DSA Lab03

23K2001

M.Muzammil Siddiqui

BCS-3J

Q1:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == nullptr){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = tail->getNext();

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

};

int main(){

    int arr[] = {3,1,2,5,8};

    cout<<"Array:"<<endl;

    for(int i:arr)

    cout<<i<<endl;

    singleList arrList;

    for(int i:arr)

    arrList.insertAtEnd(i);

    cout<<endl<<"SingleLinked List:"<<endl;

    arrList.display();

    cout<<endl<<"Adding 9 in end:"<<endl;

    arrList.insertAtEnd(9);

    arrList.display();

    cout<<endl<<"Adding 11 at pos 3:"<<endl;

    arrList.insertAtIndex(3,11);

    arrList.display();

    cout<<endl<<"Adding 4 at start:"<<endl;

    arrList.insertAtStart(4);

    arrList.display();

    cout<<endl<<"Deleting 1,2 & 5:"<<endl;

    arrList.deleteNode(1);

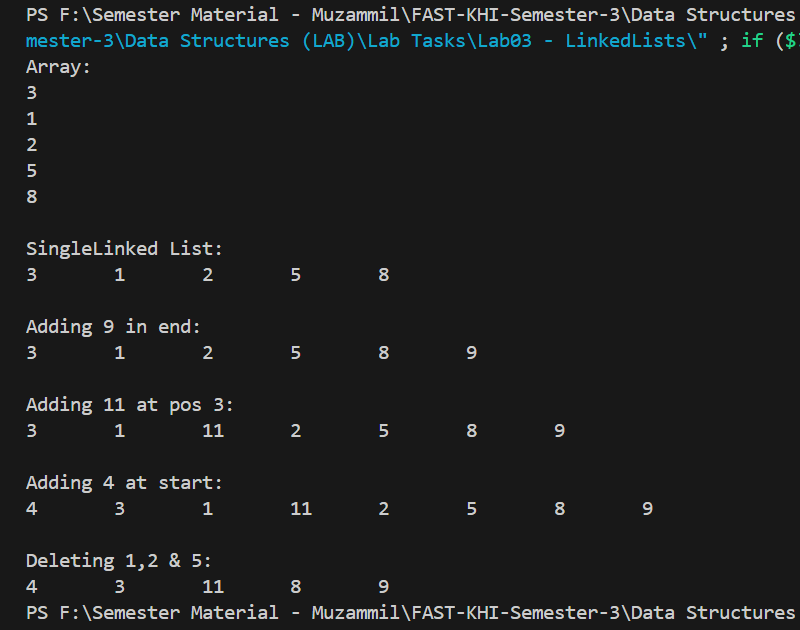
    arrList.deleteNode(2);

    arrList.deleteNode(5);

    arrList.display();

    return 0;

}



Q2:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

        void rotateList(int e) {

            if (head == nullptr || e <= 0) { return; }

            node\* temp = head;

            node\* before = nullptr;

            int n=1;

            while(temp->getNext()!=nullptr){

                temp=temp->getNext();

                n++;

            }

            e=e%n;

            if(e==0){ return; }

            temp=head;

            for (int i=0;i<e;i++){

                before = temp;

                temp = temp->getNext();

            }

            node\* start = temp;

            before->setNext(nullptr);

            node\* end = start;

            while (end->getNext() != nullptr)

                end = end->getNext();

            end->setNext(head);

            head = start;

        }

};

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<"How many elements to move to end? ";

    cin>>e;

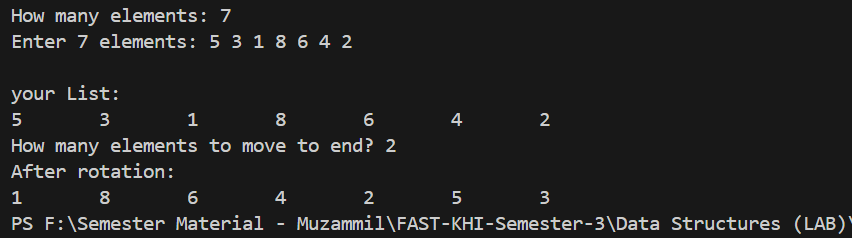
    flex.rotateList(e);

    cout<<"After rotation:"<<endl;

    flex.display();

    return 0;

}



A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

Q3:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        string name;

        node\* next;

    public:

        node(){next = nullptr;}

        node(string val){

            name = val;

            next = nullptr;

        }

        string getData(){ return name;}

        void setData(string n){name = n;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            if(head==nullptr){

                cout<<"No names present in the list."<<endl;

                return;

            }

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\n";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(string val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(string val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

            cout<<"Reservation under name: "<<val<<" has been created."<<endl;

        }

        void insertAtIndex(int index,string val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(string val){

            if(head==nullptr){

            cout<<"No names present in the list."<<endl;

            return;

            }

            if (head->getData()==val){

            node\* temp = head;

            head = head->getNext();

            delete temp;

            cout<<"Reservation under name: "<<val<<" has been cancelled."<<endl;

            return;

            }

            node\* before = nullptr;

            node\* temp = head;

            while(temp!=nullptr && temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

                }

            if(temp==nullptr){

                cout<<"No reservation was found under name: "<<val<<endl;

                return;

                }

            before->setNext(temp->getNext());

            delete temp;

            cout<<"Reservation under name: "<<val<<" has been cancelled."<<endl;

        }

        void check(string val){

            node\* temp = head;

            while(temp!=nullptr && temp->getData()!=val)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No reservation was found under name: "<<val<<endl;

            else

                cout<<"Ticket is reserved under name: "<<val<<endl;

        }

        void sortNames(){

            if (head==nullptr || head->getNext()==nullptr) return;

            bool swapped;

            do{

                swapped = false;

                node\* current = head;

                node\* prev = nullptr;

                while(current!=nullptr && current->getNext()!=nullptr) {

                    if(current->getData() > current->getNext()->getData()){

                        string temp = current->getData();

                        current->setData(current->getNext()->getData());

                        current->getNext()->setData(temp);

                        swapped = true;

                    }

                    prev = current;

                    current = current->getNext();

                }

            } while(swapped);

        }

};

int main(){

    int c;

    string n;

    singleList passengers;

    cout<<"\t\t\*\*\*Welcome to SHAANDAAR Airlines Ticket Reservation System\*\*\*"<<endl;

    do{

        cout<<"1. Reserve a ticket"<<endl;

        cout<<"2. Cancel reservation"<<endl;

        cout<<"3. Check ticket"<<endl;

        cout<<"4. Display passengers"<<endl;

        cout<<"5. Exit"<<endl;

        cout<<"Input choice: ";

        cin>>c;

        switch(c){

            case 1:{

                cout<<"Enter name to reserve a ticket: ";

                cin>>n;

                passengers.insertAtEnd(n);

                passengers.sortNames();

                break;

