1.1Singly_Link_List_13

Code:-

```
#include<iostream>

#include<malloc.h>

using namespace std;

struct node{

   int data;

   struct node *next;

}*list=NULL, *p, *q, *r, *s;


class nick{

   int action, value, element;

   public:

   mytech(){

      do{

         cout << endl << "1. Insert at Begining\n2. Insert at the End\n3. Insert Before an element.\n4. Insert after an element\n5. Delete at beginning\n6. Delete at end\n7. Delete Particular value\n8. Display\n9.Count\n10. Reverse\n11. Sort\n12. Exit\nEnter action you want to perform: ";

         cin >> action;

         switch (action)

         {

            case 1:

               insert_b();

               break;

            case 2:

               insert_e();

               break;

            case 3:

               insert_be();

               break;

            case 4:
               insert_ae();
               break;
            case 5:
               delete_b();
               break;
            case 6:
               delete_e();
               break;
            case 7:
               delete_v();
               break;
            case 8:
               display();
               break;
            case 9:
               count();
               break;
            case 10:
               reverse();
               break;
            case 11:
               sort();
               break;
            case 12:
               break;
         default:
            break;
         }
      }while(action!=12);
   }
```

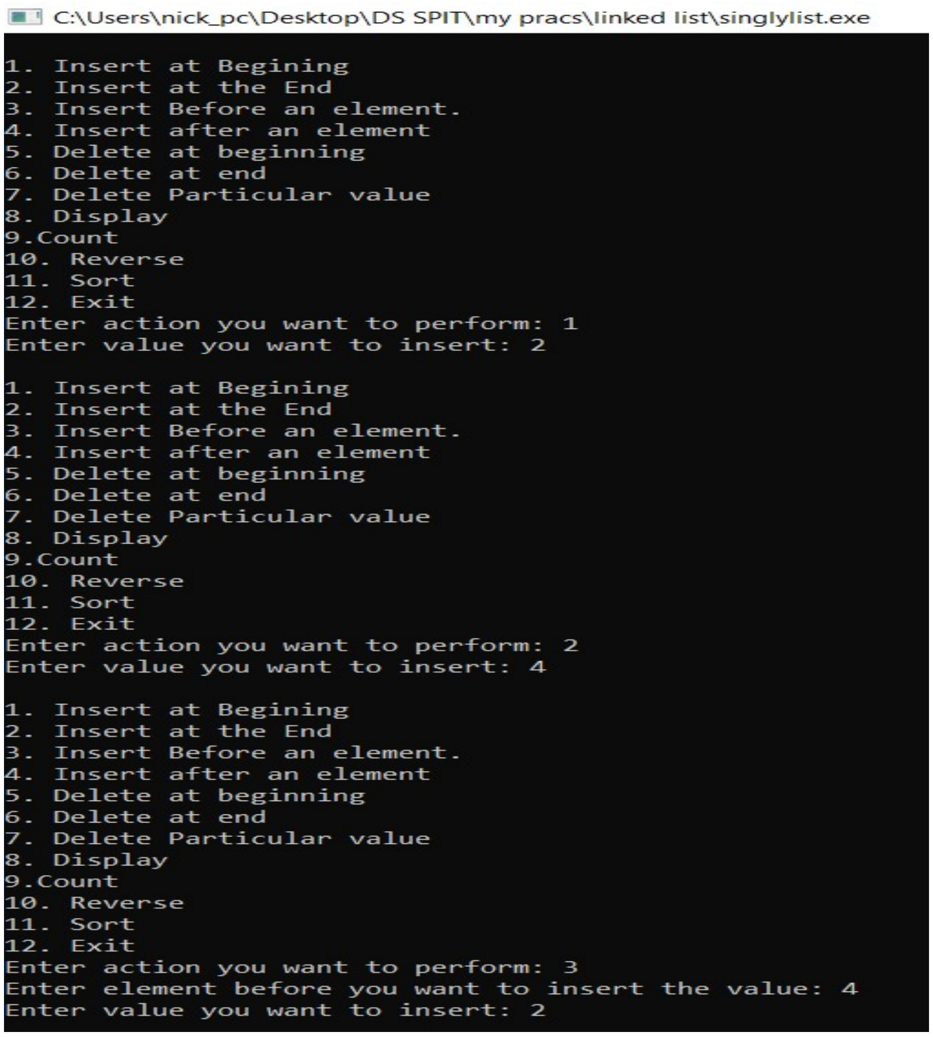
```
void insert_b(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    if(list==NULL)
        p->next = NULL;
    else
        p->next = list;
    list = p;
}

void insert_e(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    q = (struct node*)malloc(sizeof(node));
    p->data = value;
    if(list==NULL){
        p->next = NULL;
        list = p;
    }
    else{
        q = list;
        while(q->next!=NULL)
            q = q->next;
        q->next = p;
        p->next = NULL;
    }
}
void insert_be(){
    cout << "Enter element before you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;

    while((q->data != element) && (q->next != NULL)){
        r = q;
        q = q->next;
    }
    if(q->data==element)
     {
        if(list->data==element){
        p->next = q;
        list = p;
        }
        else{
        r->next = p;
        p->next = q;
        }
    }
    else{
        cout << "Data not found.";
    }

}
void insert_ae(){
    cout << "Enter element after you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;
    r = q->next;
    while((q->data != element) && (q->next != NULL)){
        q = r;
        r = r->next;
    }
    if(q->data != element)
        cout << "Data not found.";
    else if(r == NULL){
        q->next = p;
        p->next = NULL;
    }
    else{
        q->next = p;
        p->next = r;
    }
}

void delete_b(){
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        list = list->next;
        cout << "Element has been deleted." << endl;
    }
}
```

```
void delete_e(){
    p = list;
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        while(p->next != NULL){
            q = p;
            p = p->next;
        }
        q->next = NULL;
        cout << "Element has been deleted." << endl;
    }
}

void delete_v(){
    p = list;
    q = p;
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        cout << "Enter value you want to delete: ";
        cin >> value;
        while(p->data != value && p->next!=NULL){
            q = p;
            p = p->next;
        }
        if(p->data != value)
            cout << "Element not found.";
        else{
            if(p==q)
                list = p->next;
            else
                q->next = p->next;

            cout << "Element has been deleted." << endl;
        }
    }
}

void display(){
    if(list==NULL)
        cout << "No Element in the linked list.";
    else{
        p = list;
        cout << "Elements in the linked list are: ";
        while(p!=NULL){
            cout << p->data << " ";
            p = p->next;
        }
    }
}

void count(){
    int count = 0;
    p = list;
    while(p != NULL){
        count++;
        p = p->next;
    }
    cout << "Count = " << count << endl;
}
```

```
 void reverse(){
   if(list==NULL)
     cout << "No Element in the linked list.";
   else{
     struct node *temp;
     q = s = list;
     temp = NULL;
     r = q->next;
     while(r->next!=NULL){
       temp = q;
       q = r;
       r = q->next;
       q->next = temp;
     }
     list = q;
     s->next = NULL;
     cout << "Linked List has been reversed.";
   }
 }

 void sort(){
   p = list;
   int temp;
   while(p->next!=NULL){
     q = p->next;
     while(q->next!=NULL){
       if(p->data > q->data){
         temp = p->data;
         p->data = q->data;
         q->data = temp;
       }
```

