```cpp

/* ==============================================
** #1. A C++ program of implementation of Stack.
** ==============================================*/

#include <iostream>
using namespace std;

struct Node
{
   int data;
   struct Node *next;
};
struct Node* top = NULL;

void push(int val)
{
   struct Node* newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->data = val;
   newnode->next = top;
   top = newnode;
}

void pop()
{
   if(top==NULL)
      cout<<"Stack Underflow"<<endl;
   else
   {
      cout<<"The popped element is "<< top->data <<endl;
      top = top->next;
   }
}

void display()
{
   struct Node* ptr;
   if(top==NULL)
      cout<<"stack is empty";
   else
   {
      ptr = top;
      cout<<"Stack elements are: ";
      while (ptr != NULL)
      {
         cout<< ptr->data <<" ";
         ptr = ptr->next;
      }
   }
   cout<<endl;
}

int main()
{
   int ch, val;
   cout<<"1) Push in stack"<<endl;
   cout<<"2) Pop from stack"<<endl;
   cout<<"3) Display stack"<<endl;
   cout<<"4) Exit"<<endl;
   do
   {
      cout<<"Enter choice: "<<endl;
      cin>>ch;
      switch(ch)
      {
         case 1:
         {
            cout<<"Enter value to be pushed:"<<endl;
            cin>>val;
            push(val);
            break;
         }
         case 2:
         {
            pop();
            break;
         }
         case 3:
         {
            display();
            break;
         }
         case 4:
         {
            cout<<"Exit"<<endl;
            break;
         }
         default:
         {
            cout<<"Invalid Choice"<<endl;
         }
      }
   }while(ch!=4);
      return 0;
}


/* ==============================================
** #2. A C++ program of implementation of Queue.
** =============================================*/
#include <iostream>
using namespace std;
struct node {
   int data;
   struct node *next;
};

struct node* front = NULL;
struct node* rear = NULL;
struct node* temp;

void push() {
   int val;
   cout<<"push the element in queue : "<<endl;
   cin>>val;
   if (rear == NULL) {
      rear = (struct node *)malloc(sizeof(struct node));
      rear->next = NULL;
      rear->data = val;
      front = rear;
   } else {
      temp=(struct node *)malloc(sizeof(struct node));
      rear->next = temp;
      temp->data = val;
      temp->next = NULL;
      rear = temp;
   }
}

void pop() {
   temp = front;
   if (front == NULL) {
      cout<<"Underflow"<<endl;
      return;
   }
   else
   if (temp->next != NULL) {
      temp = temp->next;
      cout<<"Element popped from queue is : "<<front->data<<endl;
      free(front);
      front = temp;
   } else {
      cout<<"Element popped from queue is : "<<front->data<<endl;
      free(front);
      front = NULL;
      rear = NULL;
      }
}

void Display() {
   temp = front;
   if ((front == NULL) && (rear == NULL)) {
      cout<<"Queue is empty"<<endl;
      return;
   }
   cout<<"Queue elements are: ";
   while (temp != NULL) {
      cout<<temp->data<<" ";
      temp = temp->next;
   }
   cout<<endl;
}

int main() {
   int ch;
   cout<<"1) push element to queue"<<endl;
   cout<<"2) pop element from queue"<<endl;
   cout<<"3) Display all the elements of queue"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter your choice : "<<endl;
      cin>>ch;
      switch (ch) {
         case 1: push();
         break;
         case 2: pop();
         break;
         case 3: Display();
         break;
         case 4: cout<<"Exit"<<endl;
         break;
         default: cout<<"Invalid choice"<<endl;
      }
   } while(ch!=4);
   return 0;
}


/* =====================================================
 * #3. A C++ Program to Implement Stack Using Queue
 * =====================================================*/

#include<stdio.h>
#include<iostream>
#include<conio.h>
using namespace std;

struct queue1
{
    queue1 *next1;
    int data1;
}*front1 = NULL, *rear1 = NULL, *q1 = NULL, *p1 = NULL, *np1 = NULL;

struct queue2
{
    queue2 *next2;
    int data2;
}*front2 = NULL, *rear2 = NULL, *q2 = NULL, *p2 = NULL, *np2 = NULL;

void enqueue1(int x)
{
    np1 = new queue1;
    np1->data1 = x;
    np1->next1 = NULL;
    if (front1 == NULL)
    {
        rear1 = np1;
        rear1->next1 = NULL;
        front1 = rear1;
    }
    else
    {
        rear1->next1 = np1;
        rear1 = np1;
        rear1->next1 = NULL;
    }
}

```

```cpp
int dequeue1()
{
    int x;
    if (front1 == NULL)
    {
        cout<<"no elements present in queue\n";
    }
    else
    {
        q1 = front1;
