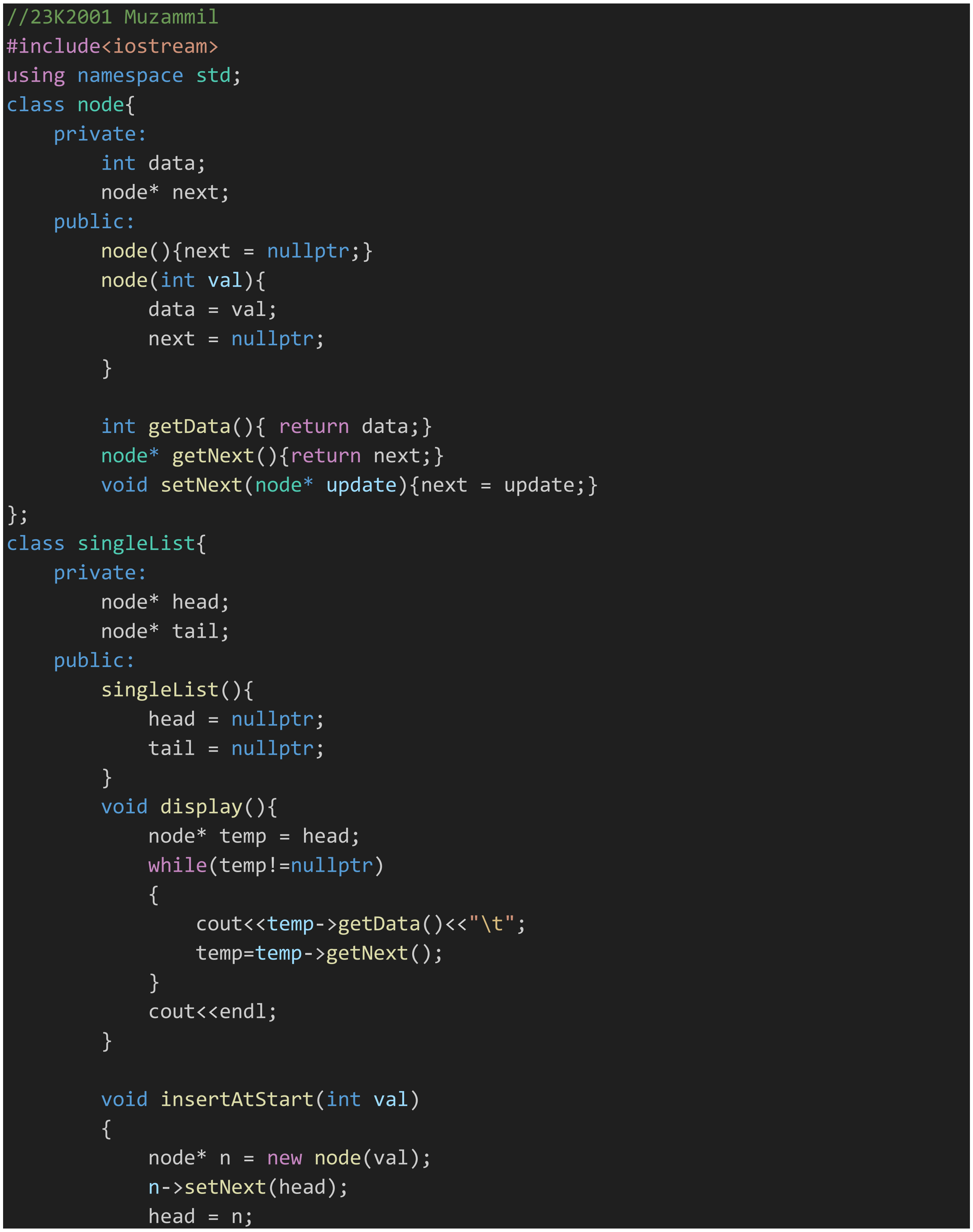
DSA Lab03

23K2001

M.Muzammil Siddiqui

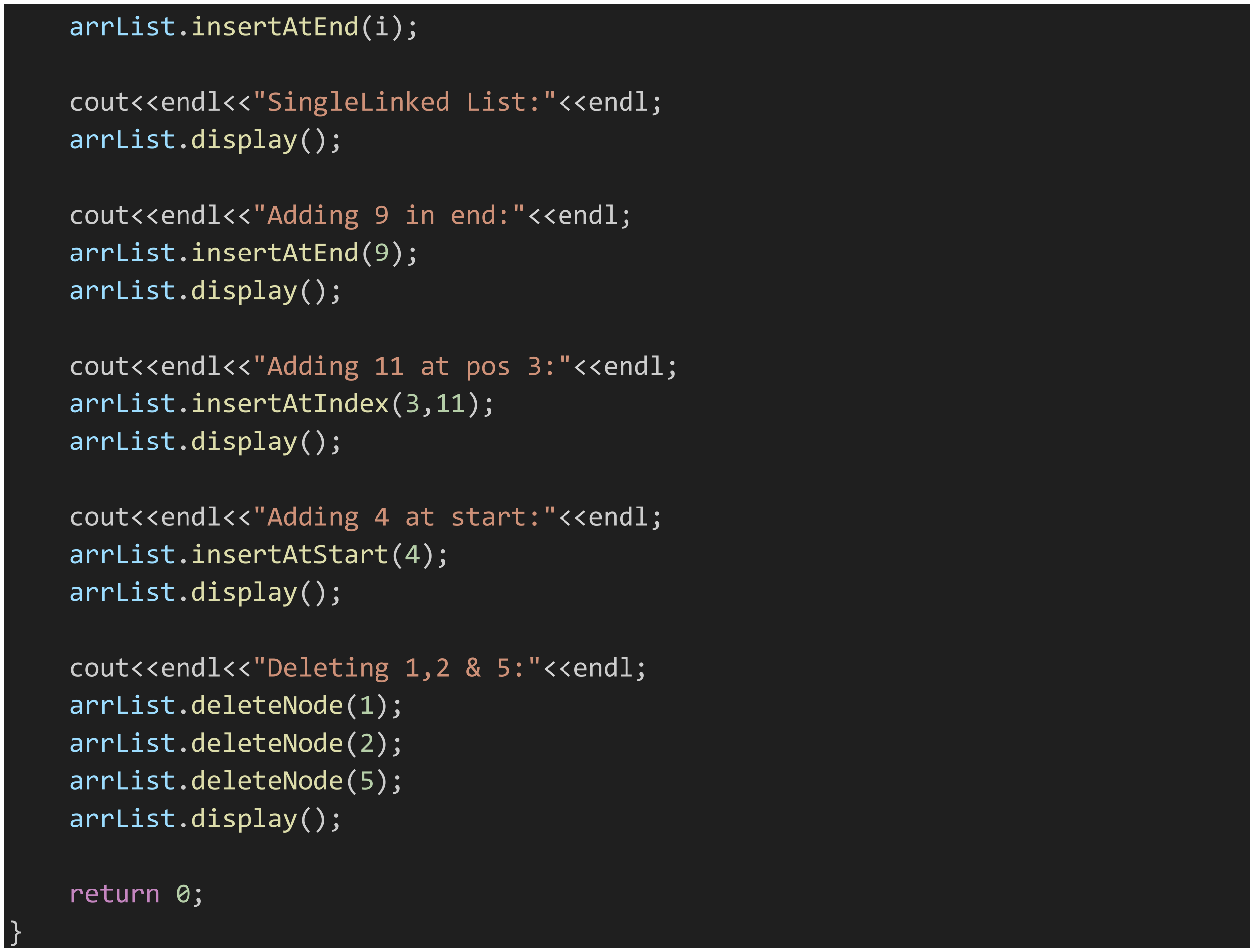
BCS-3J

Q1:

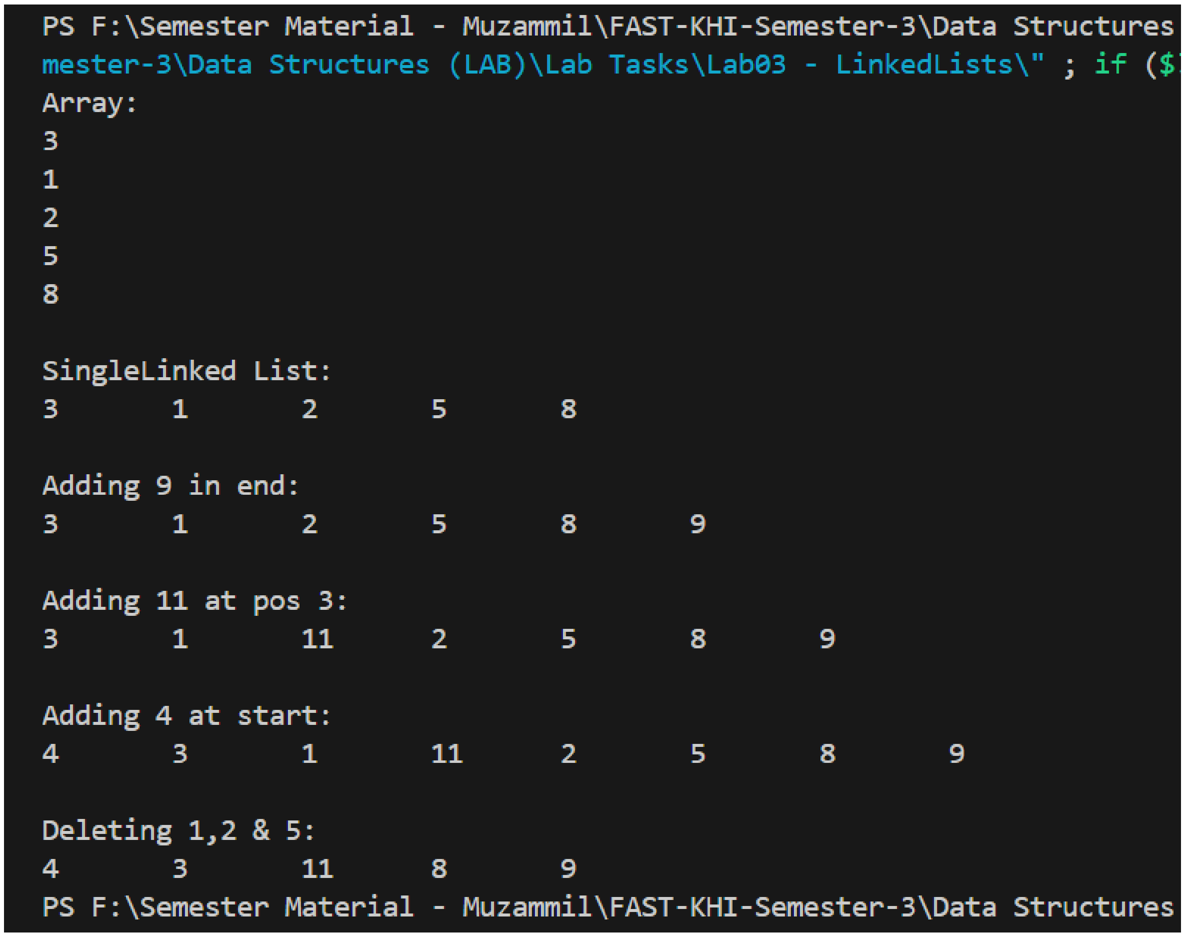


```
//23K2001 Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        int data;
        node* next;
    public:
        node(){next = nullptr;}
        node(int val){
            data = val;
            next = nullptr;
        }

        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;
```

```cpp
        }
        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)
```