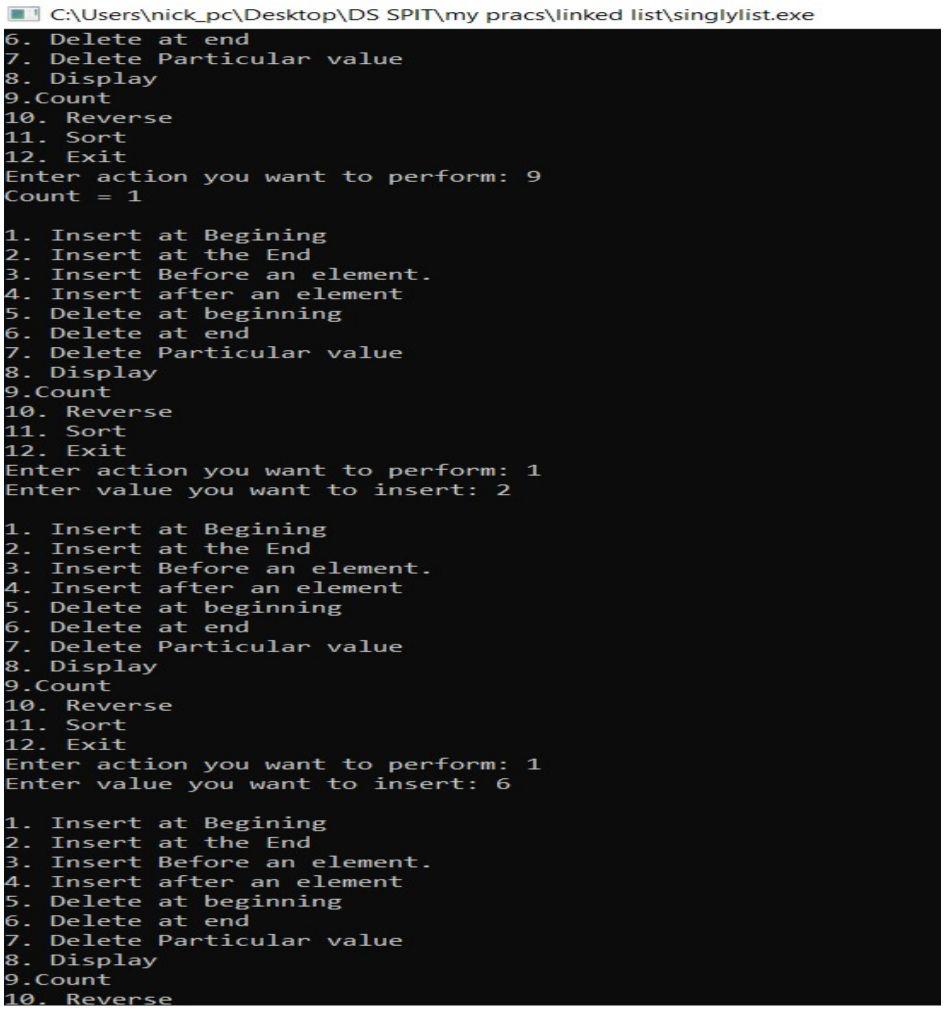
```
            q = q->next;
          }
          p = p->next;
        }
    }
};


int main(){
    nick o;
    o.mytech();
}
```

Output:-



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\singlylist.exe

```
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 1
Enter value you want to insert: 2

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 2
Enter value you want to insert: 4

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 3
Enter element before you want to insert the value: 4
Enter value you want to insert: 2
```


C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\singlylist.exe

```
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 9
Count = 1

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 1
Enter value you want to insert: 2

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 1
Enter value you want to insert: 6

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
```

1.2Doubly_Link_List_13

Code:-

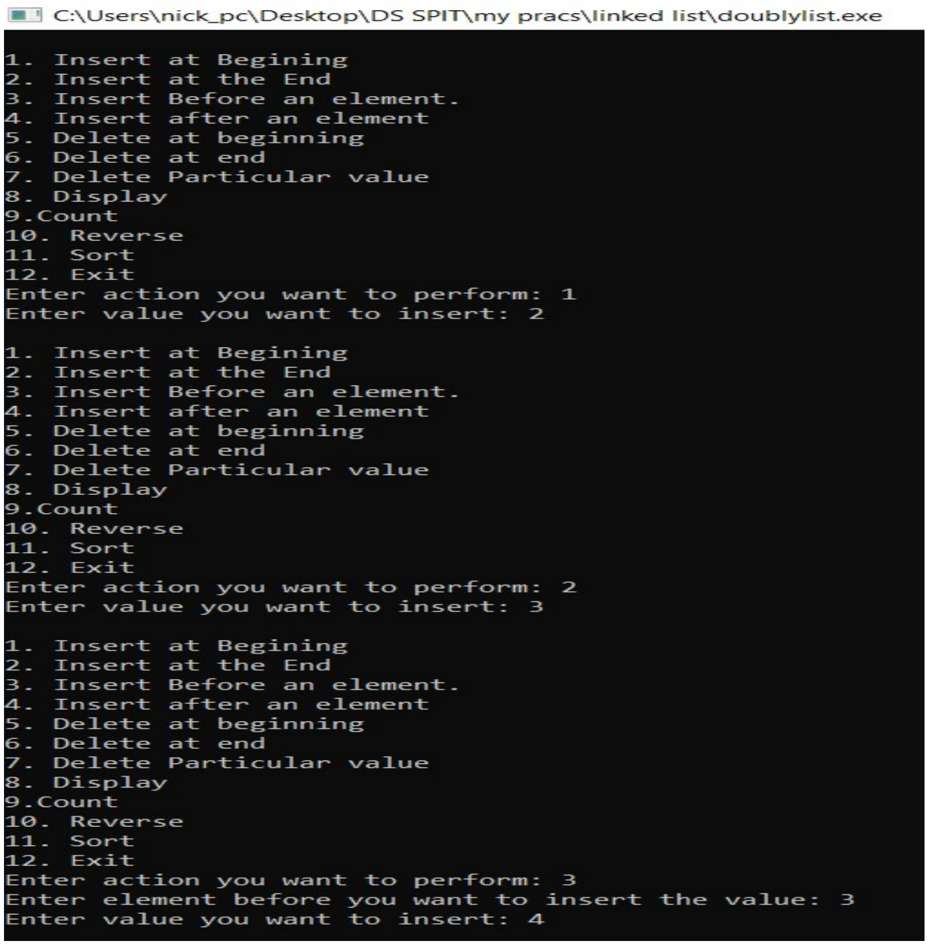
```
#include<iostream>

#include<malloc.h>

using namespace std;

struct node{

   int data;

   struct node *lptr;

   struct node *rptr;

}*list=NULL, *p, *q, *r, *s;


class nick{
    int action, value, element;

    public:

    mytech(){

        do{

            cout << endl << "1. Insert at Begining\n2. Insert at the End\n3. Insert Before an element.\n4.
Insert after an element\n5. Delete at beginning\n6. Delete at end\n7. Delete Particular value\n8.
Display\n9.Count\n10. Reverse\n11. Sort\n12. Exit\nEnter action you want to perform: ";

            cin >> action;

            switch (action)

            {

                case 1:

                    insert_b();

                    break;

                case 2:

                    insert_e();

                    break;

                case 3:

                    insert_be();

                    break;

                case 4:

                    insert_ae();

                    break;

                case 5:

                    delete_b();

                    break;

                case 6:

                    delete_e();

                    break;

                case 7:

                    delete_v();

                    break;

                case 8:
```

```
                display();
                break;
            case 9:
                count();
                break;
            case 10:
                reverse();
                break;
            case 11:
                sort();
                break;
            case 12:
                break;
        default:
            break;
        }
    }while(action!=12);
}

void insert_b(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    if(list==NULL)
        p->lptr = p->rptr = NULL;
    else{
        p->lptr = NULL;
        p->rptr = list;
        list->lptr = p;
    }
    list = p;
  }

  void insert_e(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    q = (struct node*)malloc(sizeof(node));
    p->data = value;
    if(list==NULL){
      p->lptr = p->rptr = NULL;
      list = p;
    }
    else{
      q = list;
      while(q->rptr!=NULL)
        q = q->rptr;
      q->rptr = p;
      p->rptr = NULL;
      p->lptr = q;
    }
  }
  void insert_be(){
    cout << "Enter element before you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;
    while((q->data != element) && (q->rptr != NULL)){
      r = q;
      q = q->rptr;
    }
    if(q->data==element){
      if(list->data==element){
        p->rptr = q;
        p->lptr = NULL;
        list = p;
      }
      else{
        r->rptr = p;
        p->lptr = r;
        p->rptr = q;
        q->lptr = p;
      }
    }
    else{
      cout << "Data not found.";
    }

  }
  void insert_ae(){
    cout << "Enter element after you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;
    r = q->rptr;
```