            }

            case 2:{

                cout<<"Enter name to cancel a reservation: ";

                cin>>n;

                passengers.deleteNode(n);

                break;

            }

            case 3:{

                cout<<"Enter name to check a reservation: ";

                cin>>n;

                passengers.check(n);

                break;

            }

            case 4:{

                passengers.display();

                break;

            }

            case 5:

                break;

            default:{

                cout<<"Invalid choice!"<<endl;

                break;

            }

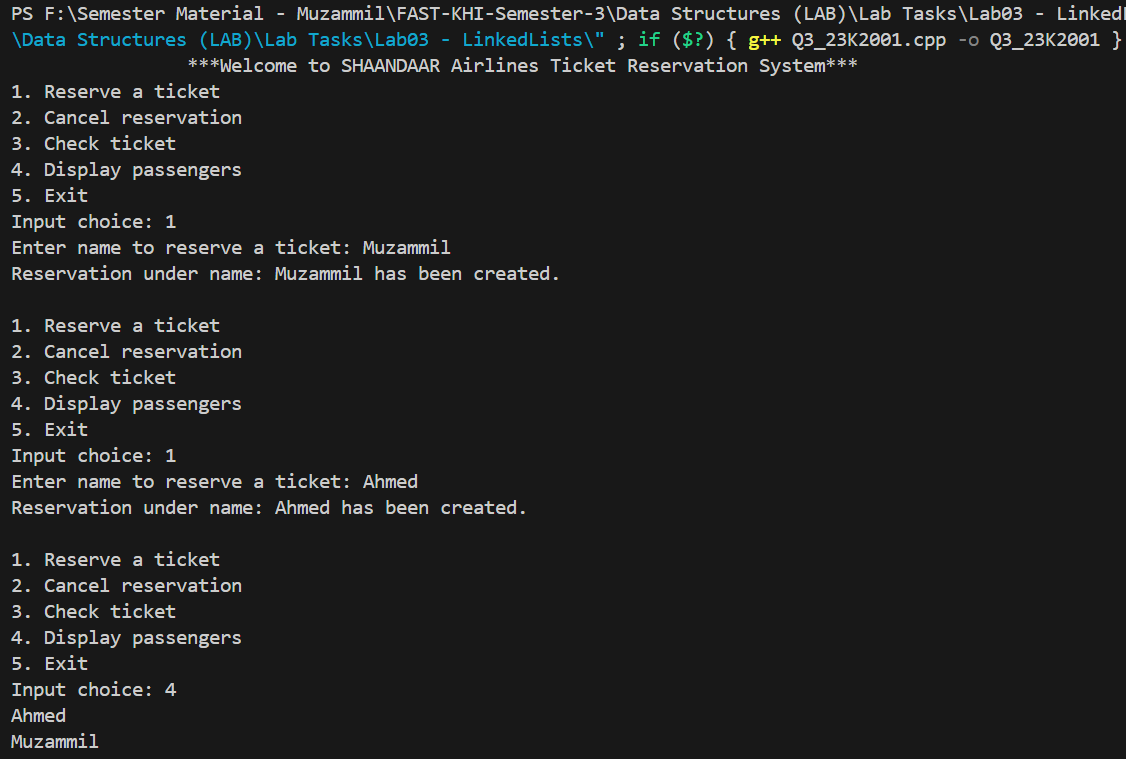
        }

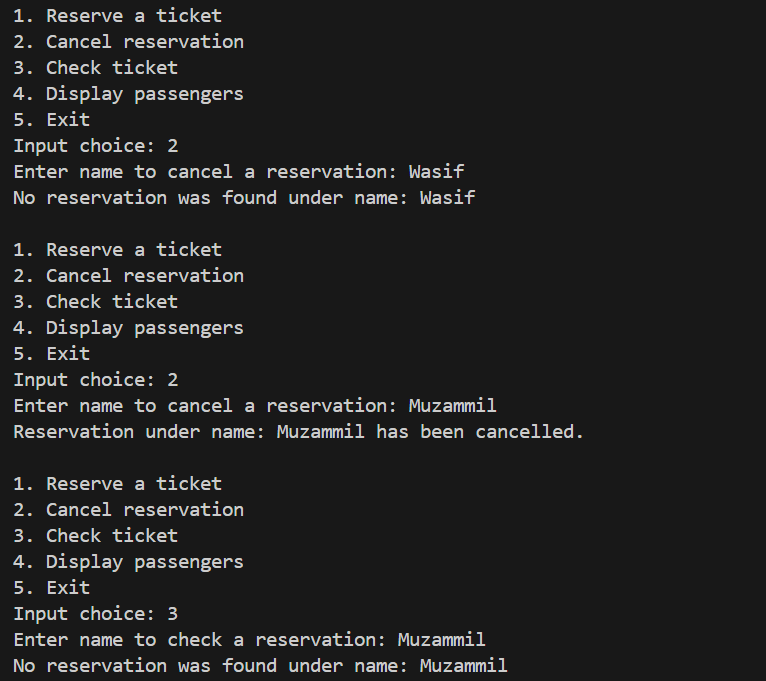
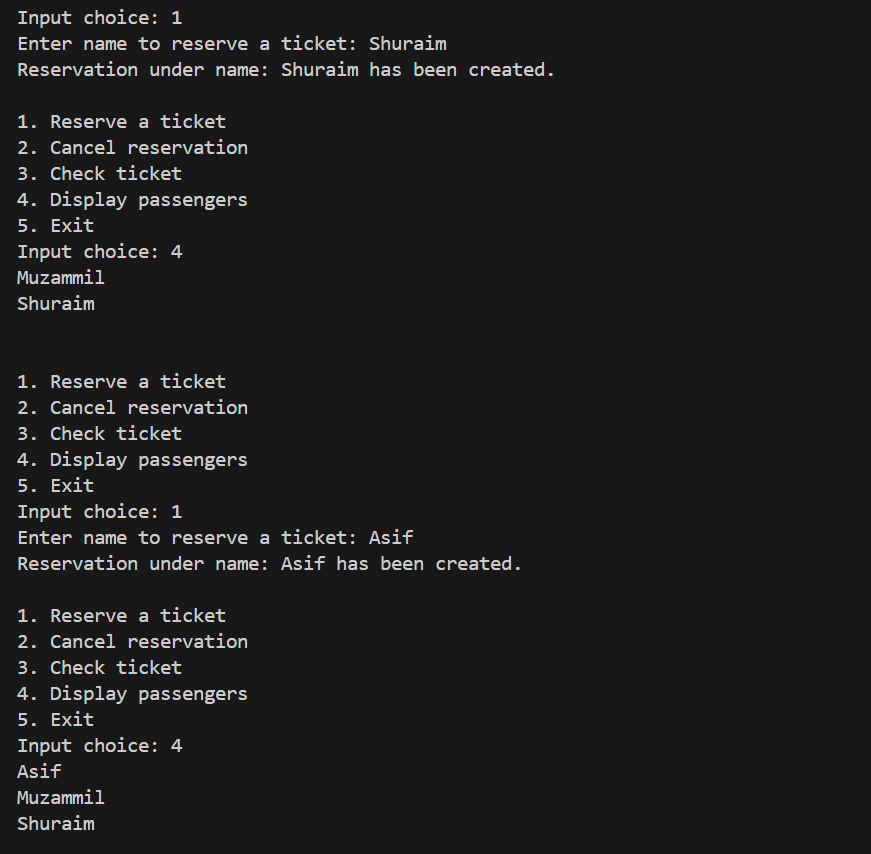
        cout<<endl;