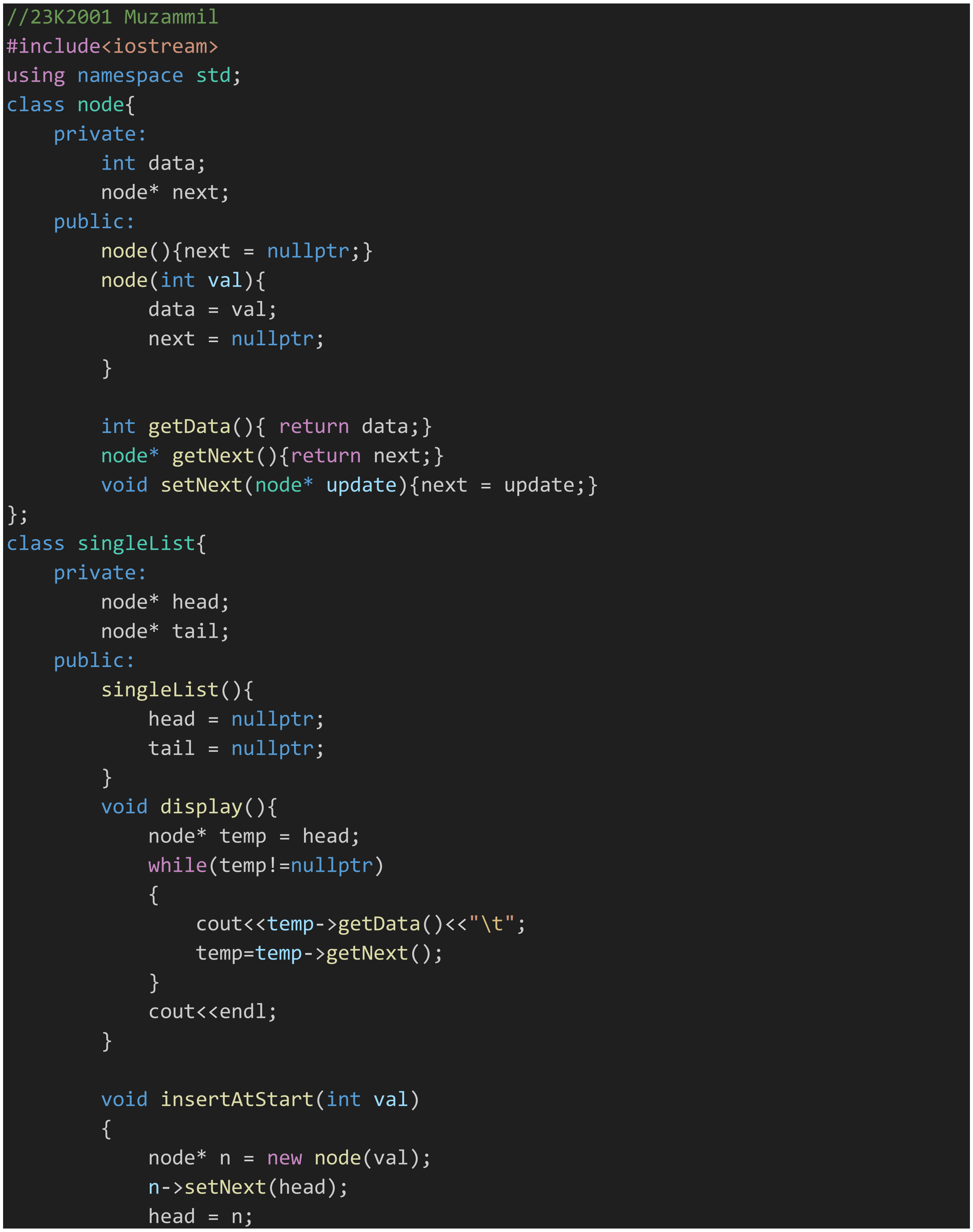


```cpp
    arrList.insertAtEnd(i);

    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;
}
```

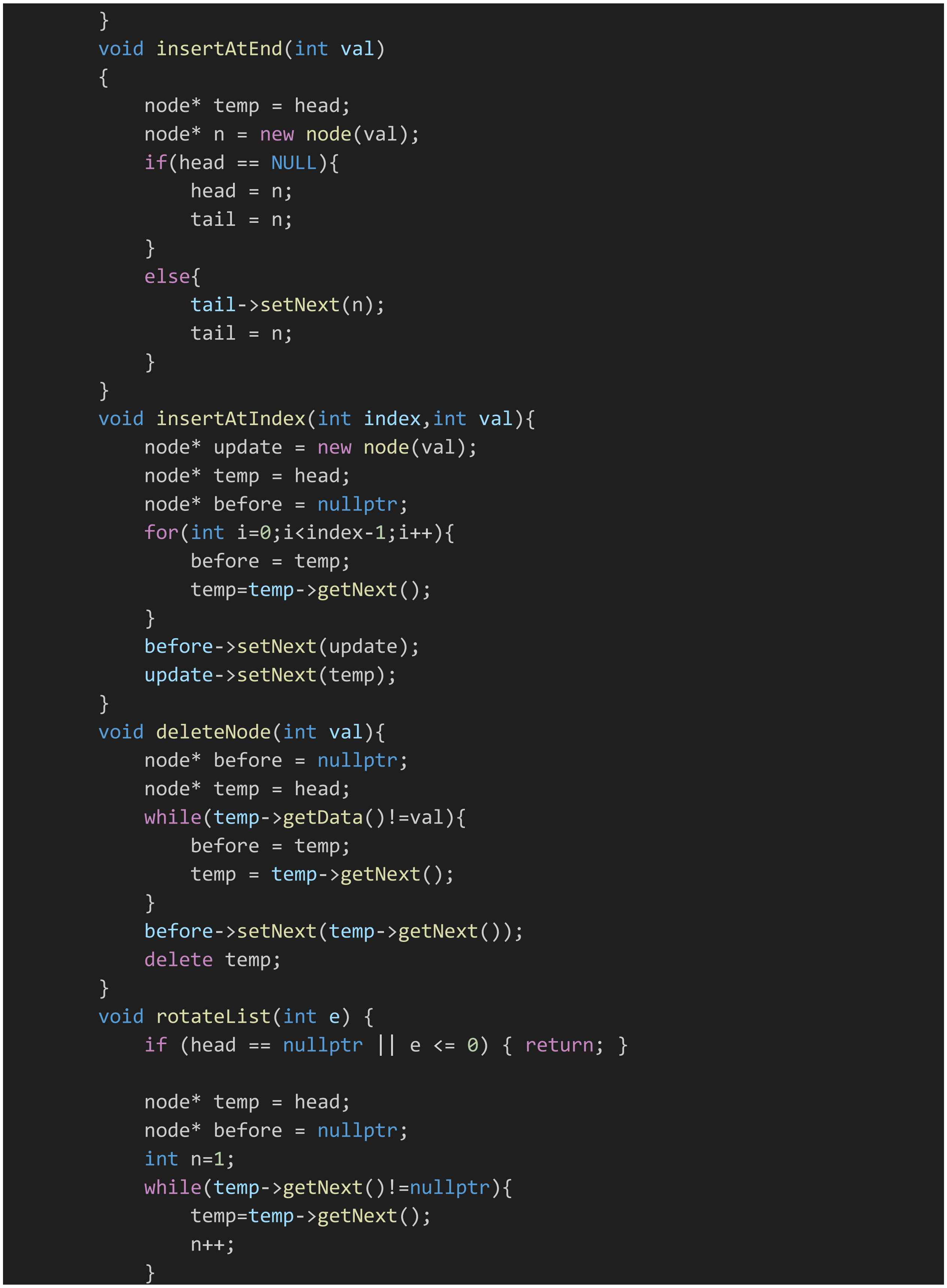


```
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($
Array:
3
1
2
5
8

SingleLinked List:
3       1       2       5       8

Adding 9 in end:
3       1       2       5       8       9

Adding 11 at pos 3:
3       1       11      2       5       8       9

Adding 4 at start:
4       3       1       11      2       5       8       9

Deleting 1,2 & 5:
4       3       11      8       9
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```

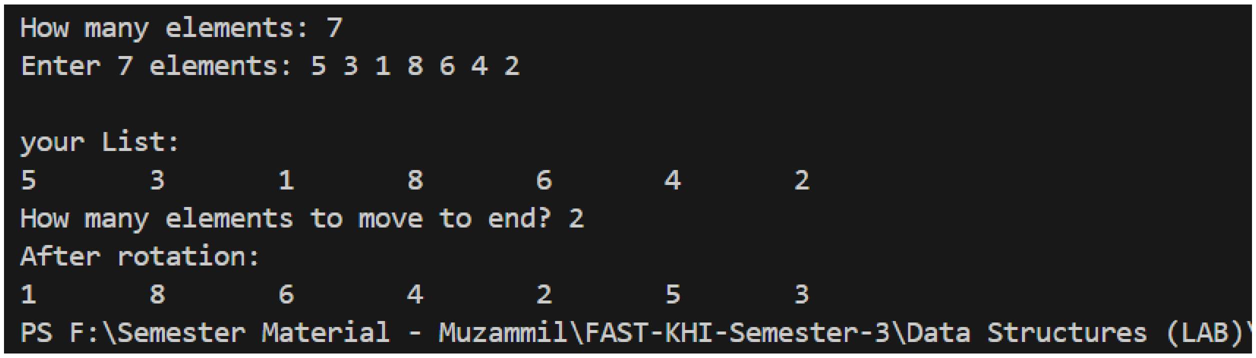
Q2:



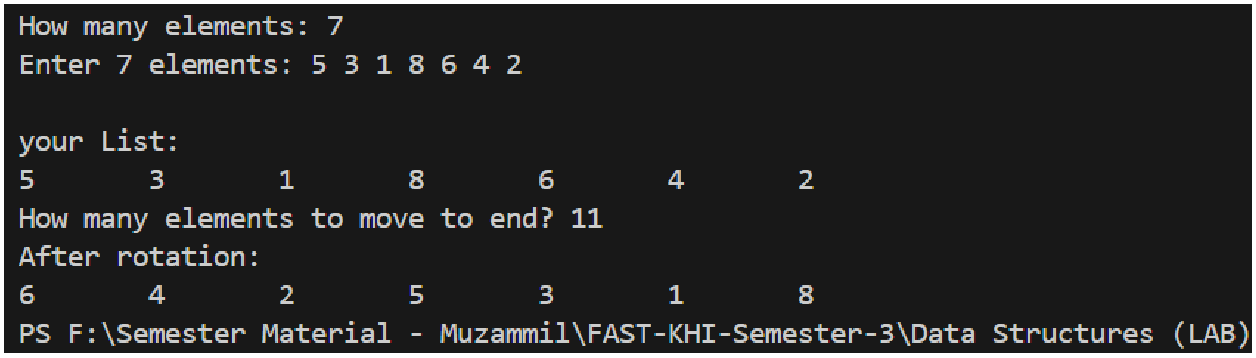
```
//23K2001 Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        int data;
        node* next;
    public:
        node(){next = nullptr;}
        node(int val){
            data = val;
            next = nullptr;
        }