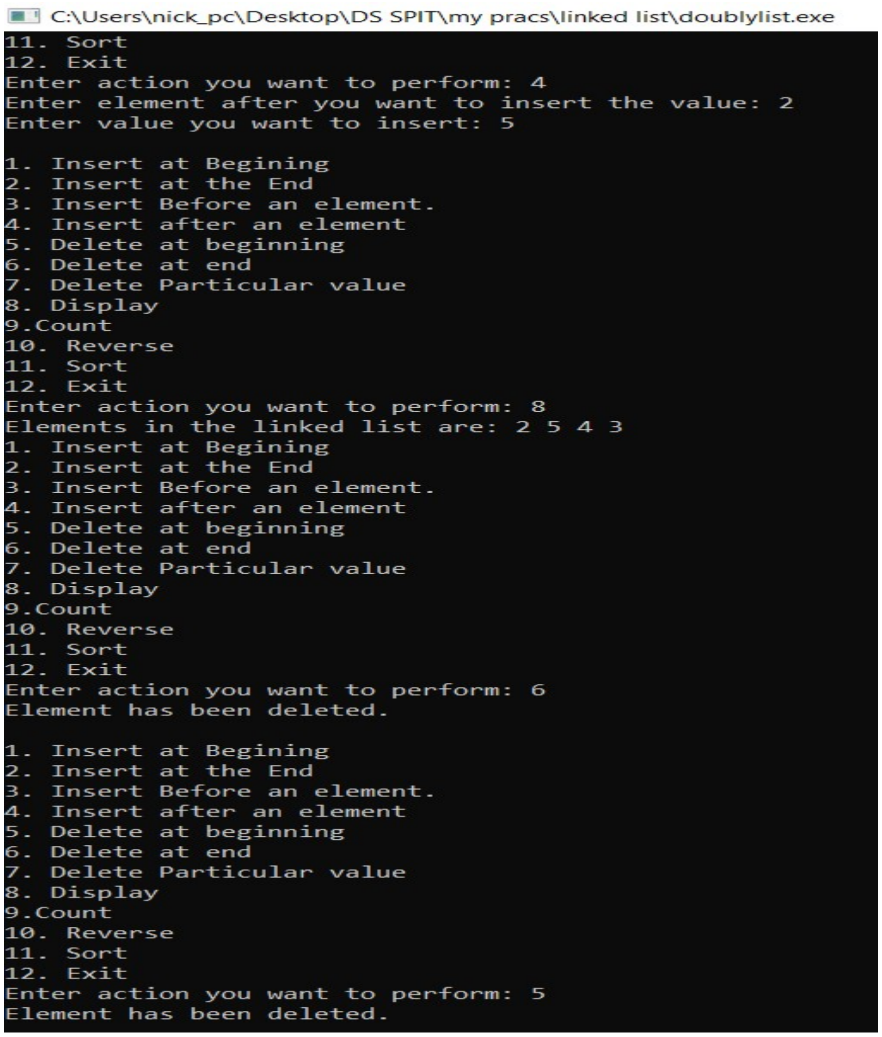
```
        while((q->data != element) && (q->rptr != NULL)){
            q = r;
            r = r->rptr;
        }
        if(q->data != element)
            cout << "Data not found.";
        else if(r == NULL){
            q->rptr = p;
            p->lptr = q;
            p->rptr = NULL;
        }
        else{
            q->rptr = p;
            p->lptr = q;
            p->rptr = r;
            r->lptr = p;
        }
    }

    void delete_b(){
        if(list==NULL)
            cout << "List is empty nothing to delete.";
        else{
            list = list->rptr;
            cout << "Element has been deleted." << endl;
        }
    }

    void delete_e(){
        p = list;
        if(list==NULL)
```

```
            cout << "List is empty nothing to delete.";
        else{
            while(p->rptr != NULL){
                q = p;
                p = p->rptr;
            }
            q->rptr = NULL;
            cout << "Element has been deleted." << endl;
        }
    }

    void delete_v(){
        p = list;
        q = p;
        if(list==NULL)
            cout << "List is empty nothing to delete.";
        else{
            cout << "Enter value you want to delete: ";
            cin >> value;
            while(p->data != value && p->rptr!=NULL){
                q = p;
                p = p->rptr;
            }
            if(p->data != value)
                cout << "Data not found.";
            else{
                if(p==q)
                    list = p->rptr;
                else
                    q->rptr = p->rptr;
                cout << "Element has been deleted." << endl;
            }
        }
    }

    void display(){
        if(list==NULL)
            cout << "No Element in the linked list.";
        else{
            p = list;
            cout << "Elements in the linked list are: ";
            while(p!=NULL){
                cout << p->data << " ";
                p = p->rptr;
            }
        }
    }

    void count(){
        int count = 0;
        p = list;
        while(p != NULL){
            count++;
            p = p->rptr;
        }
        cout << "Count = " << count << endl;
    }

    void reverse(){
        if(list==NULL)
            cout << "No Element in the linked list.";
        else{
            struct node *temp;
            q = s = list;
            temp = NULL;
            r = q->rptr;
            s->lptr = r;
            while(r!=NULL){
                temp = q;
                q = r;
                r = q->rptr;
                q->rptr = temp;
            }
            list = q;
            s->rptr = NULL;
            cout << "Linked List has been reversed.";
        }
    }

    void sort(){
        p = list;
        int temp;
        while(p->rptr!=NULL){
            q = p->rptr;
            while(q->rptr!=NULL){
                if(p->data > q->data){
                    temp = p->data;
                    p->data = q->data;
                    q->data = temp;
                }
                q = q->rptr;
            }
            p = p->rptr;
        }
    }
};


int main(){
    nick o;
            o.mytech();
}
```

Output:-

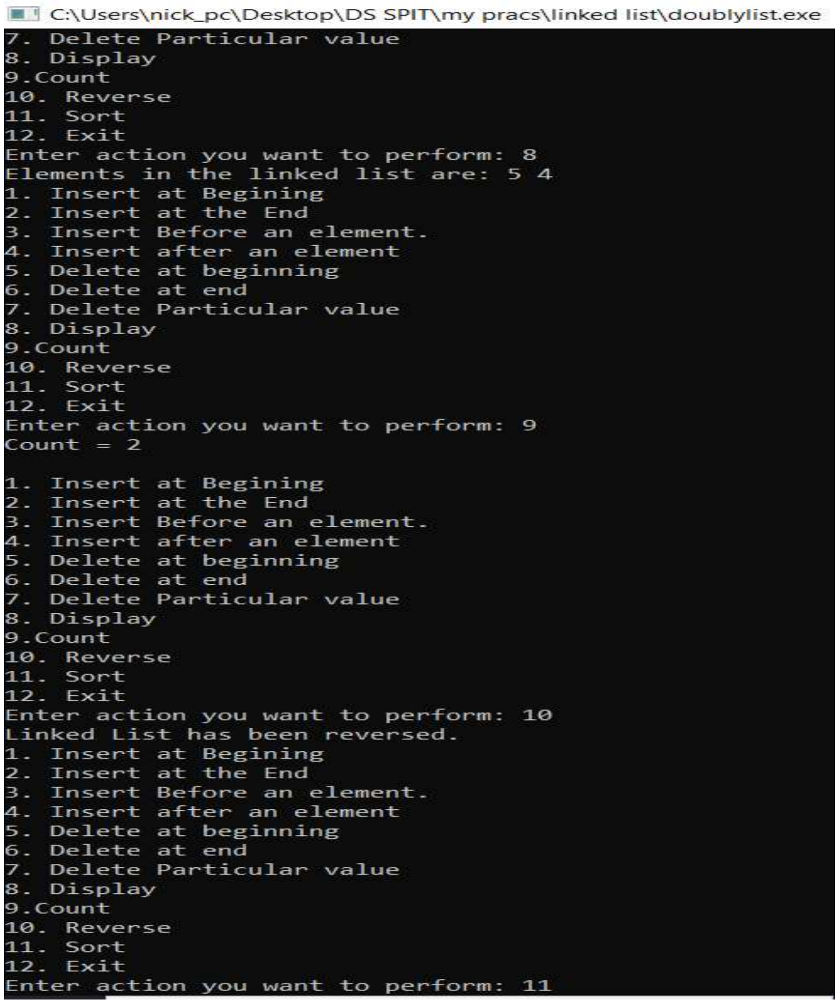


```
C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\doublylist.exe
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 1
Enter value you want to insert: 2

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 2
Enter value you want to insert: 3

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 3
Enter element before you want to insert the value: 3
Enter value you want to insert: 4
```



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\doublylist.exe

```
11. Sort
12. Exit
Enter action you want to perform: 4
Enter element after you want to insert the value: 2
Enter value you want to insert: 5

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 8
Elements in the linked list are: 2 5 4 3
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 6
Element has been deleted.

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 5
Element has been deleted.
```



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\doublylist.exe

```
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 8
Elements in the linked list are: 5 4
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 9
Count = 2

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 10
Linked List has been reversed.
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 11
```

1.3Circular_Link_List_13

Code:-

```
#include<iostream>

#include<malloc.h>

using namespace std;

struct node{

   int data;

   struct node *next;

}*list=NULL, *p, *q, *r, *s;


class nick{

   int action, value, element;
```