    } while(c != 5);

    return 0;

}





Q3: Full Version

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        string name;

        node\* next;

    public:

        node(){next = nullptr;}

        node(string val){

            name = val;

            next = nullptr;

        }

        string getData(){ return name;}

        void setData(string n){name = n;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            if(head==nullptr){

                cout<<"No names present in the list."<<endl;

                return;

            }

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\n";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(string val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(string val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

            cout<<"Reservation under name: "<<val<<" has been created."<<endl;

        }

        void insertAtIndex(int index,string val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(string val){

            if(head==nullptr){

            cout<<"No names present in the list."<<endl;

            return;

            }

            if (head->getData()==val){

            node\* temp = head;

            head = head->getNext();

            delete temp;

            cout<<"Reservation under name: "<<val<<" has been cancelled."<<endl;

            return;

            }

            node\* before = nullptr;

            node\* temp = head;

            while(temp!=nullptr && temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

                }

            if(temp==nullptr){

                cout<<"No reservation was found under name: "<<val<<endl;

                return;

                }

            before->setNext(temp->getNext());

            delete temp;

            cout<<"Reservation under name: "<<val<<" has been cancelled."<<endl;

        }

        void check(string val){

            node\* temp = head;

            while(temp!=nullptr && temp->getData()!=val)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No reservation was found under name: "<<val<<endl;

            else

                cout<<"Ticket is reserved under name: "<<val<<endl;

        }

        void sortNames(){

            if (head==nullptr || head->getNext()==nullptr) return;

            bool swapped;

            do{

                swapped = false;

                node\* current = head;

                node\* prev = nullptr;

                while(current!=nullptr && current->getNext()!=nullptr) {

                    if(current->getData() > current->getNext()->getData()){

                        string temp = current->getData();

                        current->setData(current->getNext()->getData());

                        current->getNext()->setData(temp);

                        swapped = true;

                    }

                    prev = current;

                    current = current->getNext();

                }

            } while(swapped);

        }

};

class flightNode{

    private:

        string name;

        flightNode\* next;

        singleList\* passengers;

    public:

        flightNode() : next(nullptr),passengers(nullptr){}

        flightNode(string val){

            name = val;

            next = nullptr;

            passengers = nullptr;

        }

        string getData(){ return name;}

        void setData(string n){name = n;}

        flightNode\* getNext(){return next;}

        void setNext(flightNode\* update){next = update;}

        void insertPassenger(string val){

            if(passengers == nullptr){ passengers = new singleList(); }

            passengers->insertAtEnd(val);

            passengers->sortNames();

        }

        void insertPassengers(singleList p){ passengers = &p; }

        void removePassenger(string val){ passengers->deleteNode(val);}

        void checkPassenger(string val){ passengers->check(val); }

        void displayPassengers(){ passengers->display(); }

};

class flightList{

    private:

        flightNode\* head;

        flightNode\* tail;

    public:

        flightList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            if(head==nullptr){

                cout<<"No flights present in the list."<<endl;

                return;

            }

            flightNode\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\n";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtEnd(string val)

        {

            flightNode\* temp = head;

            flightNode\* n = new flightNode(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

            cout<<"Flight under name: "<<val<<" has been added."<<endl;

        }

        void deleteNode(string val){

            if(head==nullptr){

            cout<<"No flights present in the list."<<endl;

            return;

            }

            if (head->getData()==val){

            flightNode\* temp = head;

            head = head->getNext();

            delete temp;

            cout<<"Flight under name: "<<val<<" has been cancelled."<<endl;

            return;

            }

            flightNode\* before = nullptr;

            flightNode\* temp = head;

            while(temp!=nullptr && temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

                }

            if(temp==nullptr){

                cout<<"No flight was found under name: "<<val<<endl;

                return;

                }

            before->setNext(temp->getNext());

            delete temp;

            cout<<"Flight under name: "<<val<<" has been cancelled."<<endl;

        }

        void checkFlight(string val){

            flightNode\* temp = head;

            while(temp!=nullptr && temp->getData()!=val)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No flight was found under name: "<<val<<endl;

            else

                cout<<"Flight is approved under name: "<<val<<endl;

        }

        void insert(string fname,string pname){

            flightNode\* temp = head;

            while(temp!=nullptr && temp->getData()!=fname)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No flight was found under name: "<<fname<<endl;

            else{ temp->insertPassenger(pname); }

        }

        void cancelTicket(string fname,string pname){

            flightNode\* temp = head;

            while(temp!=nullptr && temp->getData()!=fname)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No flight was found under name: "<<fname<<endl;

            else{ temp->removePassenger(pname); }

        }

        void checkTicket(string fname,string pname){

            flightNode\* temp = head;

            while(temp!=nullptr && temp->getData()!=fname)

                temp = temp->getNext();

            if(temp==nullptr)

                cout<<"No flight was found under name: "<<fname<<endl;

            else{ temp->checkPassenger(pname); }

        }

};

int main(){

    int c;

    string n1,n2;

    flightList flights;

    singleList passengers;

    cout<<"\t\t\*\*\*Welcome to SHAANDAAR Airlines Ticket Reservation System\*\*\*"<<endl;

    do{

        cout<<"1. Add a flight"<<endl;

        cout<<"2. Reserve a ticket"<<endl;

        cout<<"3. Cancel reservation"<<endl;

        cout<<"4. Check a ticket"<<endl;

        cout<<"5. Check a flight"<<endl;

        cout<<"6. Display flights"<<endl;

        cout<<"7. Exit"<<endl;

        cout<<"Input choice: ";

        cin>>c;

        switch(c){

            case 1:{

                cout<<"Enter flight name: ";

                cin>>n1;

                flights.insertAtEnd(n1);

                break;