        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;
```



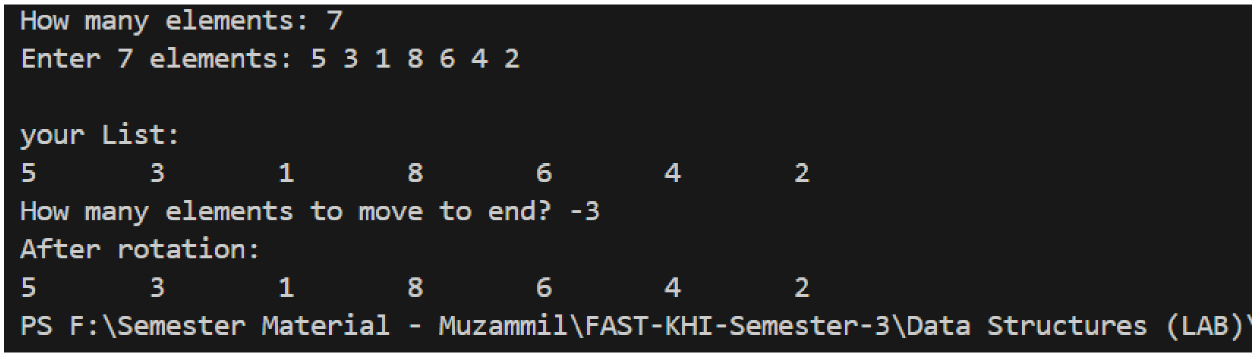
```cpp
        }
        void insertAtEnd(int val)
        {
            node* temp = head;
            node* n = new node(val);
            if(head == NULL){
                head = n;
                tail = n;
            }
            else{
                tail->setNext(n);
                tail = n;
            }
        }
        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;
}
```



```
How many elements: 7
Enter 7 elements: 5 3 1 8 6 4 2

your List:
5       3       1       8       6       4       2
How many elements to move to end? 2
After rotation:
1       8       6       4       2       5       3
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
```



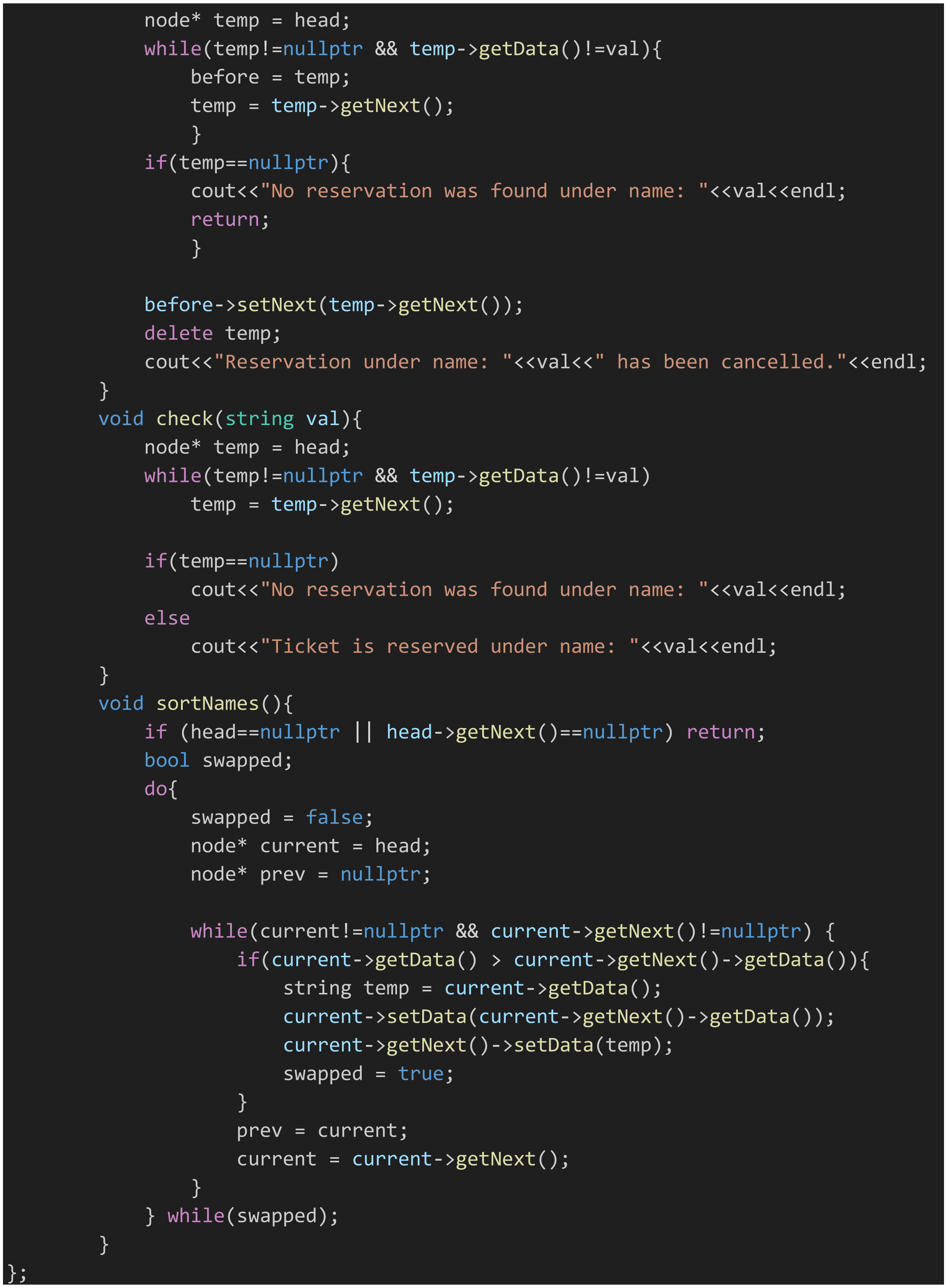
```
How many elements: 7
Enter 7 elements: 5 3 1 8 6 4 2

your List:
5       3       1       8       6       4       2
How many elements to move to end? 11
After rotation:
6       4       2       5       3       1       8
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
```



```
How many elements: 7
Enter 7 elements: 5 3 1 8 6 4 2

your List:
5       3       1       8       6       4       2
How many elements to move to end? -3
After rotation:
5       3       1       8       6       4       2
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
```

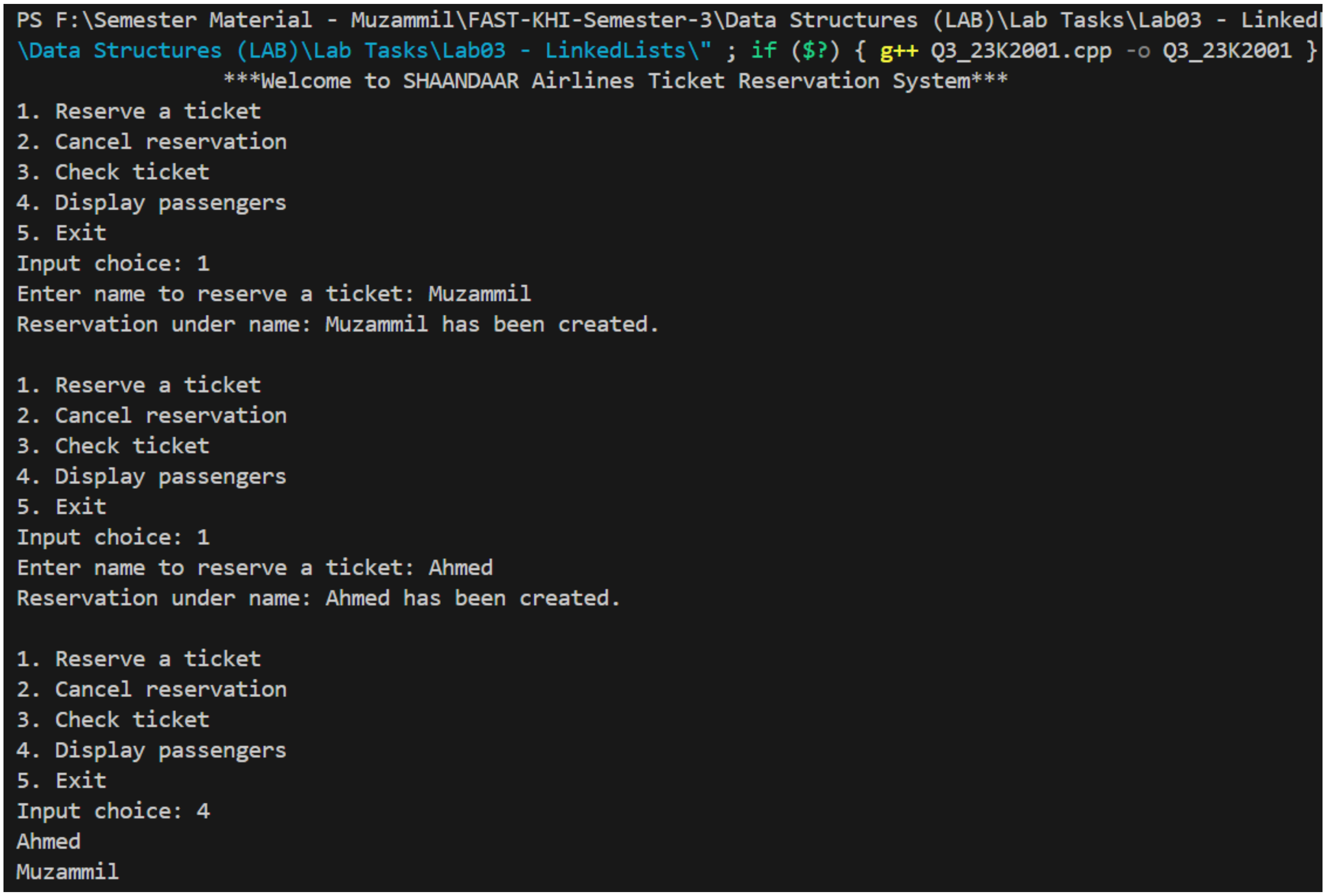
Q3:

```
//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;
```

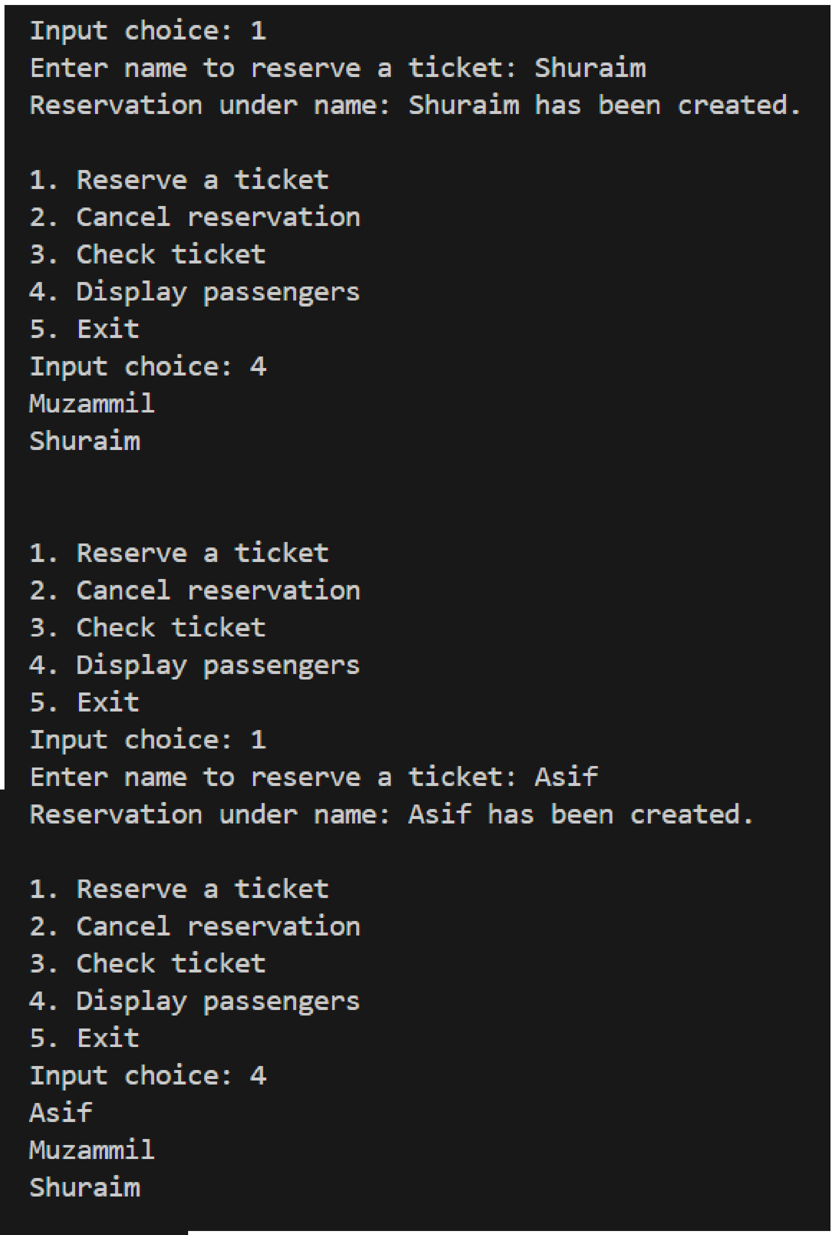