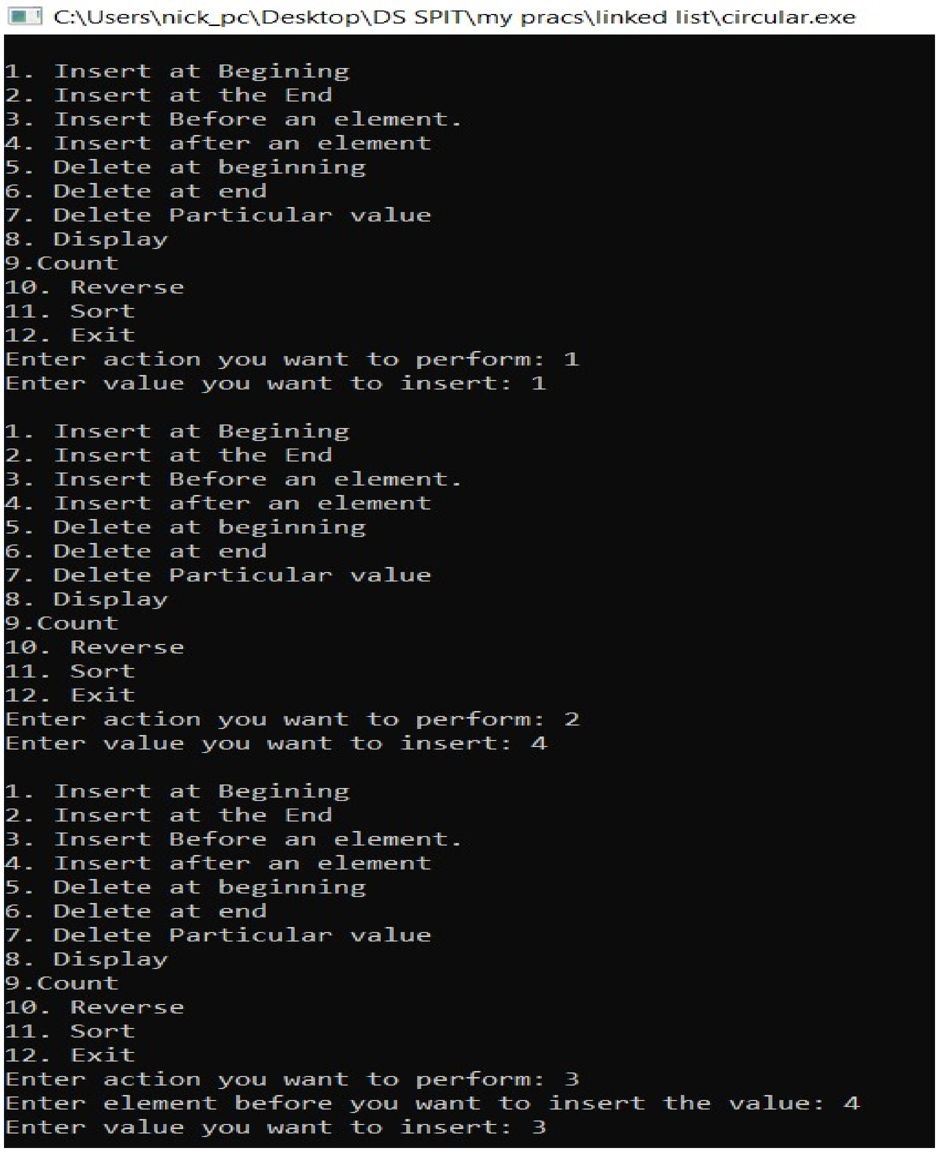
```
public:
void mytech(){
    do{
        cout << endl << "1. Insert at Begining\n2. Insert at the End\n3. Insert Before an element.\n4. Insert after an element\n5. Delete at beginning\n6. Delete at end\n7. Delete Particular value\n8. Display\n9.Count\n10. Reverse\n11. Sort\n12. Exit\nEnter action you want to perform: ";
        cin >> action;
        switch (action)
        {
            case 1:
                insert_b();
                break;
            case 2:
                insert_e();
                break;
            case 3:
                insert_be();
                break;
            case 4:
                insert_ae();
                break;
            case 5:
                delete_b();
                break;
            case 6:
                delete_e();
                break;
            case 7:
                delete_v();
                break;
            case 8:
                display();
                break;
            case 9:
                count();
                break;
            case 10:
                reverse();
                break;
            case 11:
                sort();
                break;
            case 12:
                break;
        default:
            cout << "Invalid entry.\nTry again.\n";
            break;
        }
    }while(action!=12);
}

void insert_b(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));

    p->data = value;
    if(list==NULL)
        p->next = p;
    else{
        q = list;
        while(q->next!=list)
            q = q->next;
        q->next = p;
        p->next = list;
    }
    list = p;
}

void insert_e(){
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    q = (struct node*)malloc(sizeof(node));
    p->data = value;
    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 insert_be(){
    cout << "Enter element before you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;


    while((q->data != element) && (q->next != NULL)){
        r = q;
        q = q->next;
    }
    if(q->data==element)
     {
        if(list->data==element){
            s = list;
            while(s->next!=list)
                s = s->next;
            s->next = p;
            p->next = list;
            list = p;
        }
        else{
            r->next = p;
            p->next = q;
        }
    }
    else{
        cout << "Data not found.";
    }


}
void insert_ae(){
    cout << "Enter element after you want to insert the value: ";
    cin >> element;
    cout << "Enter value you want to insert: ";
    cin >> value;
    p = (struct node*)malloc(sizeof(node));
    p->data = value;
    q = list;
    r = q->next;
    while((q->data != element) && (q->next != NULL)){
        q = r;
        r = r->next;
    }
    if(q->data != element)
        cout << "Data not found.";
    else if(r == list){
        q->next = p;
        p->next = list;
    }
    else{
        q->next = p;
        p->next = r;
    }
}

void delete_b(){
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        p = list;
        while(p->next!=list){
            q = p;
            p = p->next;
        }
        if(list == p)
            list = NULL;
        else{
            q->next = list;
            list = list->next;
        }
        cout << "Element has been deleted." << endl;
    }
}

void delete_e(){
    if(list==NULL)
        cout << "List is empty nothing to delete." << endl;
    else{
        p = list;
        while(p->next != list){
            q = p;
            p = p->next;
        }
        if(list == p)
            list = NULL;
        else
            q->next = list;
        cout << "Element has been deleted." << endl;
    }
}

void delete_v(){
    if(list==NULL)
        cout << "List is empty nothing to delete." << endl;
    else{
        p = list;
        q = p;
```

```
            cout << "Enter value you want to delete: ";
            cin >> value;
            while(p->data != value && p->next!=list){
                q = p;
                p = p->next;
            }
            if(p->data != value)
                cout << "Element not found.";
            else{
                if(p==q){
                    if(p->next==p)
                        list = NULL;
                    else{
                        r = list;
                        while(r->next!=list){
                            r = r->next;
                        }
                        r = p->next;
                        list = p;
                    }
                }
                else if(p->next = list){
                    q->next = list;
                }
                else
                    q->next = p->next;
                cout << "Element has been deleted." << endl;
            }
        }
    }
```

```
void display(){
    if(list==NULL)
        cout << "No Element in the linked list." << endl;
    else{
        p = list;
        cout << "Elements in the linked list are: ";
        do{
            cout << p->data << " ";
            p = p->next;
        }while(p!=list);
        cout << endl;
    }
}

void count(){
    int count = 0;
    if(list!=NULL){
        p = list;
        do{
            count++;
            p = p->next;
        }while(p != list);
    }
    cout << "Count = " << count << endl;
}

void reverse(){
    if(list==NULL)
        cout << "No Element in the linked list.";
    else{
        struct node *temp;
```