            }

            case 2:{

                cout<<"Enter flight name: ";

                cin>>n1;

                cout<<"Enter name to reserve a ticket: ";

                cin>>n2;

                flights.insert(n1,n2);

                break;

            }

            case 3:{

                cout<<"Enter flight name: ";

                cin>>n1;

                cout<<"Enter name to cancel a ticket: ";

                cin>>n2;

                flights.cancelTicket(n1,n2);

                break;

            }

            case 4:{

                cout<<"Enter flight name: ";

                cin>>n1;

                cout<<"Enter name to check a reservation: ";

                cin>>n2;

                flights.checkTicket(n1,n2);

                break;

            }

            case 5:{

                cout<<"Enter flight name: ";

                cin>>n1;

                flights.checkFlight(n1);

                break;

            }

            case 6:{

                flights.display();

                break;

            }

            case 7:

                break;

            default:{

                cout<<"Invalid choice!"<<endl;

                break;

            }

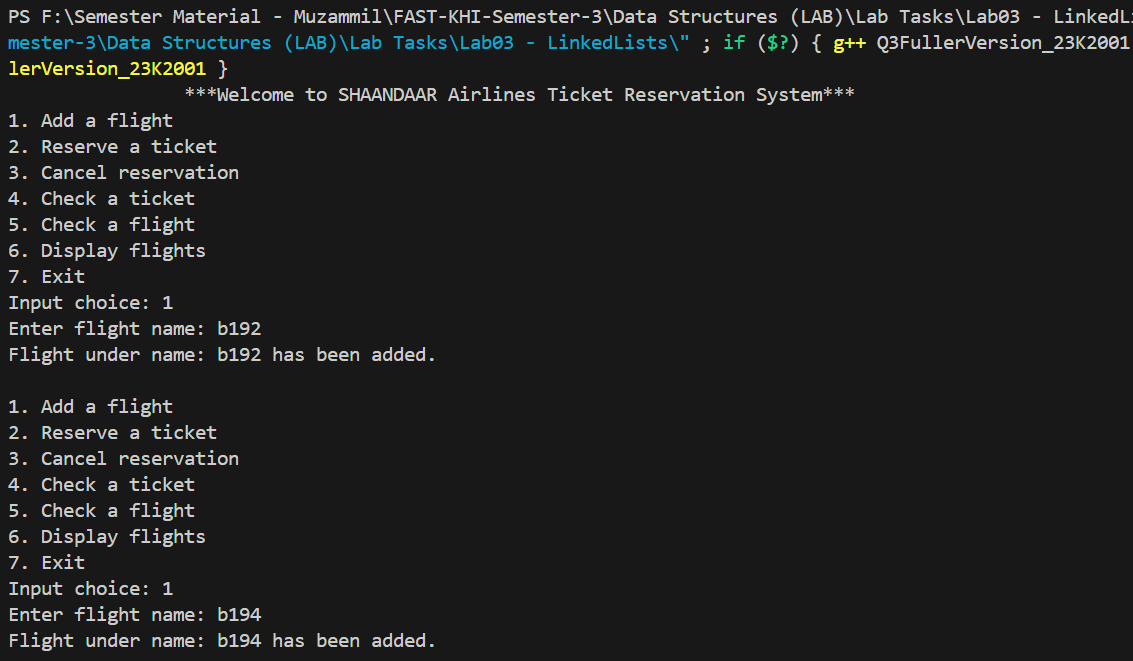
        }

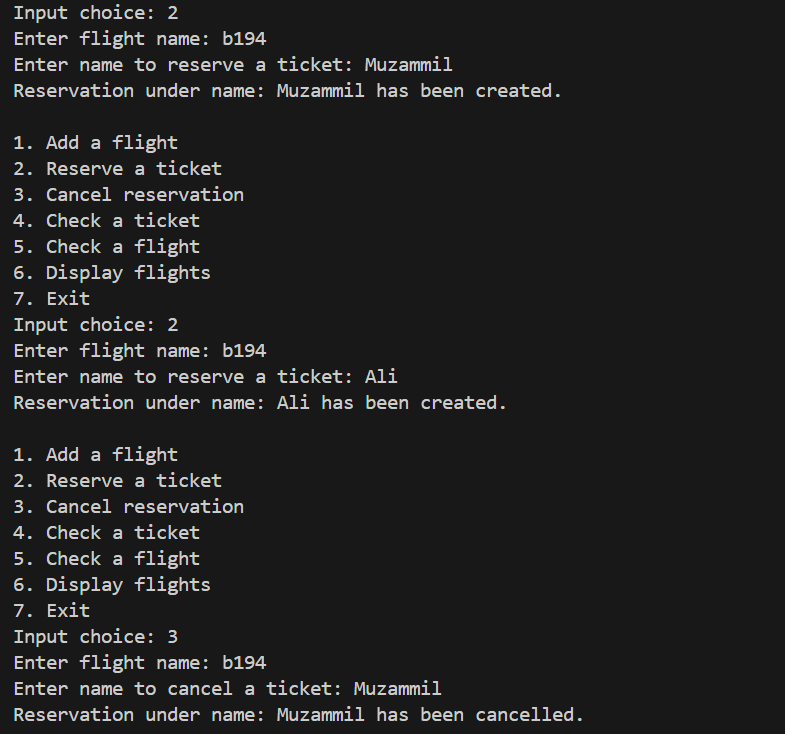
        cout<<endl;

    } while(c != 7);