        front1 = front1->next1;
        x = q1->data1;
        delete(q1);
        return x;
    }
}

void enqueue2(int x)
{
    np2 = new queue2;
    np2->data2 = x;
    np2->next2 = NULL;
    if (front2 == NULL)
    {
        rear2 = np2;
        rear2->next2 = NULL;
        front2=rear2;
    }
    else
    {
        rear2->next2 = np2;
        rear2 = np2;
        rear2->next2 = NULL;
    }
}

int dequeue2()
{
    int x;
    if (front2 == NULL)
    {
        cout<<"no elements present in queue\n";
    }
    else
    {
        q2 = front2;
        front2 = front2->next2;
        x = q2->data2;
        delete(q2);
        return x;
    }
}

int main()
{
    int n, x, i = 0;
    cout<<"Enter the number of elements to be entered into stack\n";
    cin>>n;
    while (i < n)
    {
        cout<<"enter the element to be entered\n";
        cin>>x;
        enqueue1(x);
        i++;
    }
    cout<<"\n\nElements popped\n\n";
    while (front1 != NULL || front2 != NULL)
    {
        if (front2 == NULL)
        {
            while (front1->next1 != NULL)
            {
                enqueue2(dequeue1());
            }
            cout<<dequeue1()<<endl;
        }
        else if (front1 == NULL)
        {
            while (front2->next2 != NULL)
            {
                enqueue1(dequeue2());
            }
            cout<<dequeue2()<<endl;
        }
    }
    getch();
}

/* ====================================================
 * #4. A C++ Program to Implement Queue Using Stack
 * ====================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <iostream>

using namespace std;

struct sNode
{
    int data;
    struct sNode *next;
};

void push(struct sNode **top_ref, int new_data);

int pop(struct sNode **top_ref);

struct queue
{
    struct sNode *stack1;
    struct sNode *stack2;
};

```

```
void enQueue(struct queue *q, int x)
{
    push(&q->stack1, x);
}

int deQueue(struct queue *q)
{
    int x;

    if (q->stack1 == NULL && q->stack2 == NULL)
    {
        cout << "Queue is empty";
        exit(0);
    }
    if (q->stack2 == NULL)
    {
        while (q->stack1 != NULL)
        {
            x = pop(&q->stack1);
            push(&q->stack2, x);
        }
    }

    x = pop(&q->stack2);
    return x;
}

void push(struct sNode **top_ref, int new_data)
{

    struct sNode *new_node = (struct sNode *)malloc(sizeof(struct sNode));

    if (new_node == NULL)
    {
        cout << "Stack overflow \n";
        exit(0);
    }


    new_node->data = new_data;

    new_node->next = (*top_ref);

    (*top_ref) = new_node;
}

int pop(struct sNode **top_ref)
{
    int res;
    struct sNode *top;

    if (*top_ref == NULL)
    {
        cout << "Stack overflow \n";
        exit(0);
    }
    else
```

```cpp
    {
        top = *top_ref;
        res = top->data;
        *top_ref = top->next;
        free(top);
        return res;
    }
}

int main()
{

    struct queue *q = (struct queue *)malloc(sizeof(struct queue));
    q->stack1 = NULL;
    q->stack2 = NULL;
    cout << "Enqueuing...";
    cout << endl;
    enQueue(q, 1);
    cout << "Enqueuing...";
    cout << endl;
    enQueue(q, 2);
    cout << "Enqueuing...";
    cout << endl;
    enQueue(q, 3);

    cout << "Dequeuing...";
    cout << deQueue(q) << " ";
    cout << endl;
    cout << "Dequeuing...";
    cout << deQueue(q) << " ";
    cout << endl;
    cout << "Dequeuing...";
    cout << deQueue(q) << " ";
    cout << endl;
}


/* ====================================================
 * #5. A C++ program of Implementation of Linked List
 * ====================================================*/

#include <iostream>
using namespace std;

struct Node
{
    int data;
    Node *next;
};

class LinkedList
{

    Node *head;

public:

    LinkedList()
    {
        head = NULL;
    }


    void insert(int val)
    {
        Node *new_node = new Node;
        new_node->data = val;
        new_node->next = NULL;

        if (head == NULL)
            head = new_node;
        else
        {
            new_node->next = head;
            head = new_node;
        }
    }

    bool search(int val)
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->data == val)
                return true;
            temp = temp->next;
        }
        return false;
    }

    void remove(int val)
    {
        if (head->data == val)
        {
            delete head;
            head = head->next;
            return;
        }

        if (head->next == NULL)
        {
            if (head->data == val)
            {
                delete head;