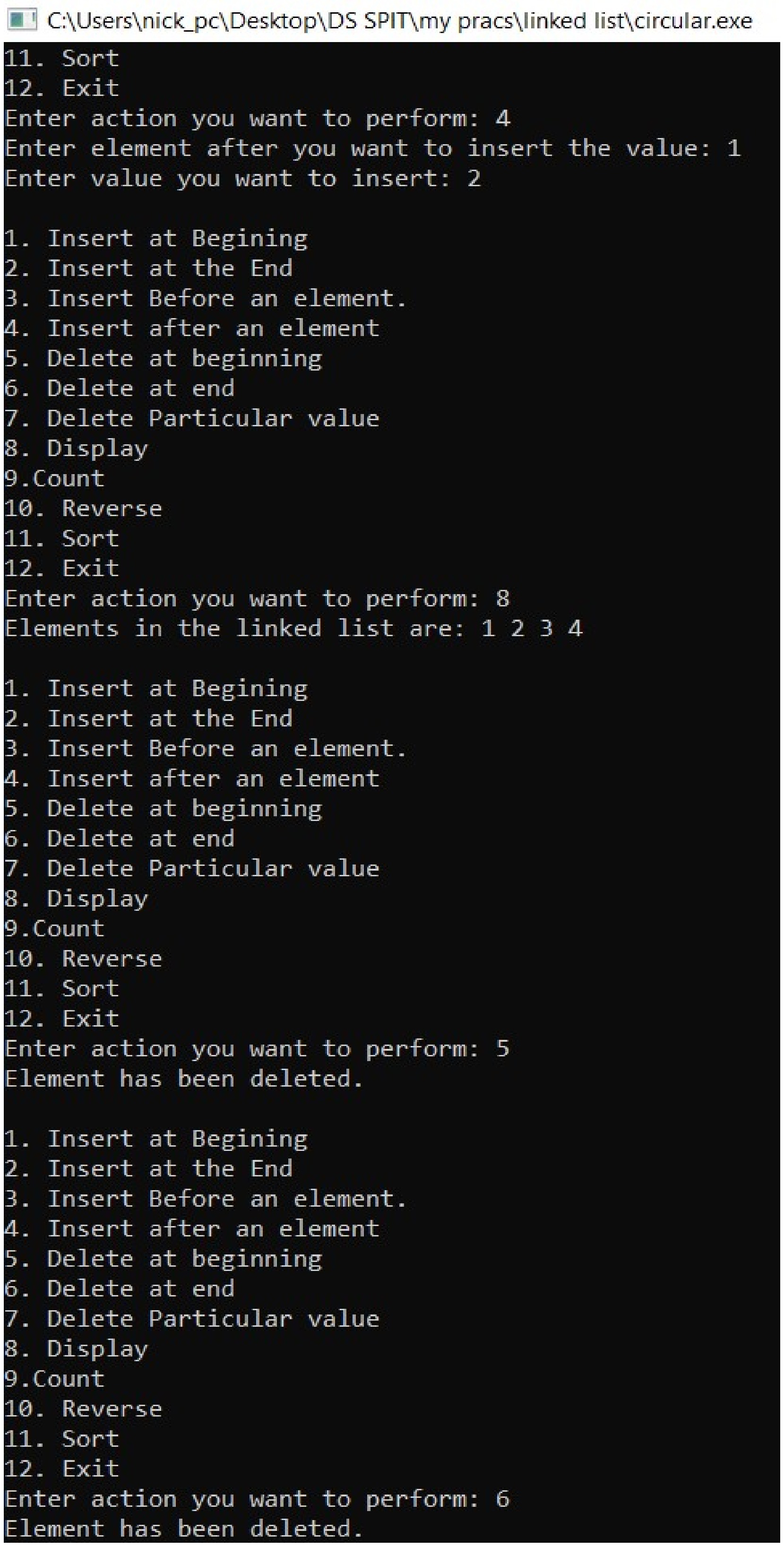
```
            q = s = list;
            temp = NULL;
            r = q->next;
            while(r!=list){
                temp = q;
                q = r;
                r = q->next;
                q->next = temp;
            }
            list = q;
            s->next = q;
            cout << "Linked List has been reversed.";
        }
    }

    void sort(){
        p = list;
        int temp;
        while(p->next!=list){
            q = p->next;
            while(q!=list){
                if(p->data > q->data){
                    temp = p->data;
                    p->data = q->data;
                    q->data = temp;
                }
                q = q->next;
            }
            p = p->next;
        }
        cout << "Elements sorted...";
        display();
    }
};

int main(){
    nick o;
    o.mytech();
}
```

Output:-



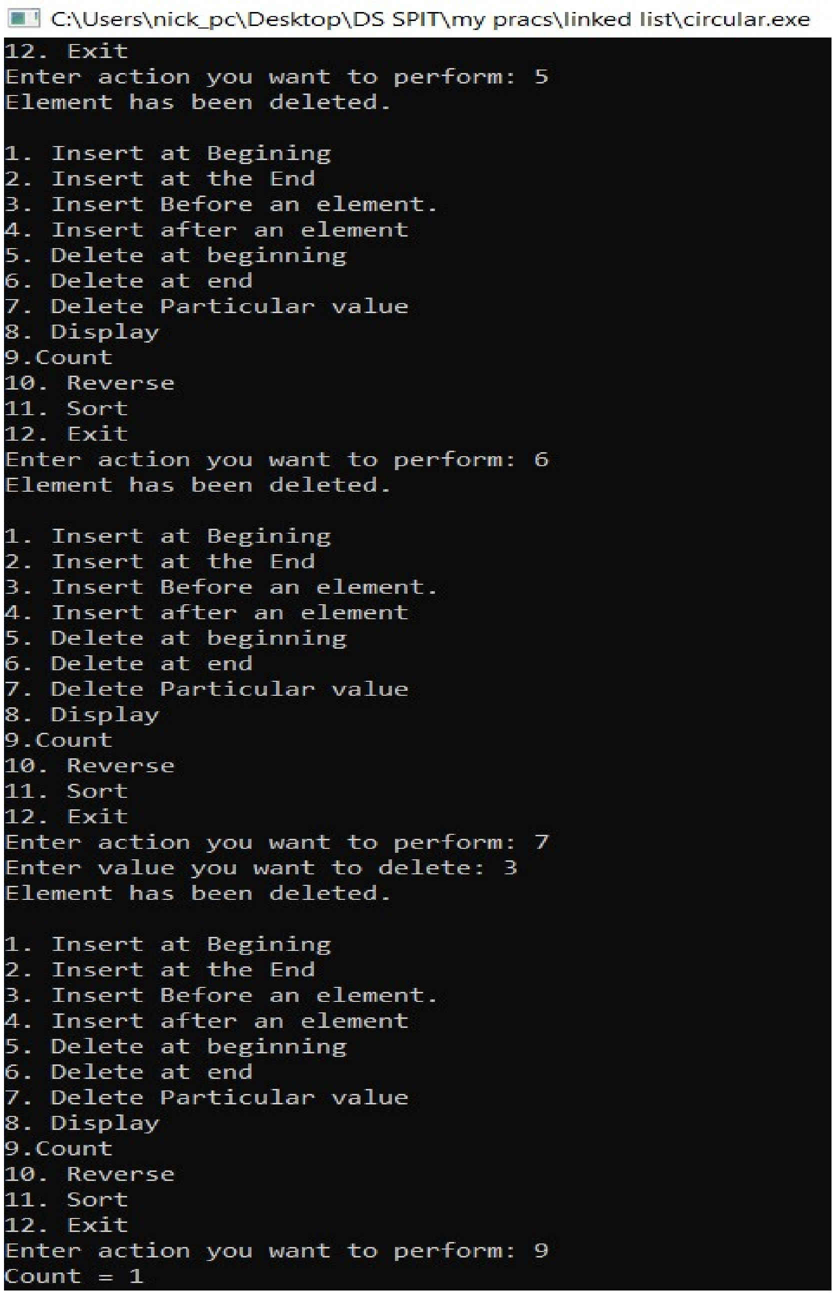
C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\circular.exe

```
1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 1
Enter value you want to insert: 1

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 2
Enter value you want to insert: 4

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 3
Enter element before you want to insert the value: 4
Enter value you want to insert: 3
```



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\circular.exe

```
11. Sort
12. Exit
Enter action you want to perform: 4
Enter element after you want to insert the value: 1
Enter value you want to insert: 2

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 8
Elements in the linked list are: 1 2 3 4

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 5
Element has been deleted.

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 6
Element has been deleted.
```



```
C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\circular.exe
12. Exit
Enter action you want to perform: 5
Element has been deleted.

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 6
Element has been deleted.

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 7
Enter value you want to delete: 3
Element has been deleted.

1. Insert at Begining
2. Insert at the End
3. Insert Before an element.
4. Insert after an element
5. Delete at beginning
6. Delete at end
7. Delete Particular value
8. Display
9.Count
10. Reverse
11. Sort
12. Exit
Enter action you want to perform: 9
Count = 1
```

1.4polynomial_13

Code:-

```
#include<iostream>

#include<malloc.h>

using namespace std;

struct node{

   int data;
```