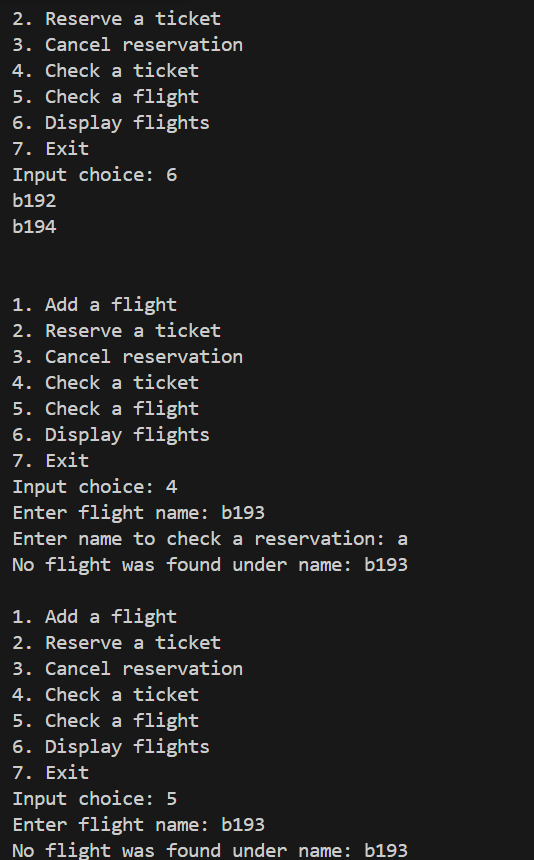
    return 0;

}





A screen shot of a computer program

Description automatically generated

Q4:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

        void evenFirst(){

            node\* temp = head;

            singleList evens;

            singleList odds;

            while(temp!=nullptr){

                if(temp->getData()%2==0)

                    evens.insertAtEnd(temp->getData());

                else

                    odds.insertAtEnd(temp->getData());

                temp = temp->getNext();

            }

            temp = head;

            while(temp!=nullptr){

                node\* next = temp->getNext();

                delete temp;

                temp = next;

            }

            head = nullptr;

            if(evens.head==nullptr){

                cout<<"No even elements were found!"<<endl;

                head = odds.head;

            }

            else{

                head = evens.head;

                evens.tail->setNext(odds.head);

            }

            if(odds.tail==nullptr){

                cout<<"No odd elements were found!"<<endl;

                tail = evens.tail;

            }

            else

                tail = odds.tail;

        }

};

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

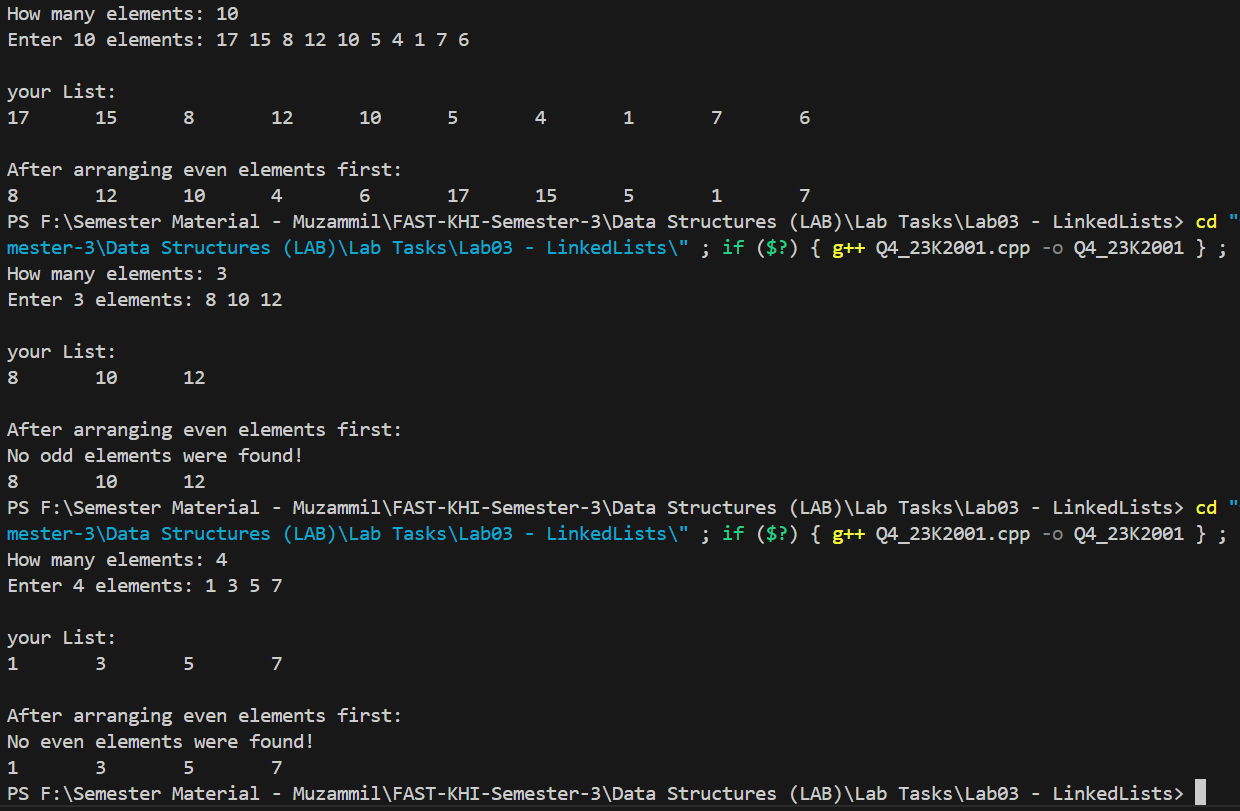
    cout<<endl<<"After arranging even elements first:"<<endl;

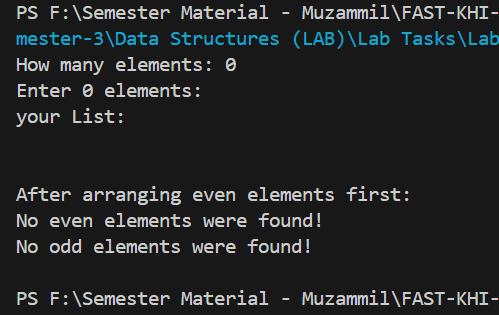
    flex.evenFirst();

    flex.display();

    return 0;

}





Q5:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        char data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(char val){

            data = val;

            next = nullptr;

        }

        char getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(char val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(char val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void checkPalindrome(){

            if(head==nullptr || head->getNext()==nullptr){

                cout<<"This list is a palindrome."<<endl;

                return;

            }

            node\* one=head;

            singleList reversedList;

            while(one!=nullptr){

                reversedList.insertAtEnd(one->getData());

                one=one->getNext();

            }

            node\* prev=nullptr;

            node\* current=reversedList.head;

            node\* next=nullptr;

            while(current!=nullptr){

                next=current->getNext();

                current->setNext(prev);

                prev=current;

                current=next;