                head = NULL;
                return;
            }
            cout << "Value not found!" << endl;
            return;
        }

        Node *temp = head;
        while (temp->next != NULL)
        {
```

```cpp
            if (temp->next->data == val)
            {
                Node *temp_ptr = temp->next->next;
                delete temp->next;
                temp->next = temp_ptr;
                return;
            }
            temp = temp->next;
        }

        cout << "Value not found" << endl;
    }

    void display()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main()
{

    LinkedList l;
    // inserting elements
    l.insert(6);
    l.insert(9);
    l.insert(1);
    l.insert(3);
    l.insert(7);
    cout << "Current Linked List: ";
    l.display();

    cout << "Deleting 1: ";
    l.remove(1);
    l.display();

    cout << "Deleting 13: ";
    l.remove(13);

    cout << "Searching for 7: ";
    cout << l.search(7) << endl;

    cout << "Searching for 13: ";
    cout << l.search(13) << endl;
}


/* ==============================================
 * #6. A C++ program of Reverse Linked List
 * ==============================================*/

```

```cpp
#include <iostream>
using namespace std;
struct Node
{
    int data;
    struct Node *next;
    Node(int data)
    {
        this->data = data;
        next = NULL;
    }
};
struct LinkedList
{
    Node *head;
    LinkedList()
    {
        head = NULL;
    }
    void reverse()
    {
        Node *current = head;
        Node *prev = NULL, *next = NULL;
        while (current != NULL)
        {
            next = current->next;
            current->next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }
    void print()
    {
        struct Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
    }
    void push(int data)
    {
        Node *temp = new Node(data);
        temp->next = head;
        head = temp;
    }
};
int main()
{
    LinkedList ll;
    ll.push(40);
    ll.push(30);
    ll.push(20);
    ll.push(10);
    cout << "old linked list\n";
    ll.print();
    ll.reverse();
    cout << "\nnew Linked list \n";
    ll.print();
    return 0;
}

/* ==========================================================
 * #7. A C++ program of Implementation of Singly Linked List.
 * ==========================================================*/


#include <iostream>
using namespace std;


struct Node
{
    int data;
    Node *next;
};

class LinkedList
{
    Node *head;

public:
    LinkedList()
    {
        head = NULL;
    }

    void insert(int val)
    {
        Node *new_node = new Node;
        new_node->data = val;
        new_node->next = NULL;

        if (head == NULL)
            head = new_node;
        else
        {
            new_node->next = head;
            head = new_node;
        }
    }

    bool search(int val)
    {
        Node *temp = head;
        while (temp != NULL)
        {
            if (temp->data == val)
                return true;
            temp = temp->next;
        }
        return false;
    }
```

```cpp

    void remove(int val)
    {
        if (head->data == val)
        {
            delete head;
            head = head->next;
            return;
        }

        if (head->next == NULL)
        {
            if (head->data == val)
            {
                delete head;
                head = NULL;
                return;
            }

            cout << "Value not found!" << endl;
            return;
        }

        Node *temp = head;
        while (temp->next != NULL)
        {
            if (temp->next->data == val)
            {
                Node *temp_ptr = temp->next->next;
                delete temp->next;
                temp->next = temp_ptr;
                return;
            }
            temp = temp->next;
        }

        cout << "Value not found" << endl;
    }

    void display()
    {
        Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main()
{

    LinkedList l;
    // inserting elements
    l.insert(6);
    l.insert(9);
    l.insert(1);
    l.insert(3);
    l.insert(7);
    cout << "Current Linked List: ";
    l.display();

    cout << "Deleting 1: ";
    l.remove(1);
    l.display();

    cout << "Deleting 13: ";
    l.remove(13);

    cout << "Searching for 7: ";
    cout << l.search(7) << endl;

    cout << "Searching for 13: ";
    cout << l.search(13) << endl;
}


/* ==========================================================
 * #8. A C++ program of Implementation of Doubly Linked List.