```
            node* temp = head;
            while(temp!=nullptr && temp->getData()!=val){
                before = temp;
                temp = temp->getNext();
                }
            if(temp==nullptr){
                cout<<"No reservation was found under name: "<<val<<endl;
                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);
        }
};
```

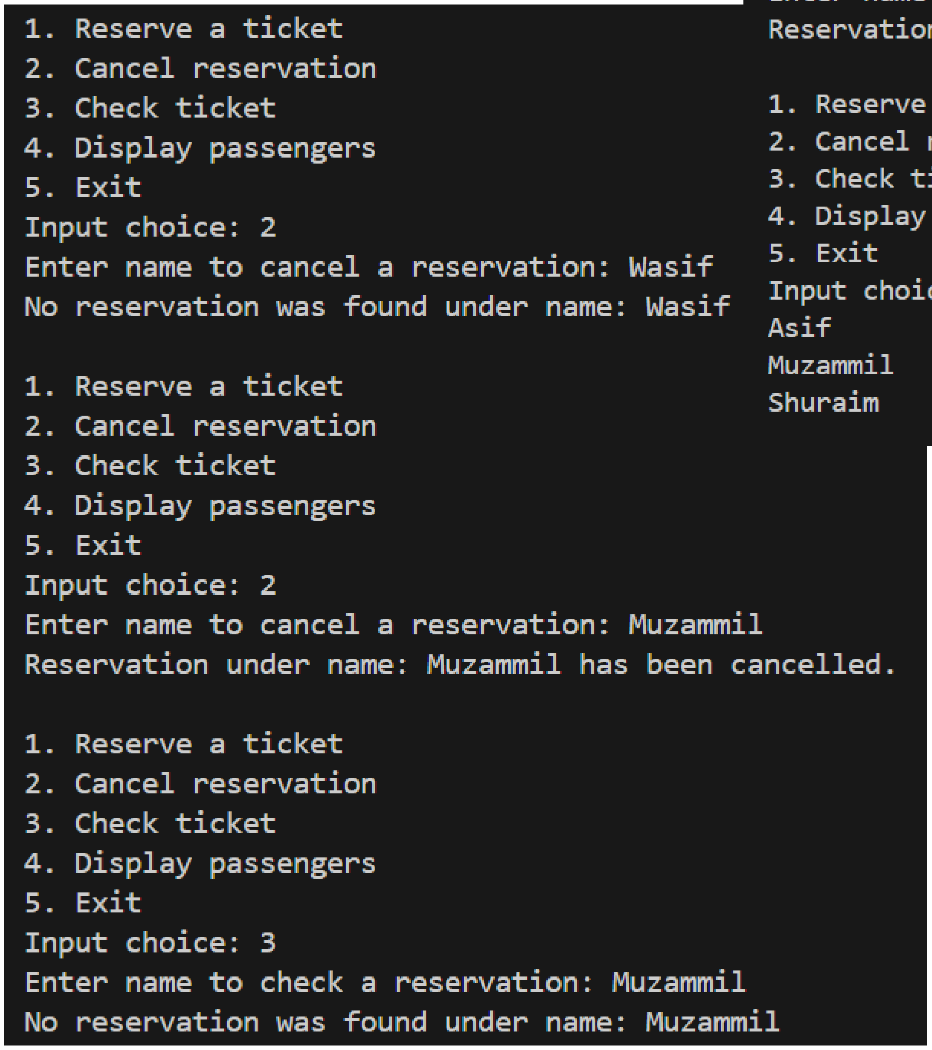
```
int main(){
    int c;
    string n;
    singleList passengers;
    cout<<"\t\t***Welcome to SHAANDAAR Airlines Ticket Reservation 
System***"<<endl;
    do{
        cout<<"1. Reserve a ticket"<<endl;
        cout<<"2. Cancel reservation"<<endl;
        cout<<"3. Check ticket"<<endl;
        cout<<"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;
}
```



```
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - Linked
\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q3_23K2001.cpp -o Q3_23K2001 }
                ***Welcome to SHAANDAAR Airlines Ticket Reservation System***
1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 1
Enter name to reserve a ticket: Muzammil
Reservation under name: Muzammil has been created.

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 1
Enter name to reserve a ticket: Ahmed
Reservation under name: Ahmed has been created.

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 4
Ahmed
Muzammil
```



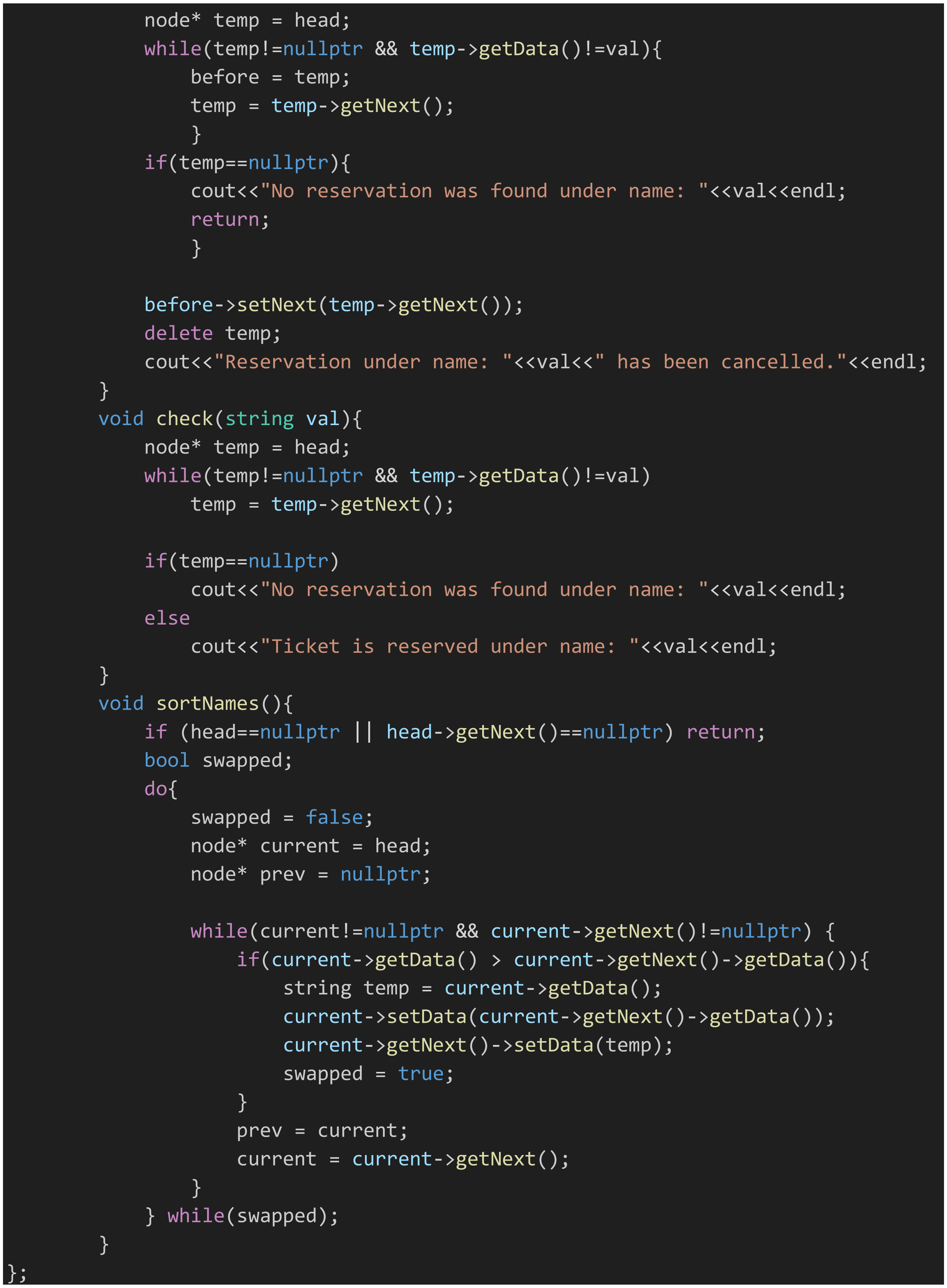
```
Input choice: 1
Enter name to reserve a ticket: Shuraim
Reservation under name: Shuraim has been created.

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 4
Muzammil
Shuraim


1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 1
Enter name to reserve a ticket: Asif
Reservation under name: Asif has been created.

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 4
Asif
Muzammil
Shuraim
```



```
1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 2
Enter name to cancel a reservation: Wasif
No reservation was found under name: Wasif

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 2
Enter name to cancel a reservation: Muzammil
Reservation under name: Muzammil has been cancelled.