            }

            reversedList.head=prev;

            one=head;

            while(one!=nullptr){

                if(one->getData()!=reversedList.head->getData()){

                cout<<"This list is NOT a palindrome."<<endl;

                return;

                }

                one=one->getNext();

                reversedList.head=reversedList.head->getNext();

            }

            cout<<"This list is a palindrome."<<endl;

        }

};

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e;

    char v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

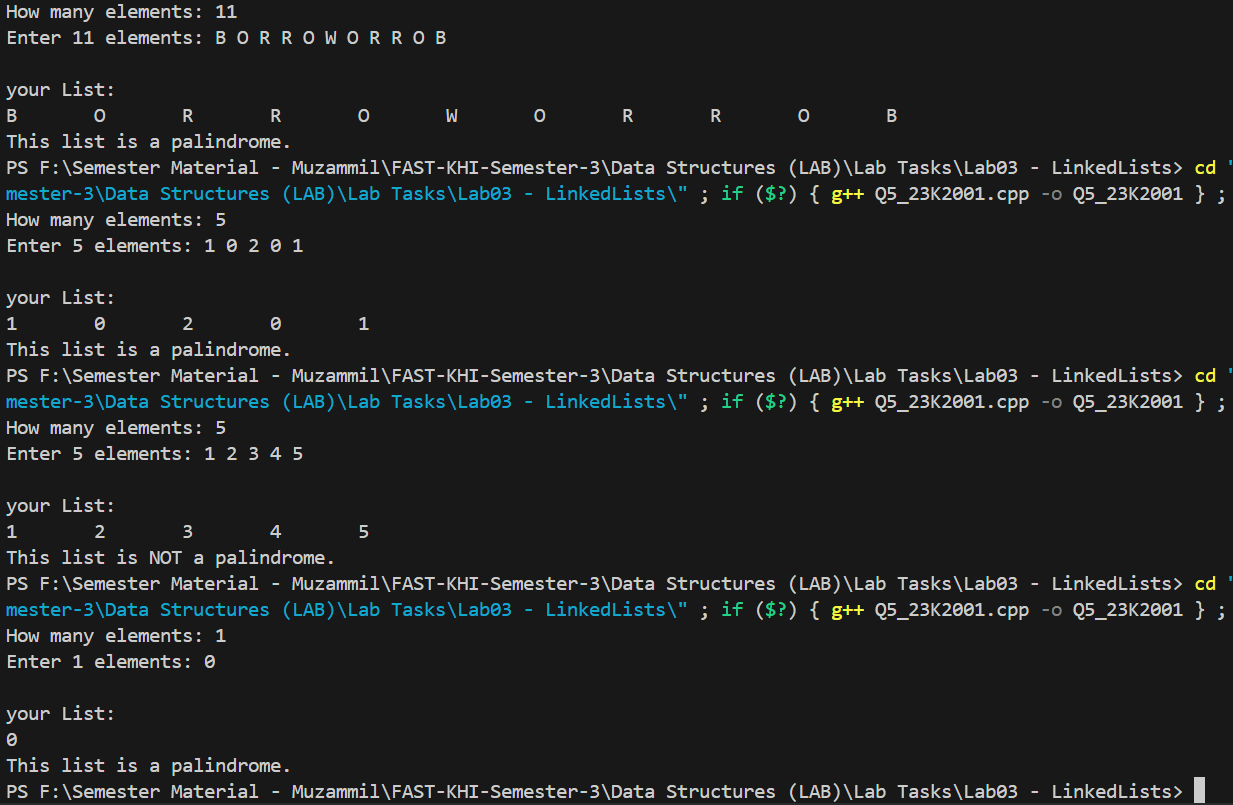
    cout<<endl<<"your List:"<<endl;

    flex.display();

    flex.checkPalindrome();

    return 0;

}



Q6:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp =  head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = tail->getNext();

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            if(head==nullptr){

            cout<<"The list is empty."<<endl;

            return;

            }

            if (head->getData()==val){

            node\* temp = head;

            head = head->getNext();

            delete temp;

            return;

            }

        node\* before = nullptr;

        node\* temp = head;

        while(temp!=nullptr && temp->getData()!=val){

            before = temp;

            temp = temp->getNext();

            }

        if(temp==nullptr){

            cout<<val<<" was not found in this list."<<endl;

            return;

            }

        before->setNext(temp->getNext());

        delete temp;

    }

};

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<"Which element to delete? ";

    cin>>e;

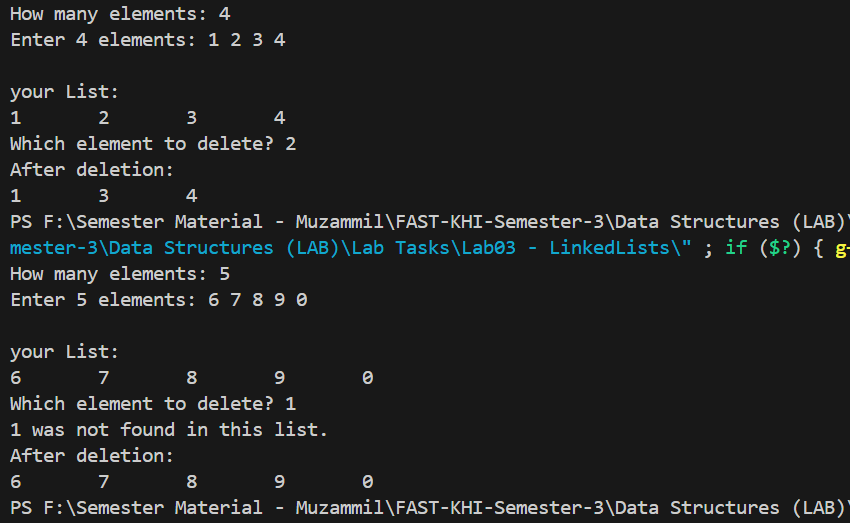
    flex.deleteNode(e);

    cout<<"After deletion:"<<endl;

    flex.display();

    return 0;

}



Q7:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class circularList{

    private:

        node\* head;

        node\* tail;

    public:

        circularList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            if(temp!=nullptr){

                do{

                    cout<<temp->getData()<<"\t";

                    temp=temp->getNext();

                } while(temp!=tail->getNext());

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            if(head==nullptr){

                head = n;

                tail = n;

                tail->setNext(head);

            }

            else{

                n->setNext(head);

                head = n;

                tail->setNext(head);

            }

        }

        void insertAtEnd(int val)

        {

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

                tail->setNext(head);

            }

            else{

                tail->setNext(n);

                tail = n;

                tail->setNext(head);