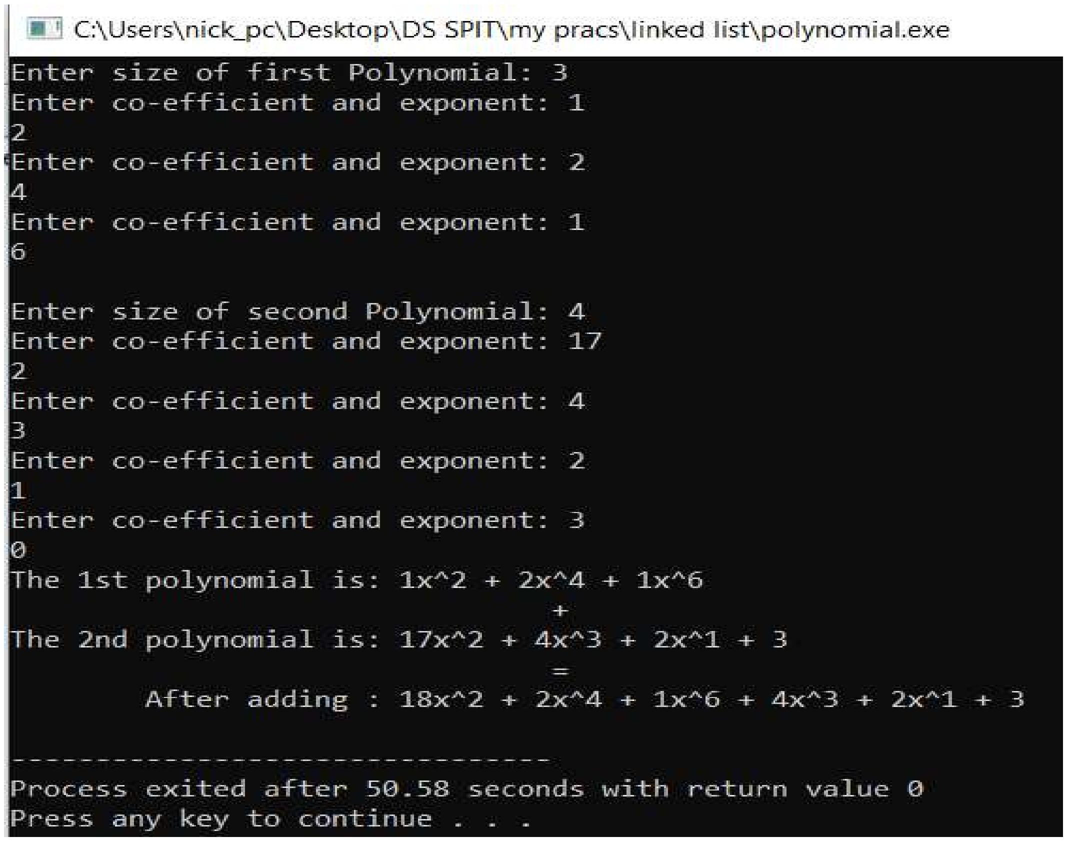
```
    struct node *lptr;

    struct node *rptr;

}*list=NULL, *p, *q, *r, *s;


class nick{

    int action, value, element;

    public:

    mytech(){

        do{

            cout << endl << "1. Insert at Begining\n2. Insert at the End\n3. Insert Before an element.\n4. 
Insert after an element\n5. Delete at beginning\n6. Delete at end\n7. Delete Particular value\n8. 
Display\n9.Count\n10. Reverse\n11. Sort\n12. Exit\nEnter action you want to perform: ";

            cin >> action;

            switch (action)

            {

                case 1:

                    insert_b();

                    break;

                case 2:

                    insert_e();

                    break;

                case 3:

                    insert_be();

                    break;

                case 4:

                    insert_ae();

                    break;

                case 5:

                    delete_b();

                    break;

                case 6:

                    delete_e();
                    break;
                case 7:
                    delete_v();
                    break;
                case 8:
                    display();
                    break;
                case 9:
                    count();
                    break;
                case 10:
                    reverse();
                    break;
                case 11:
                    sort();
                    break;
                case 12:
                    break;
            default:
                break;
            }
        }while(action!=12);
    }

    void insert_b(){
        cout << "Enter value you want to insert: ";
        cin >> value;
        p = (struct node*)malloc(sizeof(node));
        p->data = value;
        if(list==NULL)
            p->lptr = p->rptr = NULL;
        else{
            p->lptr = NULL;
            p->rptr = list;
            list->lptr = p;
        }
        list = p;
    }

    void insert_e(){
        cout << "Enter value you want to insert: ";
        cin >> value;
        p = (struct node*)malloc(sizeof(node));
        q = (struct node*)malloc(sizeof(node));
        p->data = value;
        if(list==NULL){
            p->lptr = p->rptr = NULL;
            list = p;
        }
        else{
            q = list;
            while(q->rptr!=NULL)
                q = q->rptr;
            q->rptr = p;
            p->rptr = NULL;
            p->lptr = q;
        }
    }
    void insert_be(){
        cout << "Enter element before you want to insert the value: ";
        cin >> element;
        cout << "Enter value you want to insert: ";
        cin >> value;
        p = (struct node*)malloc(sizeof(node));
        p->data = value;
        q = list;

        while((q->data != element) && (q->rptr != NULL)){
            r = q;
            q = q->rptr;
        }
        if(q->data==element){
            if(list->data==element){
                p->rptr = q;
                p->lptr = NULL;
                list = p;
            }
            else{
                r->rptr = p;
                p->lptr = r;
                p->rptr = q;
                q->lptr = p;
            }
        }
        else{
            cout << "Data not found.";
        }

    }
    void insert_ae(){
        cout << "Enter element after you want to insert the value: ";
        cin >> element;
        cout << "Enter value you want to insert: ";
        cin >> value;
        p = (struct node*)malloc(sizeof(node));
        p->data = value;
        q = list;
        r = q->rptr;
        while((q->data != element) && (q->rptr != NULL)){
            q = r;
            r = r->rptr;
        }
        if(q->data != element)
            cout << "Data not found.";
        else if(r == NULL){
            q->rptr = p;
            p->lptr = q;
            p->rptr = NULL;
        }
        else{
            q->rptr = p;
            p->lptr = q;
            p->rptr = r;
            r->lptr = p;
        }
    }

    void delete_b(){
        if(list==NULL)
            cout << "List is empty nothing to delete.";
        else{
            list = list->rptr;
            cout << "Element has been deleted." << endl;
        }
```

```
}

void delete_e(){
    p = list;
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        while(p->rptr != NULL){
            q = p;
            p = p->rptr;
        }
        q->rptr = NULL;
        cout << "Element has been deleted." << endl;
    }
}

void delete_v(){
    p = list;
    q = p;
    if(list==NULL)
        cout << "List is empty nothing to delete.";
    else{
        cout << "Enter value you want to delete: ";
        cin >> value;
        while(p->data != value && p->rptr!=NULL){
            q = p;
            p = p->rptr;
        }
        if(p->data != value)
            cout << "Data not found.";
        else{
            if(p==q)
               list = p->rptr;
            else
               q->rptr = p->rptr;
            cout << "Element has been deleted." << endl;
         }
      }
   }

   void display(){
      if(list==NULL)
         cout << "No Element in the linked list.";
      else{
         p = list;
         cout << "Elements in the linked list are: ";
         while(p!=NULL){
            cout << p->data << " ";
            p = p->rptr;
         }
      }
   }

   void count(){
      int count = 0;
      p = list;
      while(p != NULL){
         count++;
         p = p->rptr;
      }
      cout << "Count = " << count << endl;
   }
```

```
void reverse(){
    if(list==NULL)
        cout << "No Element in the linked list.";
    else{
        struct node *temp;
        q = s = list;
        temp = NULL;
        r = q->rptr;
        s->lptr = r;
        while(r!=NULL){
            temp = q;
            q = r;
            r = q->rptr;
            q->rptr = temp;
        }
        list = q;
        s->rptr = NULL;
        cout << "Linked List has been reversed.";
    }
}

void sort(){
    p = list;
    int temp;
    while(p->rptr!=NULL){
        q = p->rptr;
        while(q->rptr!=NULL){
            if(p->data > q->data){
                temp = p->data;
                p->data = q->data;
```

```
              q->data = temp;
            }
            q = q->rptr;
          }
          p = p->rptr;
        }
    }
};

int main(){
    nick o;
           o.mytech();
}
```