1. Reserve a ticket
2. Cancel reservation
3. Check ticket
4. Display passengers
5. Exit
Input choice: 3
Enter name to check a reservation: Muzammil
No reservation was found under name: Muzammil
```

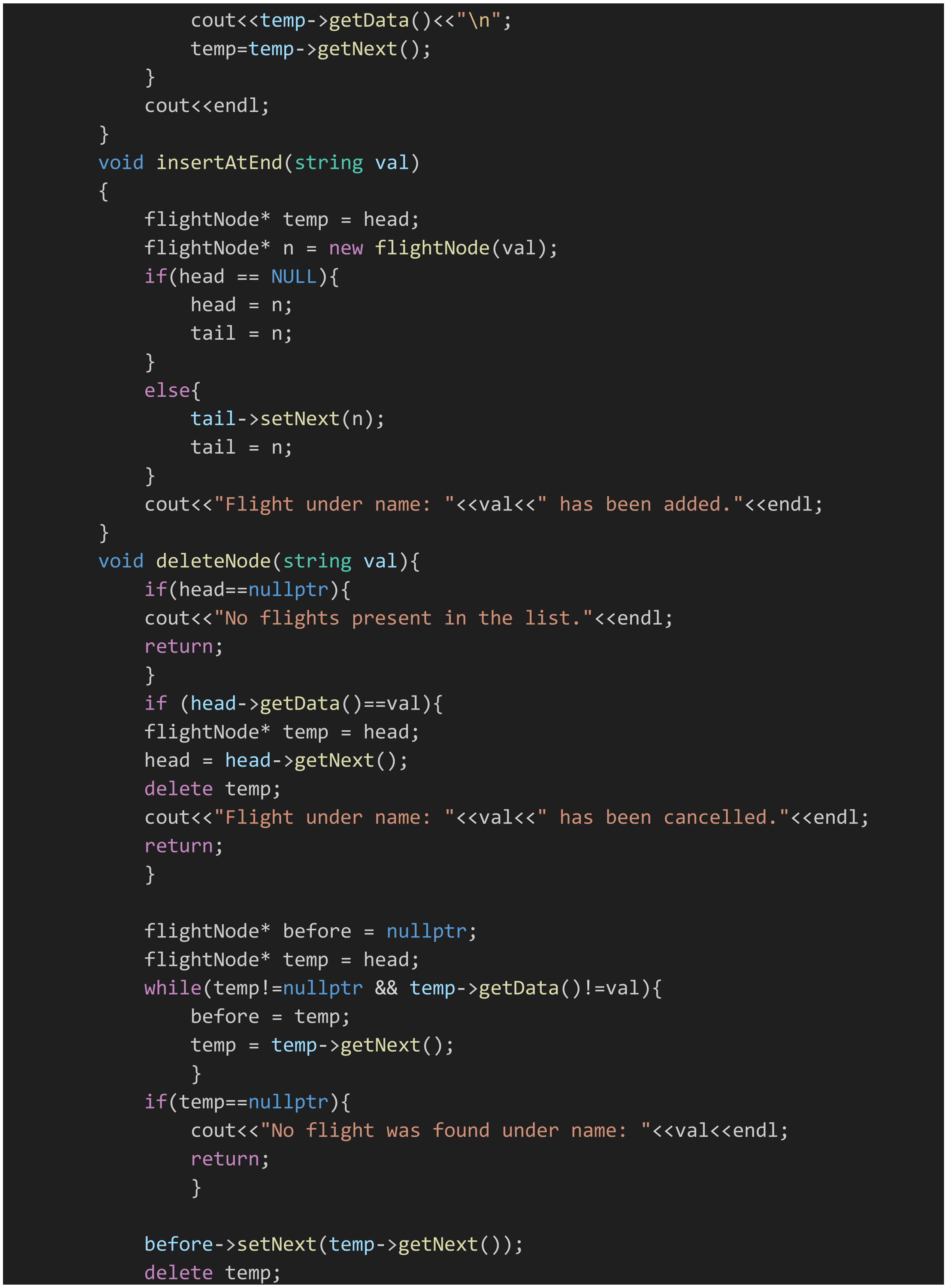
Q3: Full Version

```
//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;
```

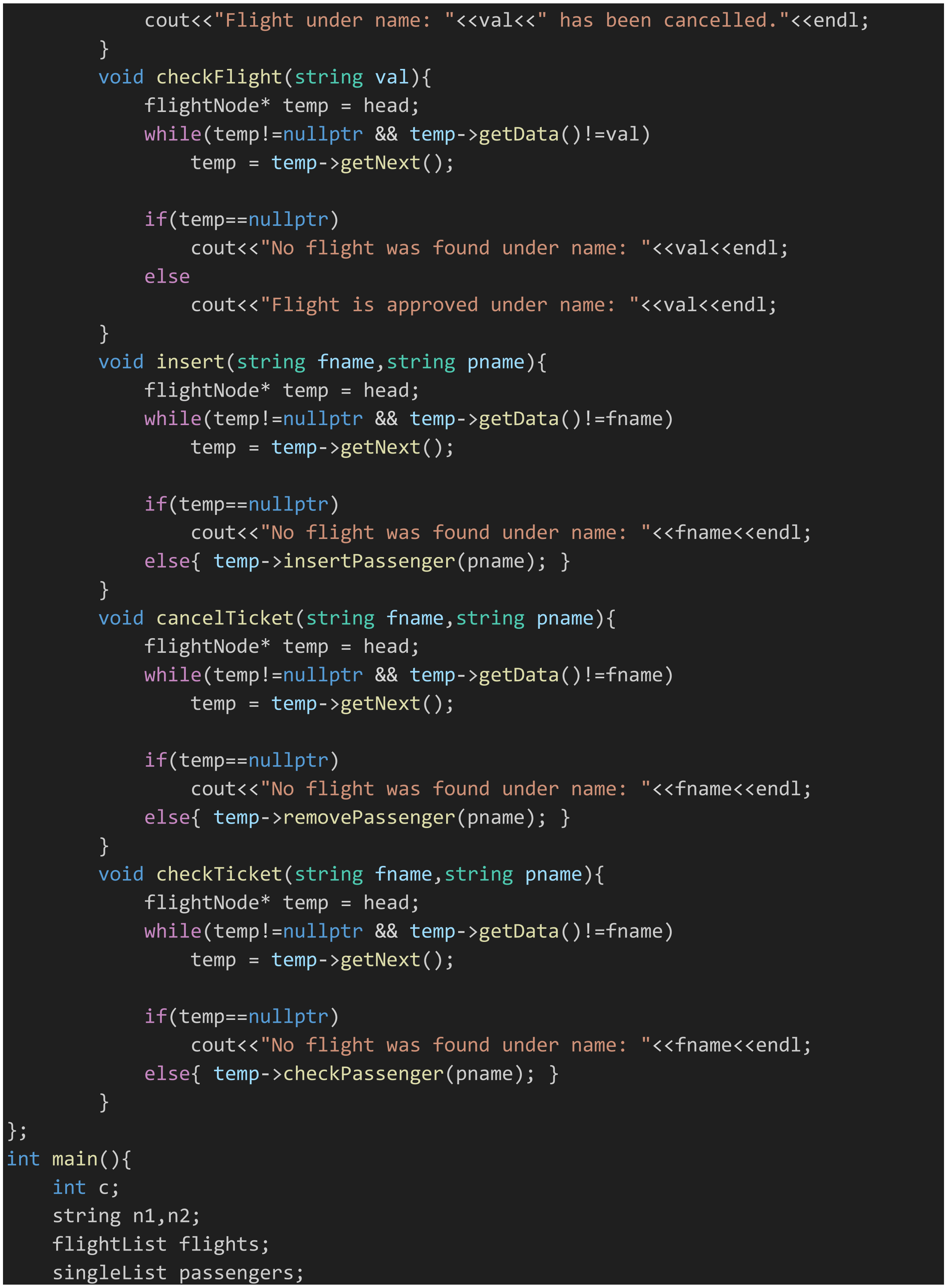


```cpp
            node* temp = head;
            while(temp!=nullptr && temp->getData()!=val){
                before = temp;
                temp = temp->getNext();
                }
            if(temp==nullptr){
                cout<<"No reservation was found under name: "<<val<<endl;
                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);
        }
};
```

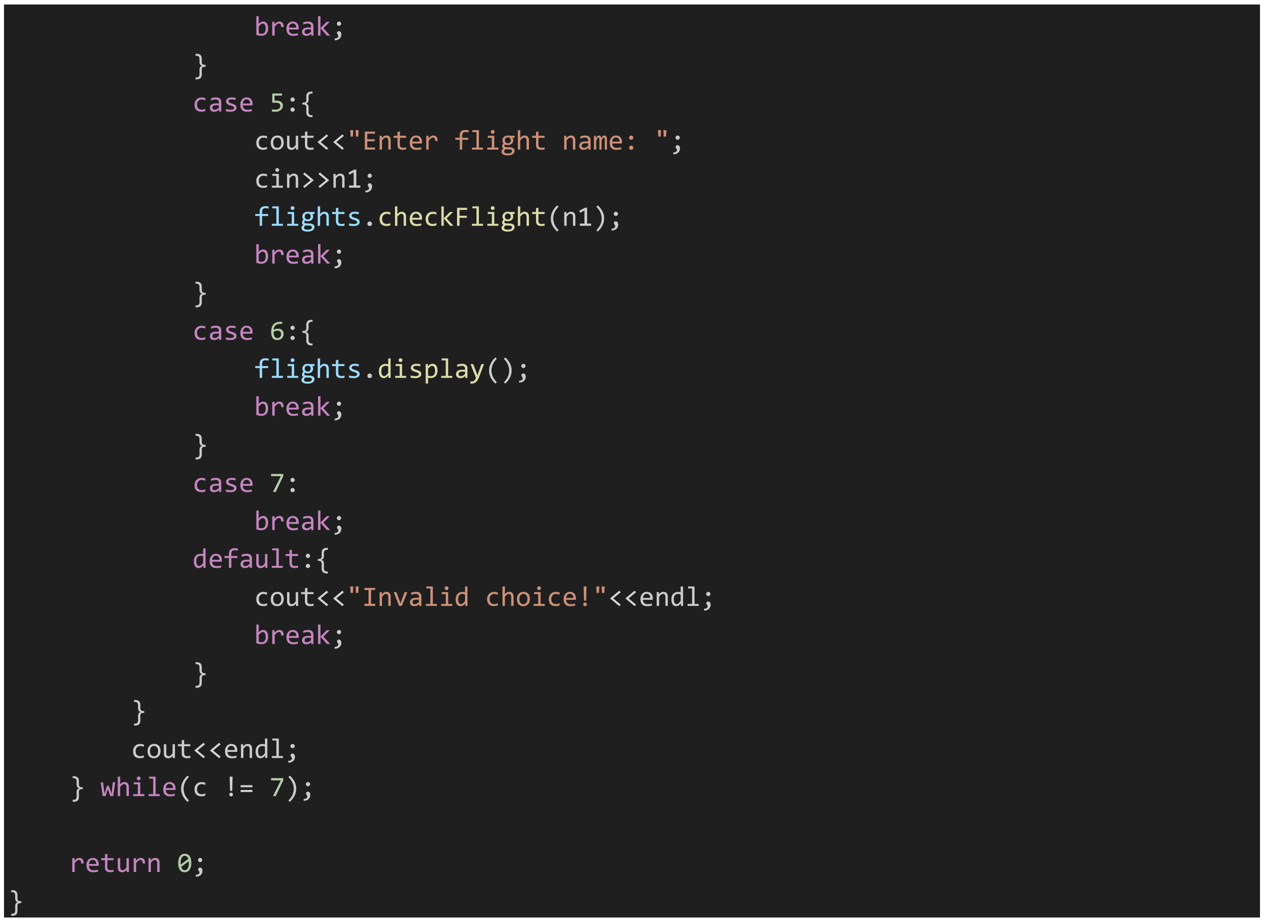
```
class flightNode{
    private:
        string name;
        flightNode* next;
        singleList* passengers;
    public:
        flightNode() : next(nullptr),passengers(nullptr){}
        flightNode(string val){
            name = val;
            next = nullptr;
            passengers = nullptr;
        }