            }

        }

        void insertAtIndex(int index,int val){

            node\* update = new node(val);

            node\* temp = head;

            node\* before = nullptr;

            for(int i=0;i<index-1;i++){

                before = temp;

                temp=temp->getNext();

            }

            before->setNext(update);

            update->setNext(temp);

        }

        void deleteNode(int val){

            if(head==nullptr){

            cout<<"The list is empty."<<endl;

            return;

            }

            if(head->getData()==val){

                node\* temp = head;

                if(head->getNext()==head){

                    head = nullptr;

                    tail = nullptr;

                }

                else{

                    head = head->getNext();

                    tail->setNext(head);

                }

                delete temp;

                return;

            }

        node\* before = nullptr;

        node\* temp = head;

        while(temp->getNext()!=head && temp->getData()!=val){

            before = temp;

            temp = temp->getNext();

            }

        if(temp->getNext()==head && temp->getData()!=val){

            cout<<val<<" was not found in this list."<<endl;

            return;

            }

        before->setNext(temp->getNext());

        if(temp==tail)

            tail = before;

        delete temp;

        }

};

int main(){

    circularList flex;

    cout<<"How many elements: ";

    int e,v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

    flex.display();

    cout<<endl<<"Adding 9 in end:"<<endl;

    flex.insertAtEnd(9);

    flex.display();

    cout<<endl<<"Adding 11 at pos 3:"<<endl;

    flex.insertAtIndex(3,11);

    flex.display();

    cout<<endl<<"Adding 4 at start:"<<endl;

    flex.insertAtStart(4);

    flex.display();

    cout<<endl<<"Deleting 1,2 & 9:"<<endl;

    flex.deleteNode(1);

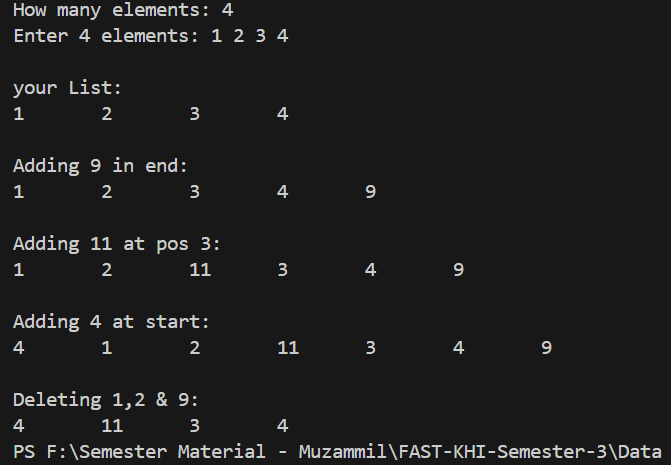
    flex.deleteNode(2);

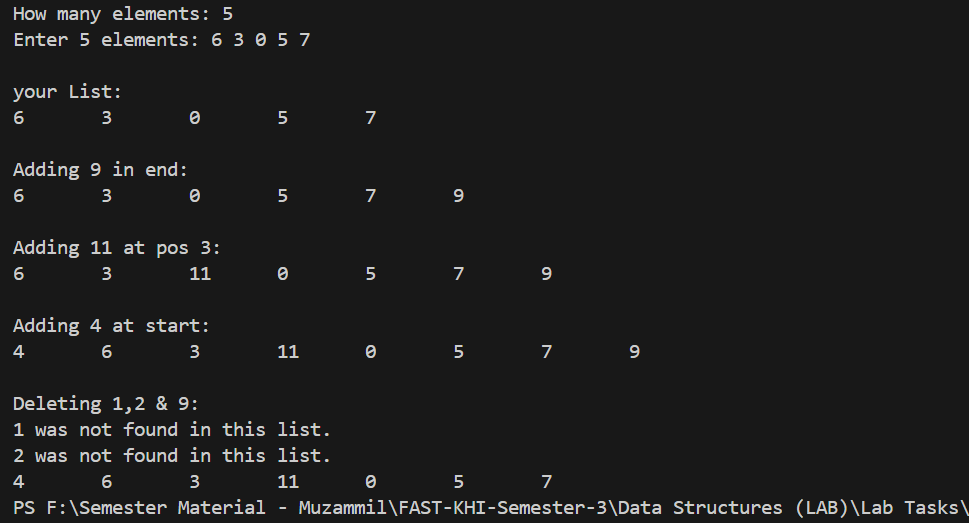
    flex.deleteNode(9);

    flex.display();

    return 0;

}





Q8:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

        node\* prev;

    public:

        node(){next = nullptr;

        prev = nullptr; }

        node(int val){

            data = val;

            next = nullptr;

            prev = nullptr;

        }

        int getData(){ return data; }

        node\* getNext(){return next;}

        node\* getPrev(){return prev;}

        void setNext(node\* update){next = update;}

        void setPrev(node\* update){prev = update;}

};

class doubleList{

    private:

        node\* head;

        node\* tail;

    public:

        doubleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        node\* getHead(){ return head; }

        node\* getTail(){ return tail; }

        void setHead(node\* update){head = update;}

        void setTail(node\* update){tail = update;}

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head->setPrev(n);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == NULL){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                n->setPrev(tail);

                tail = n;

            }

        }

        friend void concatenate(doubleList &l,doubleList &m);

};

void concatenate(doubleList &l,doubleList &m){

    if(l.getHead()==nullptr || m.getHead()==nullptr){

        cout<<"One of the provided list was empty."<<endl;

        return;

    }

    l.getTail()->setNext(m.getHead());

    m.getHead()->setPrev(l.getTail());

    l.setTail(m.getTail());

}

int main(){

    doubleList flex1,flex2;

    cout<<"How many elements for list#1: ";

    int e;

    int v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex1.insertAtEnd(v);

    }

    cout<<"How many elements for list#2: ";

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex2.insertAtEnd(v);

    }

    cout<<endl<<"your List#1:"<<endl;

    flex1.display();

    cout<<endl<<"your List#2:"<<endl;

    flex2.display();

    cout<<endl<<"After concatenation:"<<endl;

    concatenate(flex1,flex2);

    cout<<"List 1: ";