Output:-



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\linked list\polynomial.exe

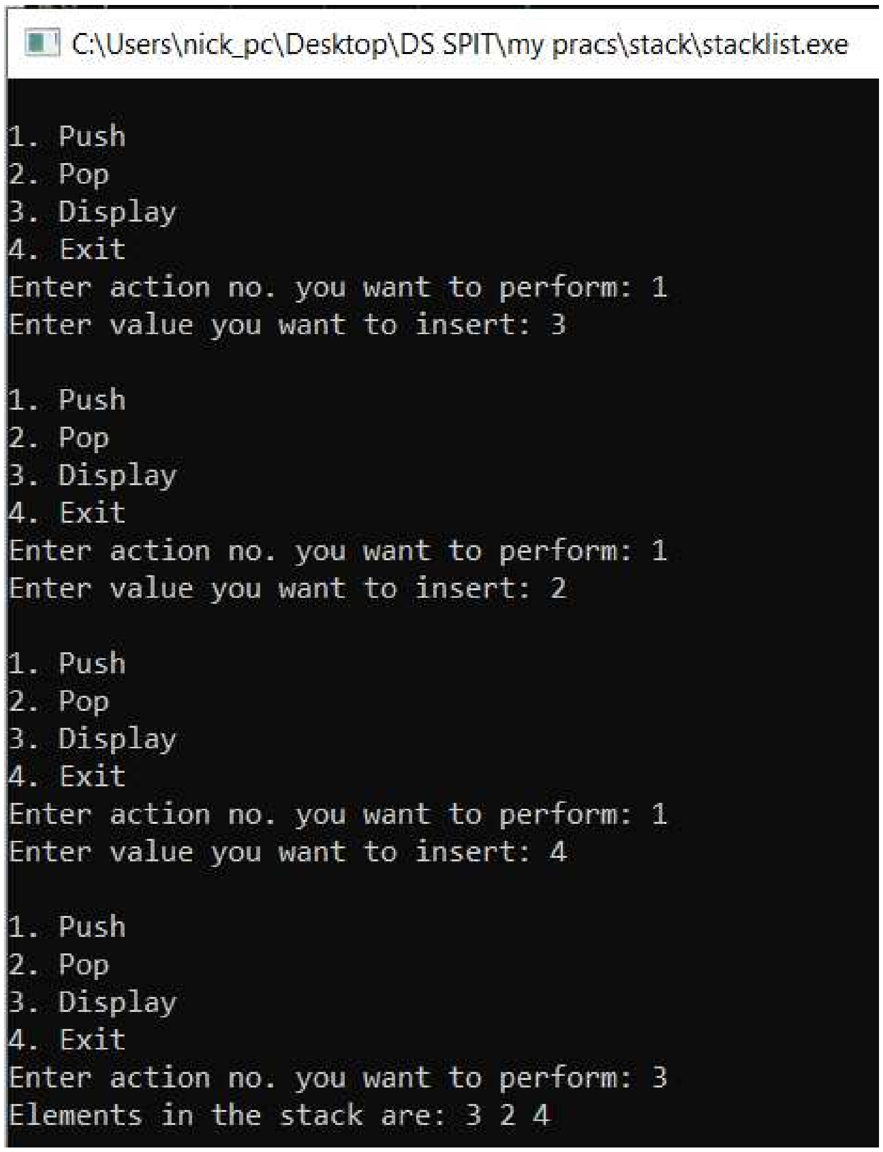
```
Enter size of first Polynomial: 3
Enter co-efficient and exponent: 1
2
Enter co-efficient and exponent: 2
4
Enter co-efficient and exponent: 1
6

Enter size of second Polynomial: 4
Enter co-efficient and exponent: 17
2
Enter co-efficient and exponent: 4
3
Enter co-efficient and exponent: 2
1
Enter co-efficient and exponent: 3
0
The 1st polynomial is: 1x^2 + 2x^4 + 1x^6
                                  +
The 2nd polynomial is: 17x^2 + 4x^3 + 2x^1 + 3
                                  =
        After adding : 18x^2 + 2x^4 + 1x^6 + 4x^3 + 2x^1 + 3

--------------------------------
Process exited after 50.58 seconds with return value 0
Press any key to continue . . .
```

1.5Stack_Link_List_13

Code:-

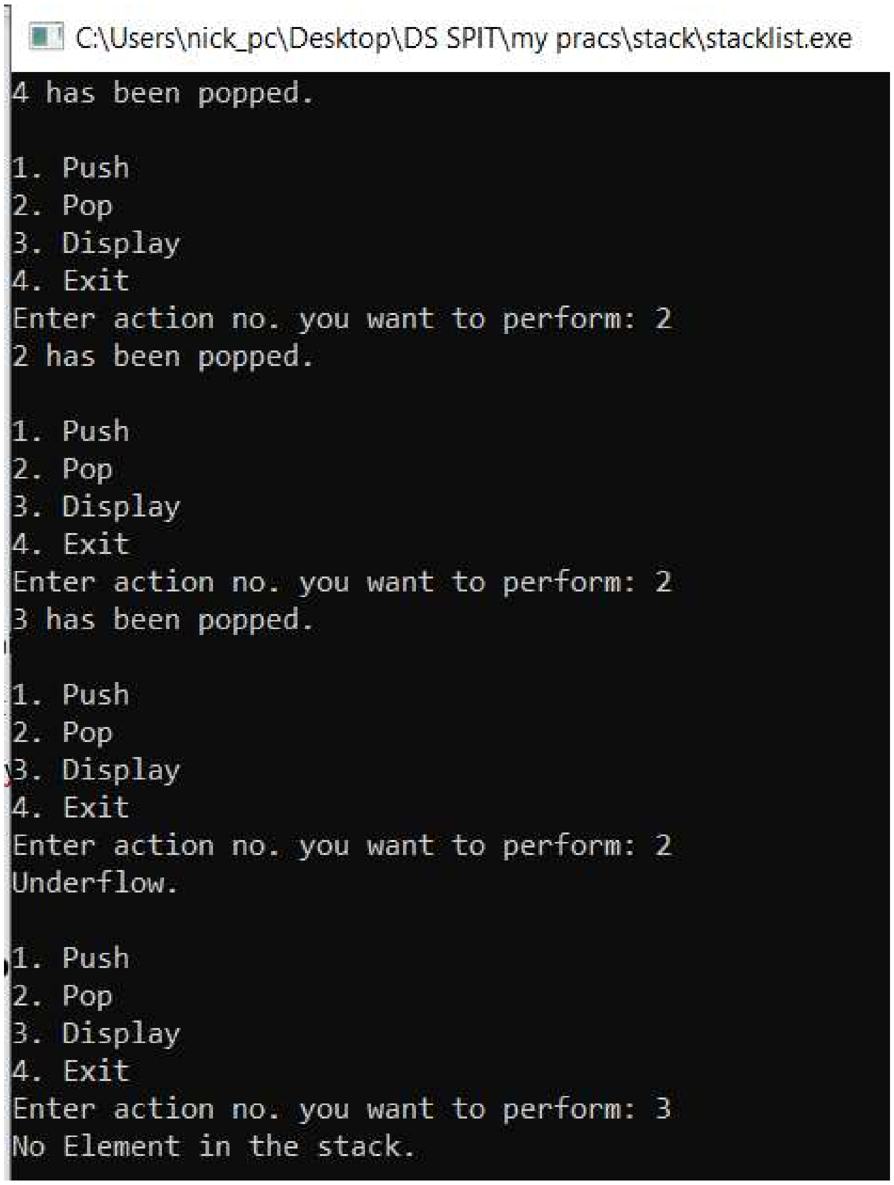
```
#include<iostream>
#include<malloc.h>
using namespace std;
struct node{
  int data;
  struct node *next;
}*list = NULL, *top = NULL, *p, *q;

class nick{
  int action, value;
  string str;
  public:
    void mytech(){
      do{
        cout << "\n1. Push\n2. Pop\n3. Display\n4. Exit\nEnter action no. you want to perform: ";
        cin >> action;
        switch (action)
        {
          case 1:
            push();
            break;
          case 2:
            pop();
            break;
          case 3:
            display();
            break;
          case 4:
            break;
          default:
            cout << "Invalid input.";
        }
      }while(action != 4);
    }

    void push(){
      cout << "Enter value you want to insert: ";
      cin >> value;
      p = (struct node*)malloc(sizeof(node));
      p->data = value;
      p->next = NULL;
      top = p;
      if(list==NULL)
        list = p;
      else{
        q = list;
        while(q->next!=NULL)
          q = q->next;
        q->next = p;
      }
    }

    void pop(){
      if(list==NULL)
        cout << "Underflow.\n";
      else{
        cout << top->data << " has been popped." << endl;
        q = list;
        if(q->next==NULL)
          list = top = NULL;
        else{
                while(q->next != top && q->next!=NULL)
                    q = q->next;
                free(top);
                top = q;
                q->next=NULL;
            }
        }
    }

    void display(){
    if(list==NULL)
        cout << "No Element in the stack." << endl;
    else{
        p = list;
        cout << "Elements in the stack are: ";
        while(p!=NULL){
            cout << p->data << " ";
            p = p->next;
        }
        cout << endl;
    }
  }
};

int main(){
    nick o;
    o.mytech();

}
```