        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)
            {
```



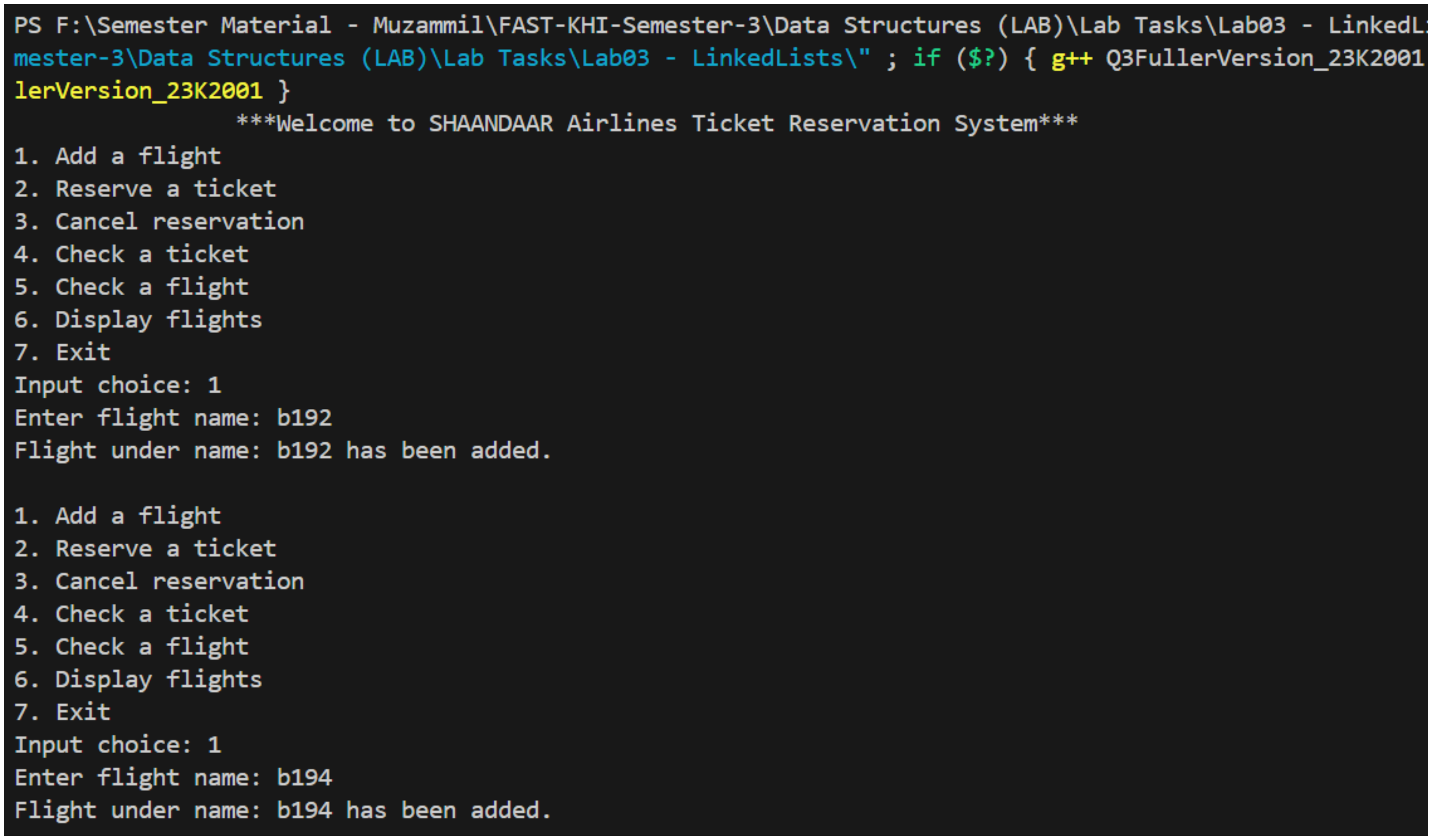
```
            cout<<temp->getData()<<"\n";
            temp=temp->getNext();
        }
        cout<<endl;
    }
    void insertAtEnd(string val)
    {
        flightNode* temp = head;
        flightNode* n = new flightNode(val);
        if(head == NULL){
            head = n;
            tail = n;
        }
        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;
```



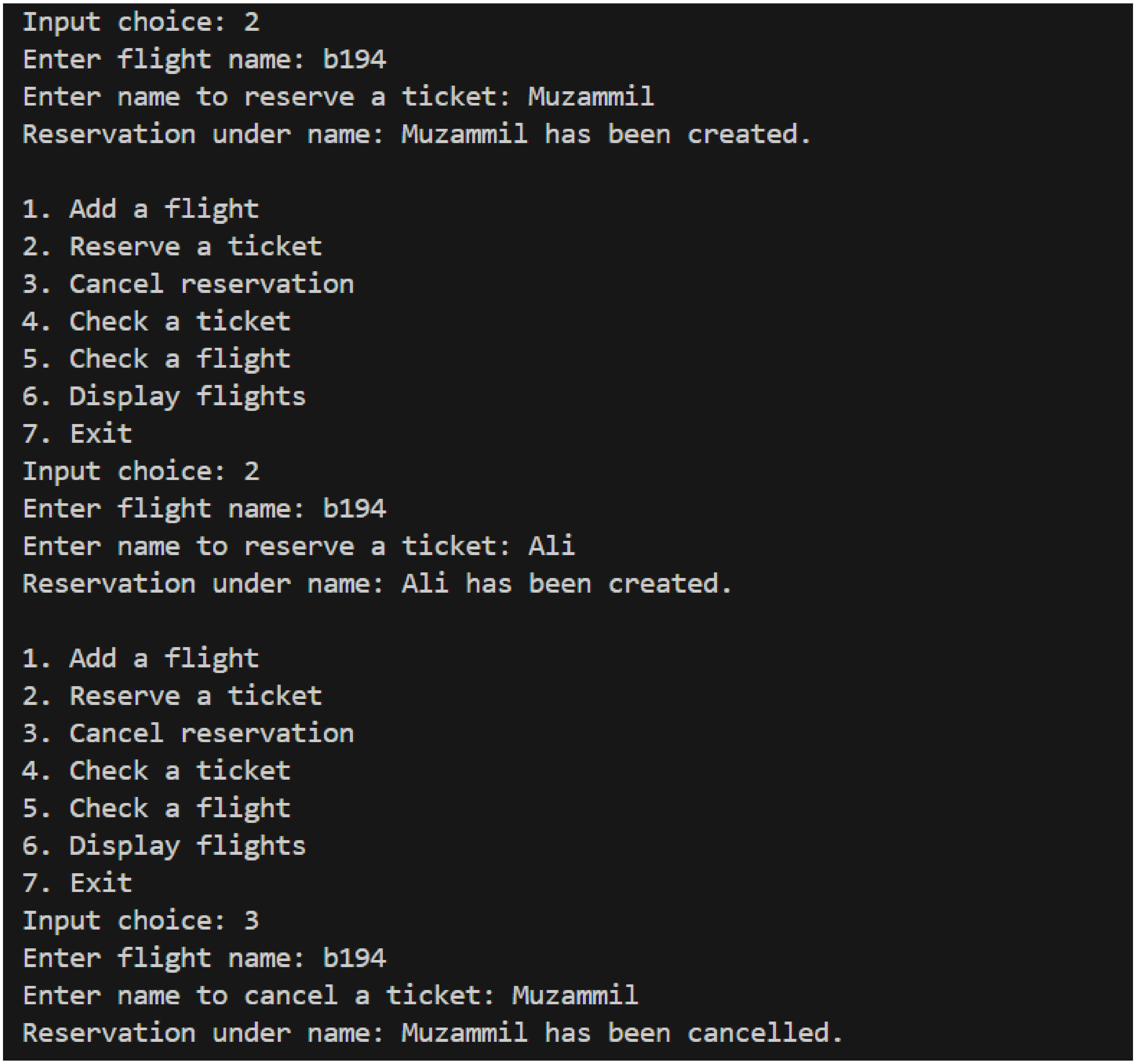
```
            cout<<"Flight under name: "<<val<<" has been cancelled."<<endl;
        }
        void checkFlight(string val){
            flightNode* temp = head;
            while(temp!=nullptr && temp->getData()!=val)
                temp = temp->getNext();

            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);
```



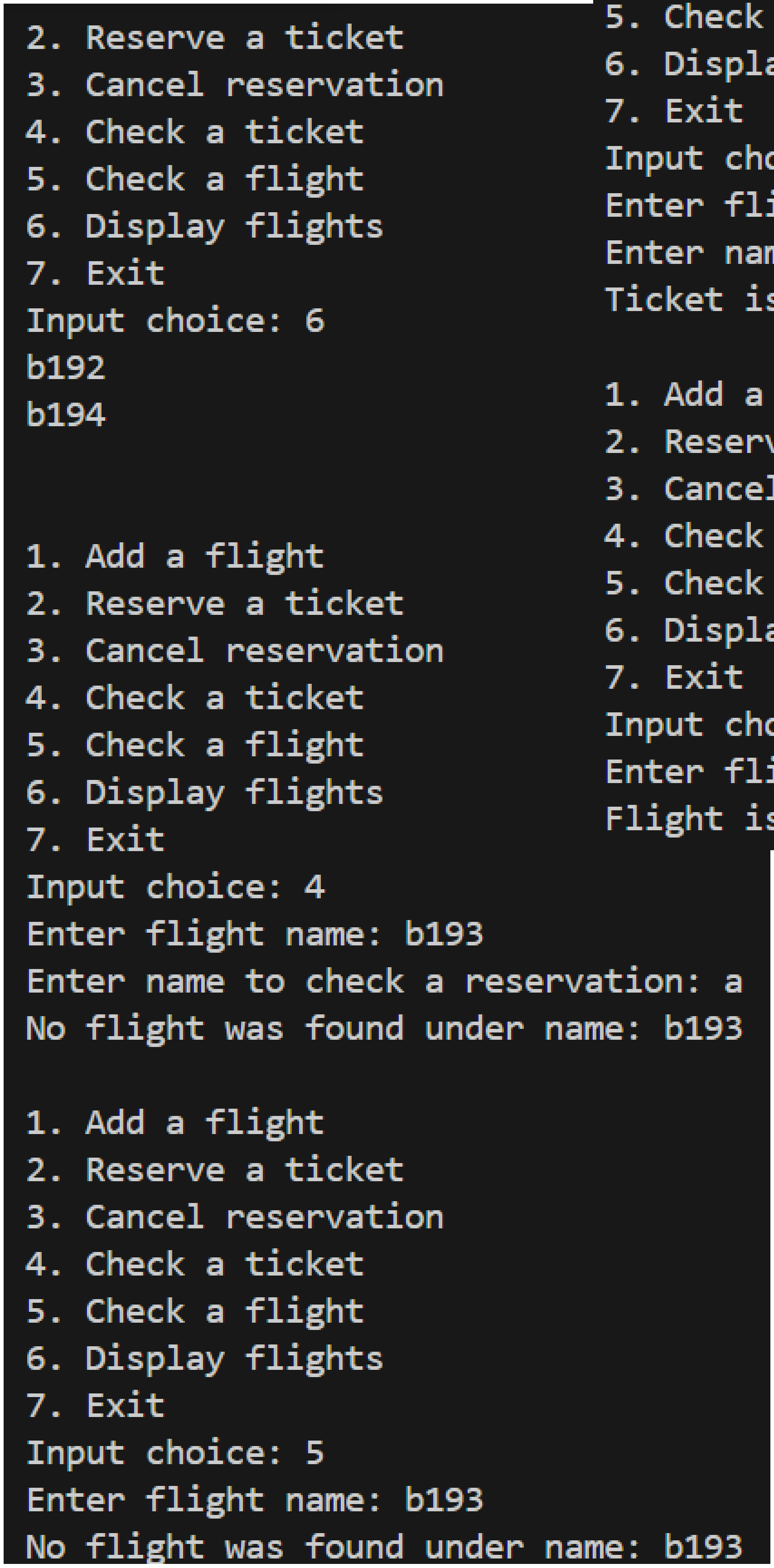
```cpp
                break;
            }
            case 5:{
                cout<<"Enter flight name: ";
                cin>>n1;
                flights.checkFlight(n1);
                break;
            }
            case 6:{
                flights.display();
                break;
            }
            case 7:
                break;
            default:{
                cout<<"Invalid choice!"<<endl;
                break;
            }
        }
        cout<<endl;
    } while(c != 7);