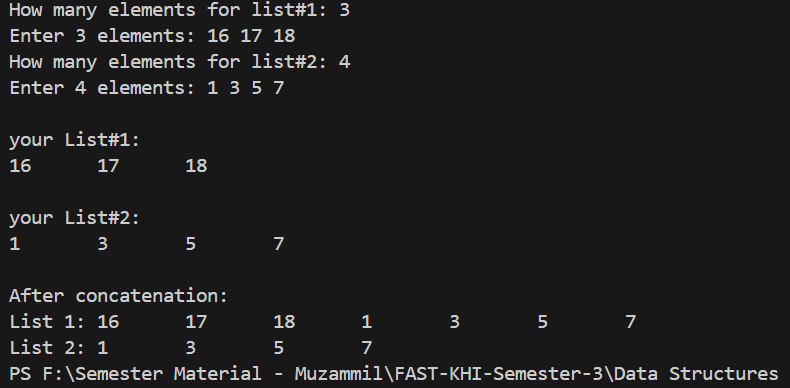
    flex1.display();

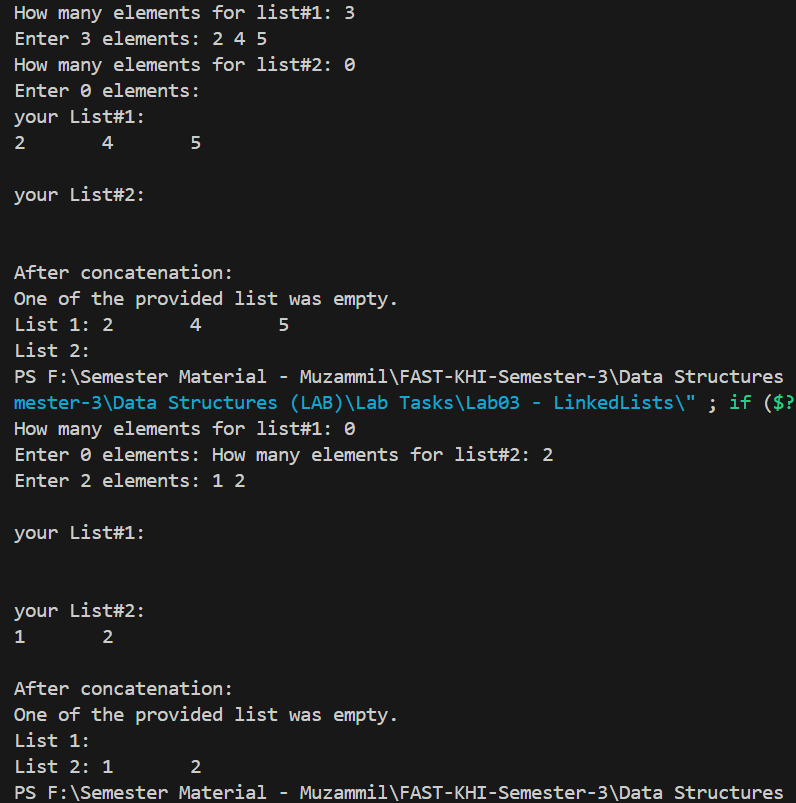
    cout<<"List 2: ";

    flex2.display();

    return 0;

}





Q9:

//23K2001 Muzammil

#include<iostream>

using namespace std;

class node{

    private:

        int data;

        node\* next;

    public:

        node(){next = nullptr;}

        node(int val){

            data = val;

            next = nullptr;

        }

        int getData(){ return data;}

        node\* getNext(){return next;}

        void setNext(node\* update){next = update;}

};

class singleList{

    private:

        node\* head;

        node\* tail;

    public:

        singleList(){

            head = nullptr;

            tail = nullptr;

        }

        void display(){

            node\* temp = head;

            while(temp!=nullptr)

            {

                cout<<temp->getData()<<"\t";

                temp=temp->getNext();

            }

            cout<<endl;

        }

        void insertAtStart(int val)

        {

            node\* n = new node(val);

            n->setNext(head);

            head = n;

        }

        void insertAtEnd(int val)

        {

            node\* temp = head;

            node\* n = new node(val);

            if(head == nullptr){

                head = n;

                tail = n;

            }

            else{

                tail->setNext(n);

                tail = n;

            }

        }

        void deletenode(int val){

            node\* before = nullptr;

            node\* temp = head;

            while(temp->getData()!=val){

                before = temp;

                temp = temp->getNext();

            }

            before->setNext(temp->getNext());

            delete temp;

        }

        void question9(){

            if(head==nullptr || head->getNext()==nullptr || head->getNext()->getNext()==nullptr){

                cout<<"Not enough nodes in the list."<<endl;

                return;

            }

            node\* ones = head;

            node\* twos = ones->getNext();

            ones->setNext(ones->getNext()->getNext());

            ones = ones->getNext();

            twos->setNext(nullptr);

            while(ones->getNext()!=nullptr){

                node\* temp = ones->getNext()->getNext();

                ones->getNext()->setNext(twos);

                twos = ones->getNext();

                ones->setNext(temp);

                if(temp!=nullptr)

                    ones = temp;

            }

            ones->setNext(twos);

        }

};

int main(){

    singleList flex;

    cout<<"How many elements: ";

    int e;

    int v;

    cin>>e;

    cout<<"Enter "<<e<<" elements: ";

    for(int i=0;i<e;i++){

        cin>>v;

        flex.insertAtEnd(v);

    }

    cout<<endl<<"your List:"<<endl;

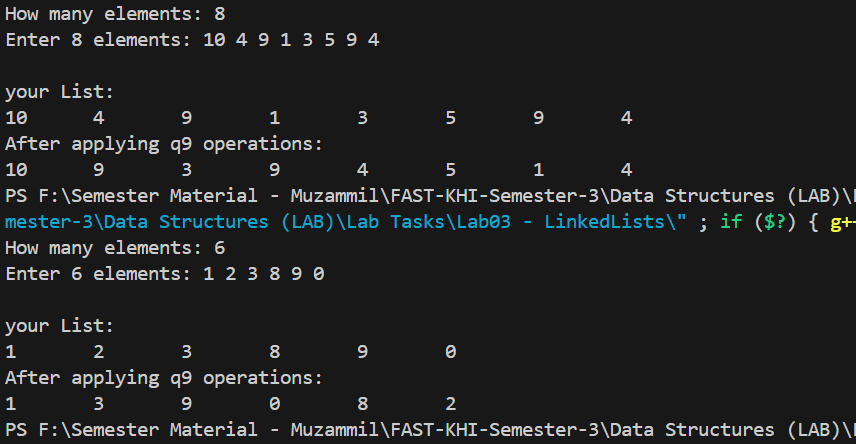
    flex.display();

    cout<<"After applying q9 operations:"<<endl;

    flex.question9();

    flex.display();

}



A screen shot of a computer

Description automatically generated