 * ==========================================================*/

#include <iostream>
#include "stdio.h"
#include "conio.h"

using namespace std;
int insertdata(int x);
void display();
void deleteint(int x);
void reversel();
int searchint(int x);
int compare_fn(int a, int b)
{
    if (a > b)
        return 1;
    else if (b > a)
        return -1;
}
int compare_no = 1;
struct node
{
    int data;
    node *prev;
    node *next;
};
node *top = NULL;
int main()
{
    int ch, d, y;
    char ans = 'y';
    while (ans == 'y')
    {
```

```cpp
        cout << "\n\t 1.Insert        2. Delete        3.Reverse      4.EXIT\nEnter Choice : 
";
        cin >> ch;
        if (ch == 1)
        {
            cout << "Enter An Element To be inserted : ";
            cin >> d;
            d = insertdata(d);
            display();
        }
        else if (ch == 2)
        {
            cout << "Enter Element To Be Deleted : ";
            cin >> d;
            deleteint(d);
            display();
        }
        else if (ch == 3)
            reversel();
        else
            return 0;
    }
    return 0;
}
int searchint(int x)
{
    int count = 0;
    node *searchele = top;
    while (searchele != NULL)
    {
        if (compare_fn(x, searchele->data) == compare_no)
        {
            searchele = searchele->next;
            count += 1;
        }
        else
            break;
    }
    return count;
}
int insertdata(int x)
{
    if (top == NULL)
    {
        top = new node;
        top->data = x;
        top->next = NULL;
        top->prev = NULL;
    }
    else if (compare_fn(top->data, x) == compare_no)
    {
        node *n = new node;
        n->data = x;
        n->next = top;
        n->prev = NULL;
        top->prev = n;
        top = n;
    }
    else
    {
        int c = searchint(x);
        node *insertele = top;
        for (int i = 0; i < c - 1; i++)
            insertele = insertele->next;
        node *n = new node;
        n->data = x;
        node *b = insertele->next;
        node *N = insertele;
        n->prev = insertele;
        n->next = b;
        insertele->next = n;
        if (b != NULL)
            b->prev = n;
    }
}
void display()
{
    cout << "Element In The Linked List Are : ";
    node *disp = top;
    while (disp != NULL)
    {
        cout << " " << disp->data;
        if (disp->next == NULL)
        {
            break;
        }
        disp = disp->next;
    }
}
void deleteint(int x)
{
    node *del = top;
    if (del->data == x)
    {
        if (del->next == NULL && del->prev == NULL)
        {
            top = NULL;
            return;
        }
        del->next->prev = NULL;
        top = del->next;
    }
    else
    {
        node *delsuc = del->next;
        if (del == NULL)
        {
            cout << "\nElement Not Found\n";
            return;
        }
        if (delsuc == NULL)
        {
            cout << "\nElement Not Found\n";
            return;
        }
        while (delsuc->data != x)
        {
            del = del->next;
            delsuc = delsuc->next;
            if (del == NULL)
            {
                cout << "\nElement Not Found\n";
                return;
            }
            if (delsuc == NULL)
            {
                cout << "\nElement Not Found\n";
                return;
            }
        }
        del->next = delsuc->next;
        if (delsuc->next != NULL)
            delsuc->next->prev = del;
    }
}
void reversel()
{
    node *a = top;
    node *b, *c, *d;
    while (a != NULL)
    {
        d = a;
        c = a->next;
        b = a->prev;
        a->prev = a->next;
        a->next = b;
        a = c;
    }
    top = d;
    cout << "After Reversing the linked list";
    display();
    compare_no *= -1;
}
```