    return 0;
}
```



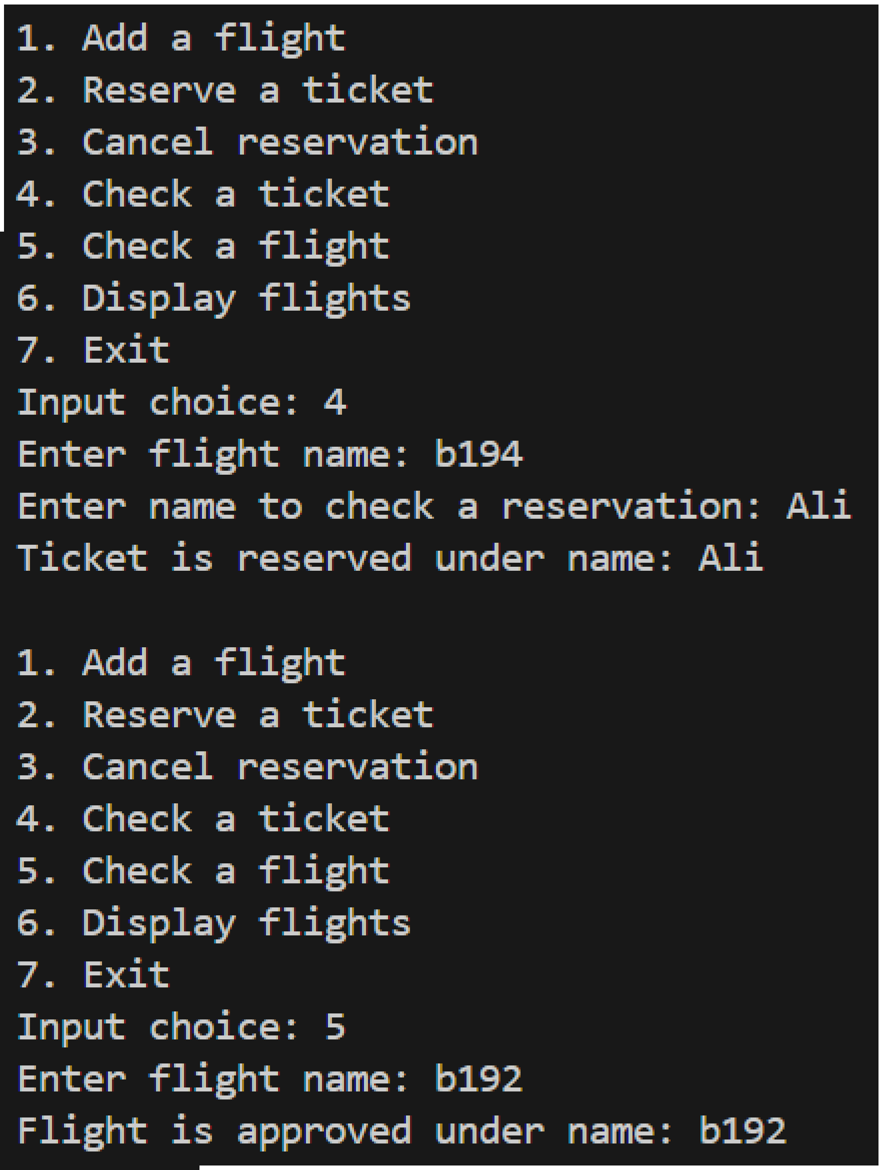
```
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedL
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q3FullerVersion_23K2001
lerVersion_23K2001 }
                ***Welcome to SHAANDAAR Airlines Ticket Reservation System***
1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 1
Enter flight name: b192
Flight under name: b192 has been added.

1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 1
Enter flight name: b194
Flight under name: b194 has been added.
```



```
Input choice: 2
Enter flight name: b194
Enter name to reserve a ticket: Muzammil
Reservation under name: Muzammil has been created.

1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 2
Enter flight name: b194
Enter name to reserve a ticket: Ali
Reservation under name: Ali has been created.

1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 3
Enter flight name: b194
Enter name to cancel a ticket: Muzammil
Reservation under name: Muzammil has been cancelled.
```



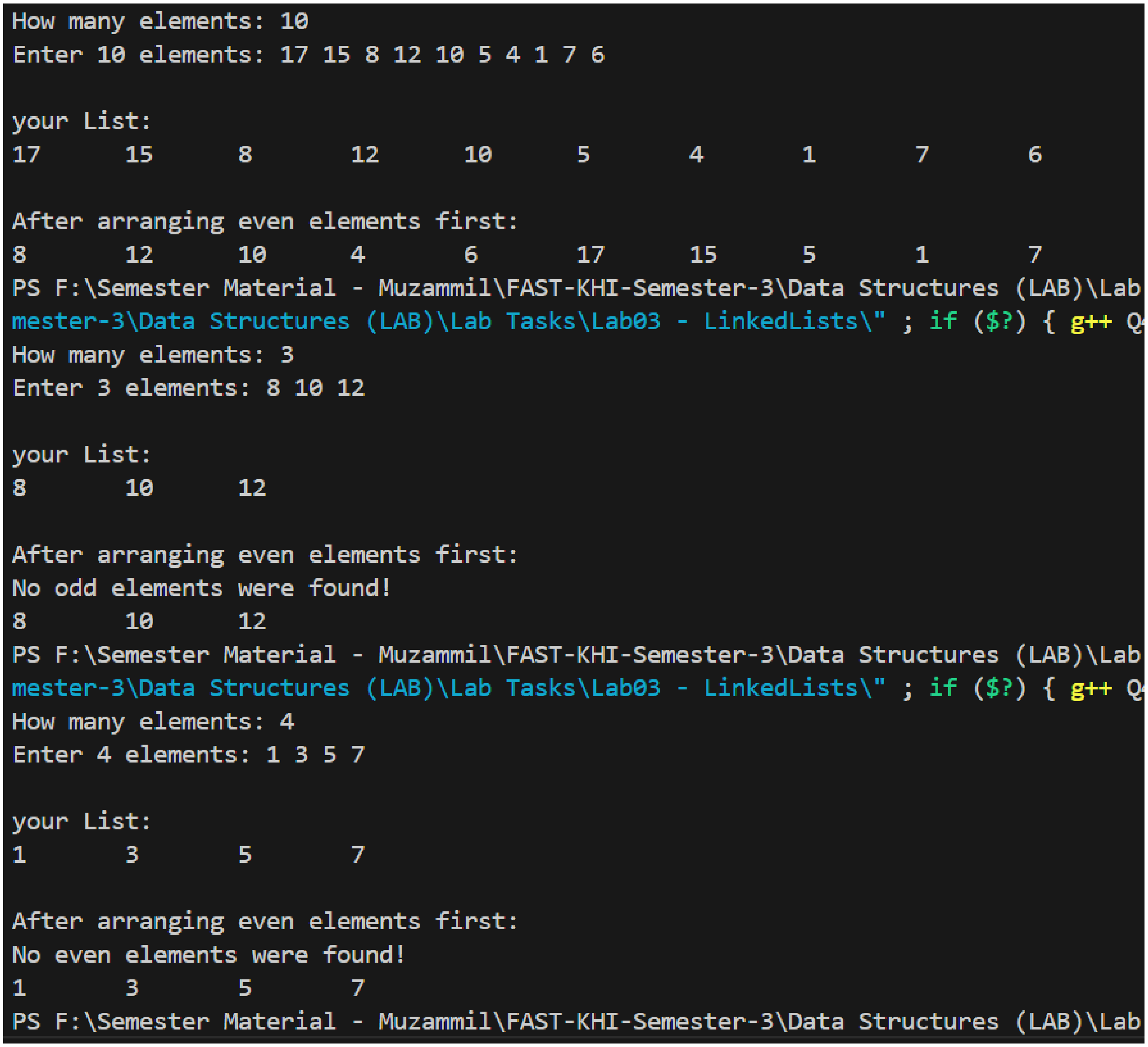
```
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 6
b192
b194


1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 4
Enter flight name: b193
Enter name to check a reservation: a
No flight was found under name: b193

1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 5
Enter flight name: b193
No flight was found under name: b193
```



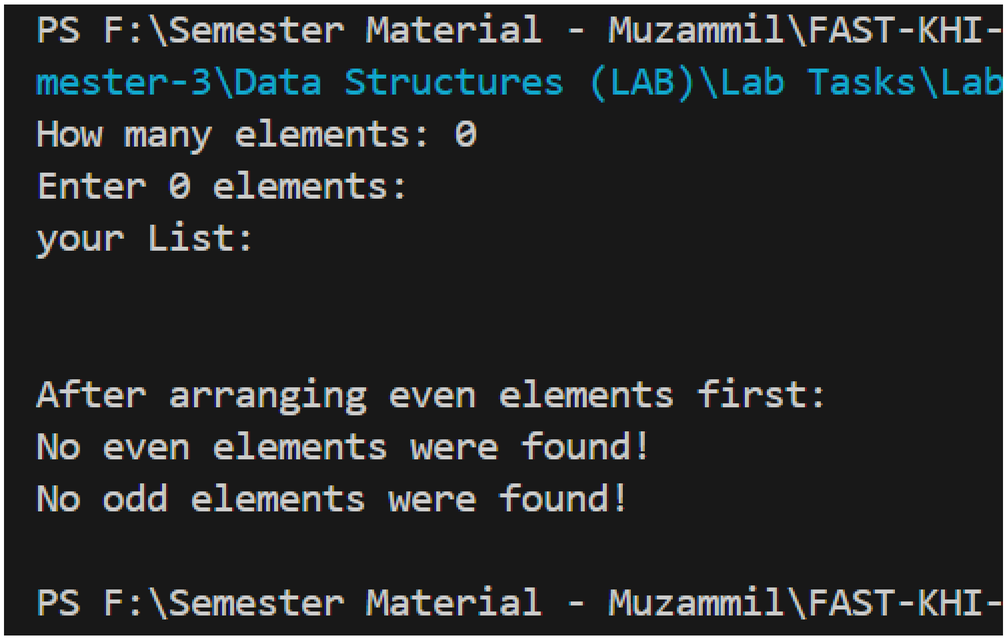
```
1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 4
Enter flight name: b194
Enter name to check a reservation: Ali
Ticket is reserved under name: Ali

1. Add a flight
2. Reserve a ticket
3. Cancel reservation
4. Check a ticket
5. Check a flight
6. Display flights
7. Exit
Input choice: 5
Enter flight name: b192
Flight is approved under name: b192
```

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;
}
```