Output:-



```
C:\Users\nick_pc\Desktop\DS SPIT\my pracs\stack\stacklist.exe

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 1
Enter value you want to insert: 3

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 1
Enter value you want to insert: 2

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 1
Enter value you want to insert: 4

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 3
Elements in the stack are: 3 2 4
```

Linked list Programs



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\stack\stacklist.exe

```
4 has been popped.

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 2
2 has been popped.

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 2
3 has been popped.

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 2
Underflow.

1. Push
2. Pop
3. Display
4. Exit
Enter action no. you want to perform: 3
No Element in the stack.
```

1.6Queue_Link_List_13

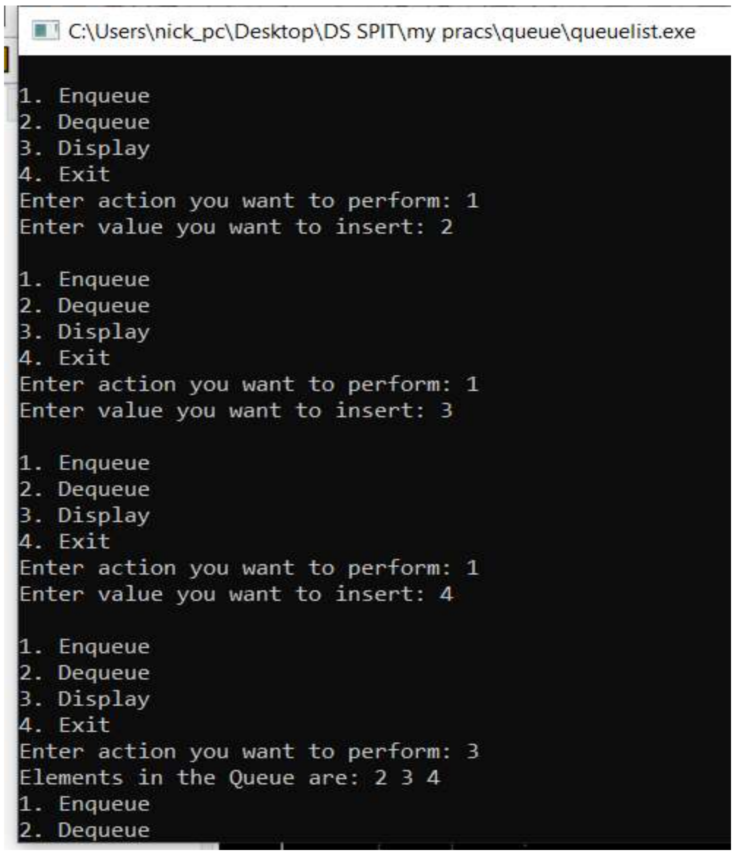
Code:-

```
#include<iostream>

#include<malloc.h>

using namespace std;


struct node{

   int data;

   struct node *next;

}*front=NULL, *rear=NULL, *p, *q, *r, *s;
```

```
class nick{
   int action, value;

   public:
   mytech(){
      do{
         cout << "\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nEnter action you want to perform: ";
         cin >> action;
         switch (action)
         {
         case 1:
            enqueue();
            break;

         case 2:
            dequeue();
            break;

         case 3:
            display();
            break;

         case 4:
            break;

         default:
            cout << "Invalid input." << endl;
            break;
         }
      }while(action != 4);
   }
   void enqueue(){
        cout << "Enter value you want to insert: ";
        cin >> value;
        p = (struct node*)malloc(sizeof(node));
        p->data = value;
        p->next = NULL;
        if(front==NULL){
            front = p;
            front = rear = p;
        }
        else{
            q = front;
            while(q->next!=NULL)
                q = q->next;
            q->next = p;
            rear = p;
        }
    }

    void dequeue(){
        if(front==NULL)
            cout << "Underflow." << endl;
        else{
            cout << front->data << " has been removed." << endl;
            front = front->next;
        }
    }

    void display(){
        if(front==NULL)
            cout << "No Element in the Queue.";
        else{
            p = front;
            cout << "Elements in the Queue are: ";
```

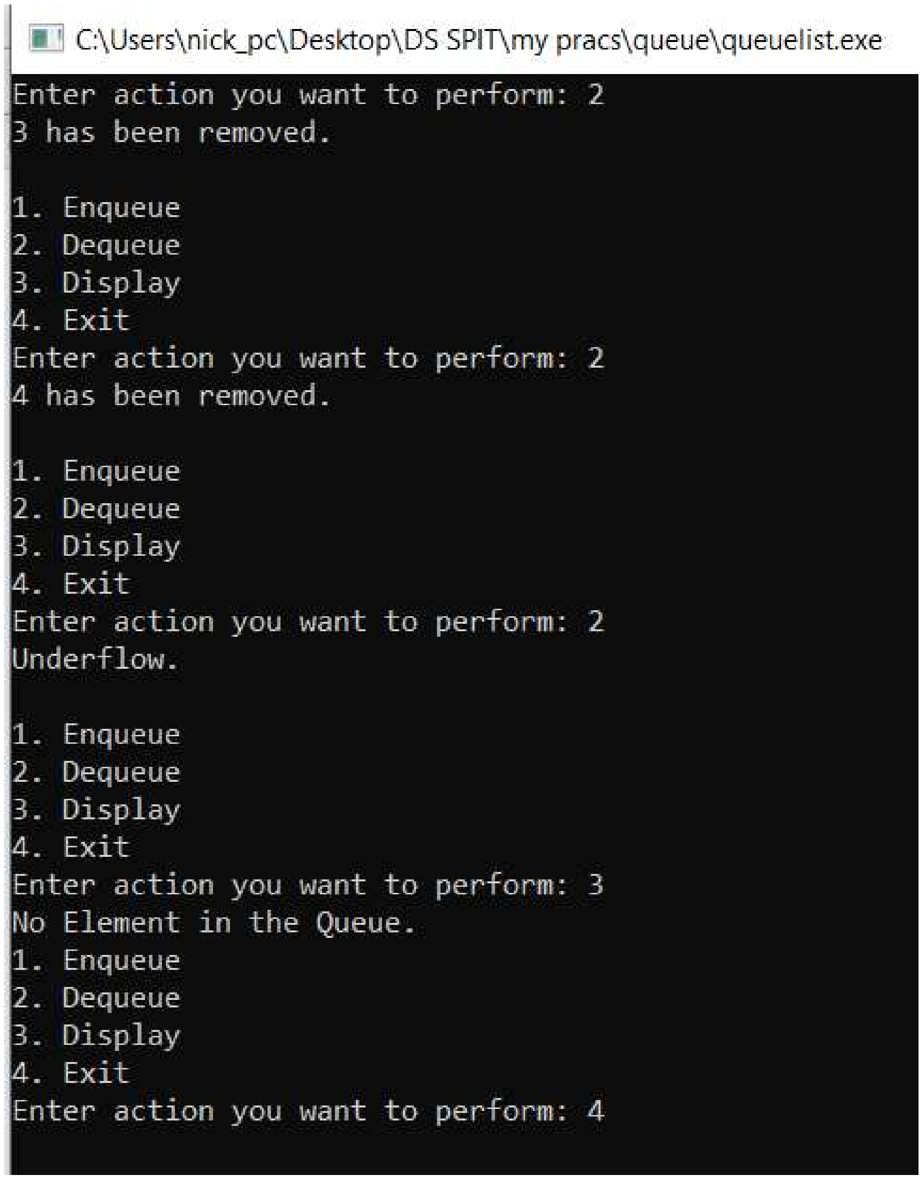
```
      while(p!=NULL){
        cout << p->data << " ";
        p = p->next;
      }
    }
  }
};

int main(){
  nick o;
  o.mytech();
}
```

Output:-



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\queue\queuelist.exe

```
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 1
Enter value you want to insert: 2

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 1
Enter value you want to insert: 3

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 1
Enter value you want to insert: 4

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 3
Elements in the Queue are: 2 3 4
1. Enqueue
2. Dequeue
```



C:\Users\nick_pc\Desktop\DS SPIT\my pracs\queue\queuelist.exe

```
Enter action you want to perform: 2
3 has been removed.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 2
4 has been removed.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 2
Underflow.

1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 3
No Element in the Queue.
1. Enqueue
2. Dequeue
3. Display
4. Exit
Enter action you want to perform: 4
```