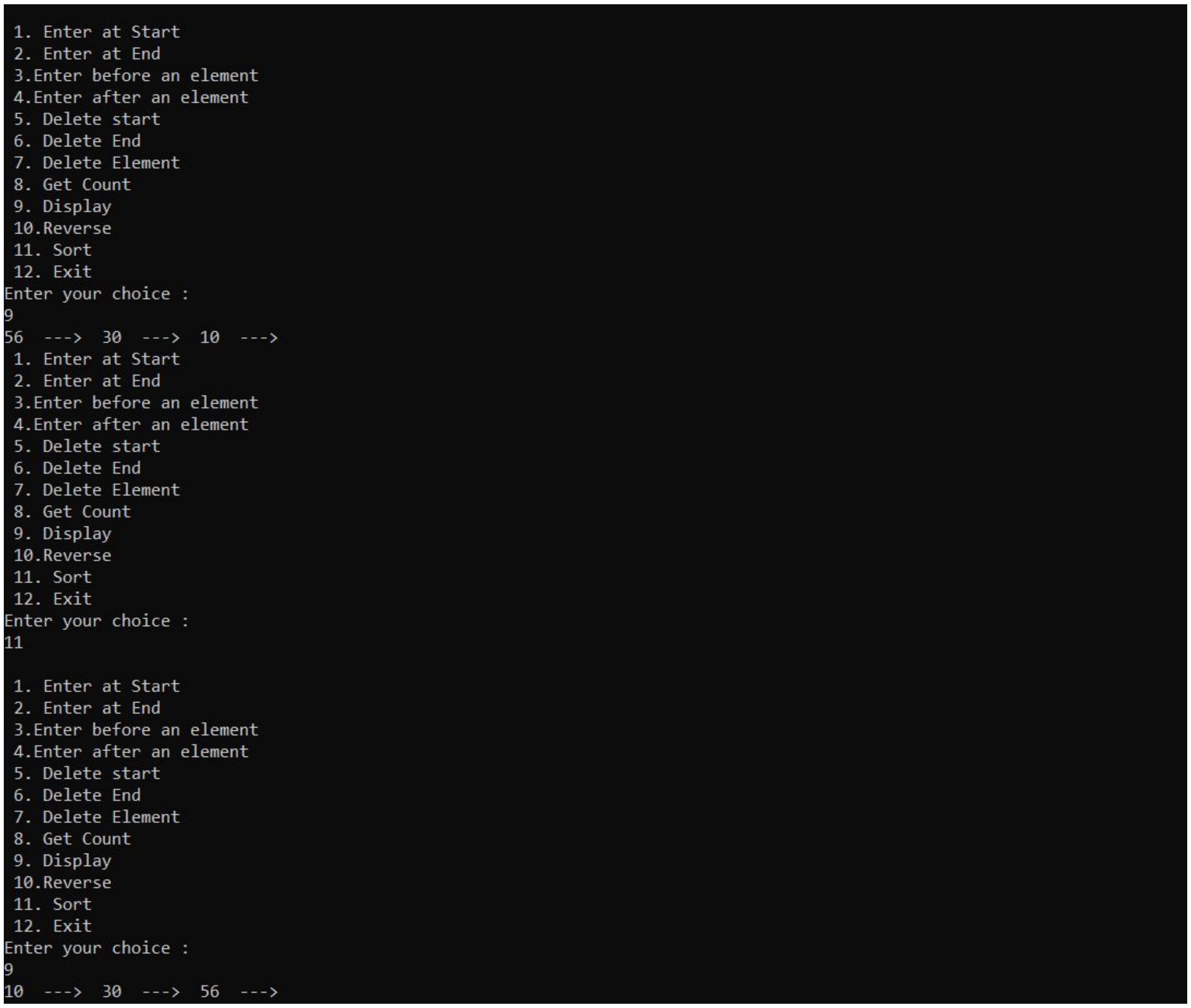
Abhishek Gupta 2019450017



```
 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
7
Enter the element to delete:
56

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
9
10  --->  30  --->  56  --->
 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
10
```

Practical 05

Abhishek Gupta **2019450017**



```
 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
9
56  --->  30  --->  10  --->
 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
11

 1. Enter at Start
 2. Enter at End
 3.Enter before an element
 4.Enter after an element
 5. Delete start
 6. Delete End
 7. Delete Element
 8. Get Count
 9. Display
 10.Reverse
 11. Sort
 12. Exit
Enter your choice :
9
10  --->  30  --->  56  --->
```