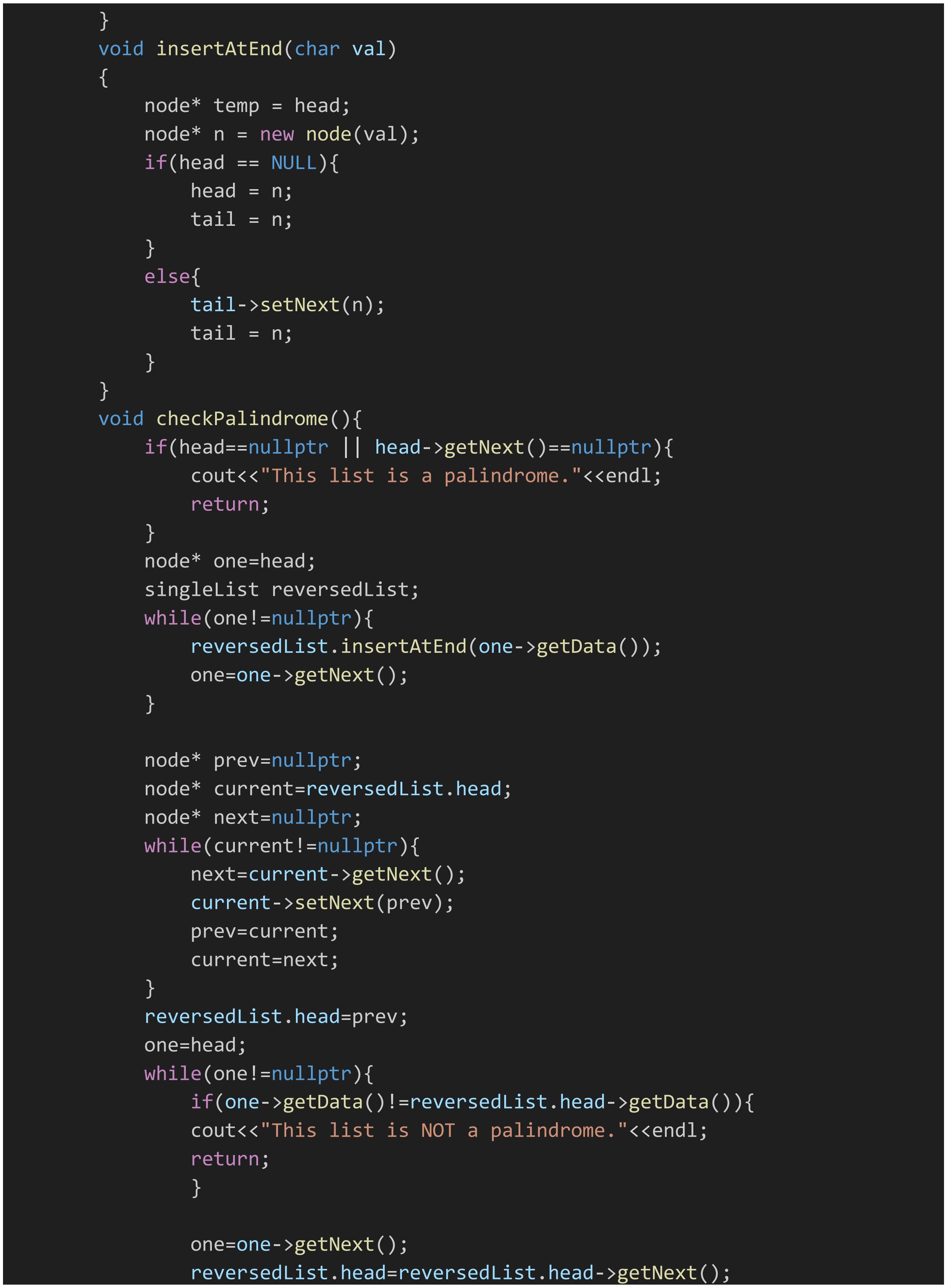
```
How many elements: 10
Enter 10 elements: 17 15 8 12 10 5 4 1 7 6

your List:
17      15      8       12      10      5       4       1       7       6

After arranging even elements first:
8       12      10      4       6       17      15      5       1       7
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q
How many elements: 3
Enter 3 elements: 8 10 12

your List:
8       10      12

After arranging even elements first:
No odd elements were found!
8       10      12
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q
How many elements: 4
Enter 4 elements: 1 3 5 7

your List:
1       3       5       7

After arranging even elements first:
No even elements were found!
1       3       5       7
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab
```



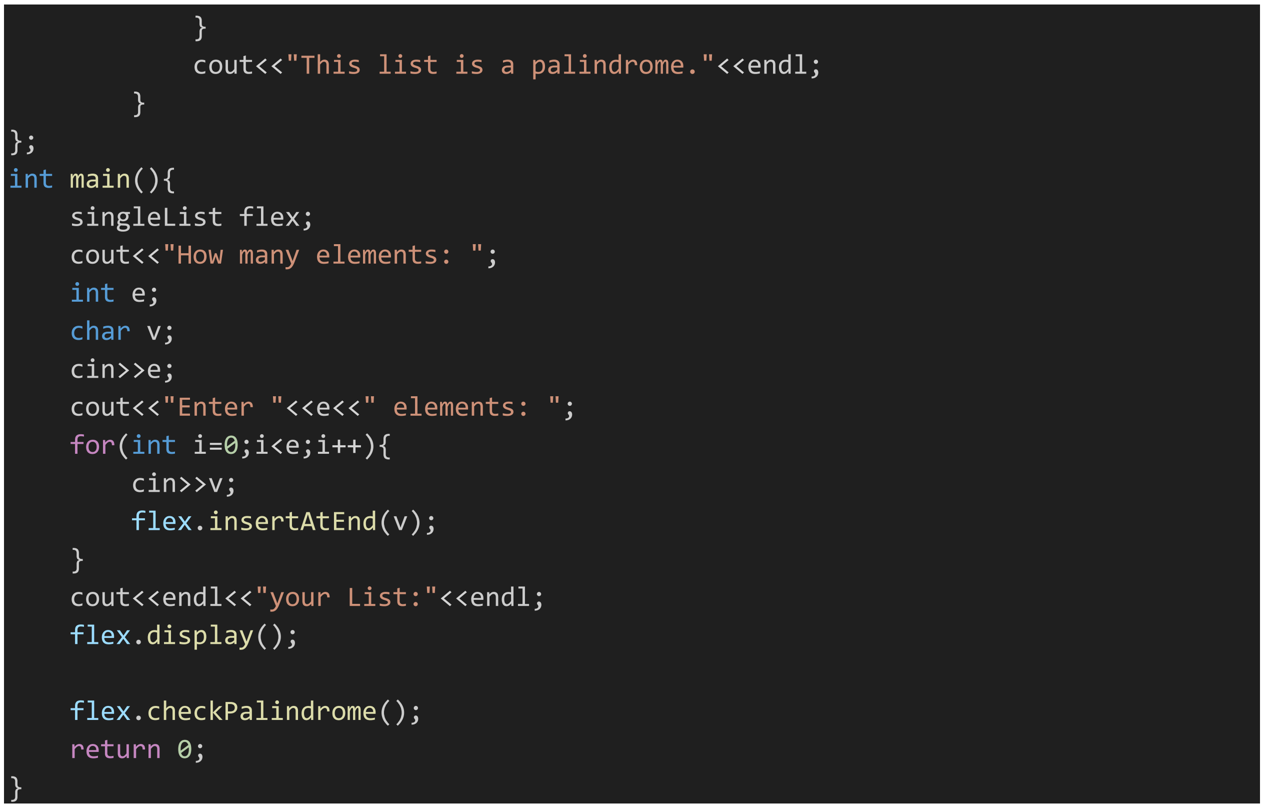
```
PS F:\Semester Material - Muzammil\FAST-KHI-
mester-3\Data Structures (LAB)\Lab Tasks\Lab
How many elements: 0
Enter 0 elements:
your List:


After arranging even elements first:
No even elements were found!
No odd elements were found!

PS F:\Semester Material - Muzammil\FAST-KHI-
```

Q5:

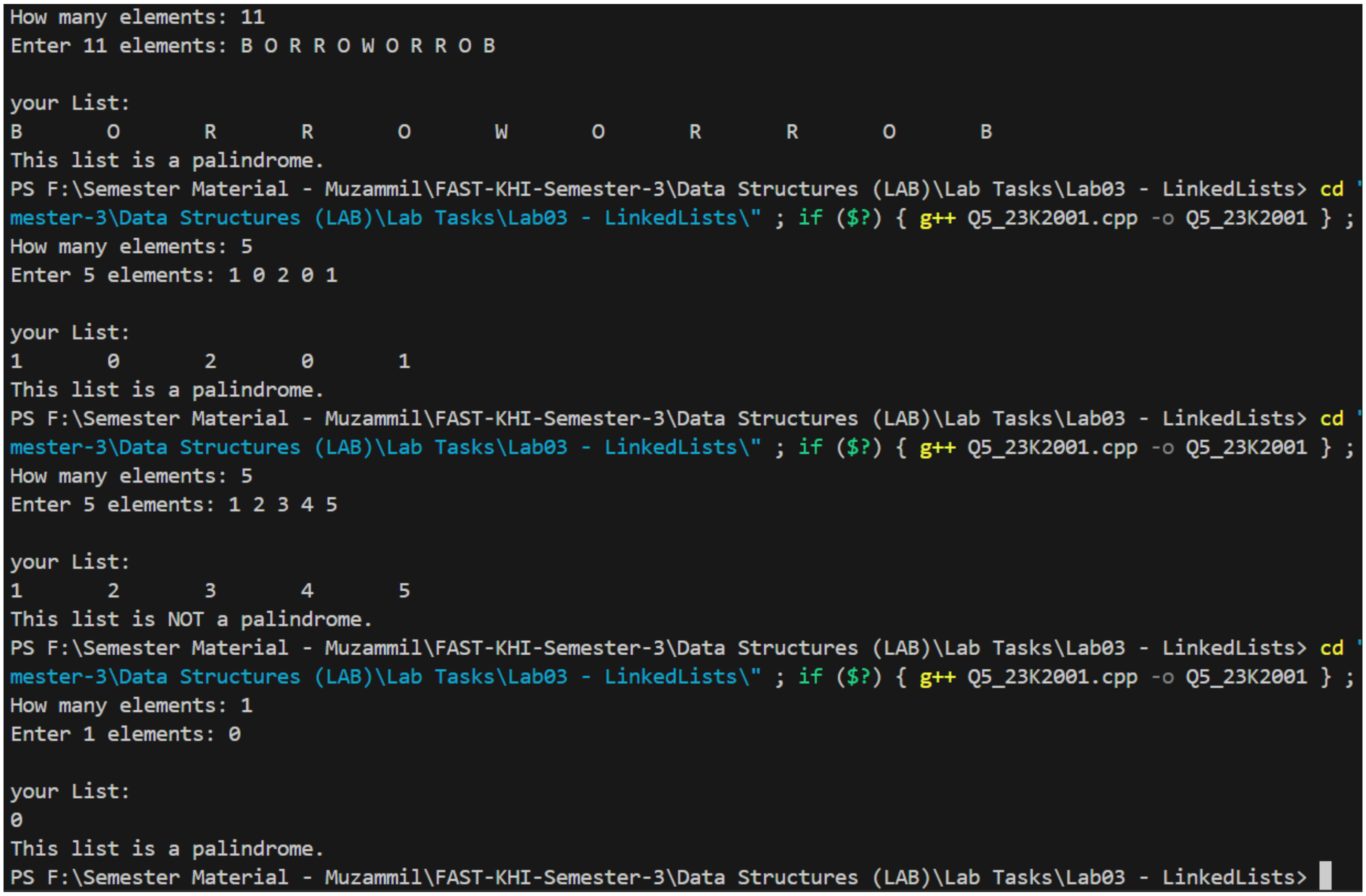
```
//23K2001 Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        char data;
        node* next;
    public:
        node(){next = nullptr;}
        node(char val){
            data = val;
            next = nullptr;
        }
        char getData(){ return data;}
        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();
```



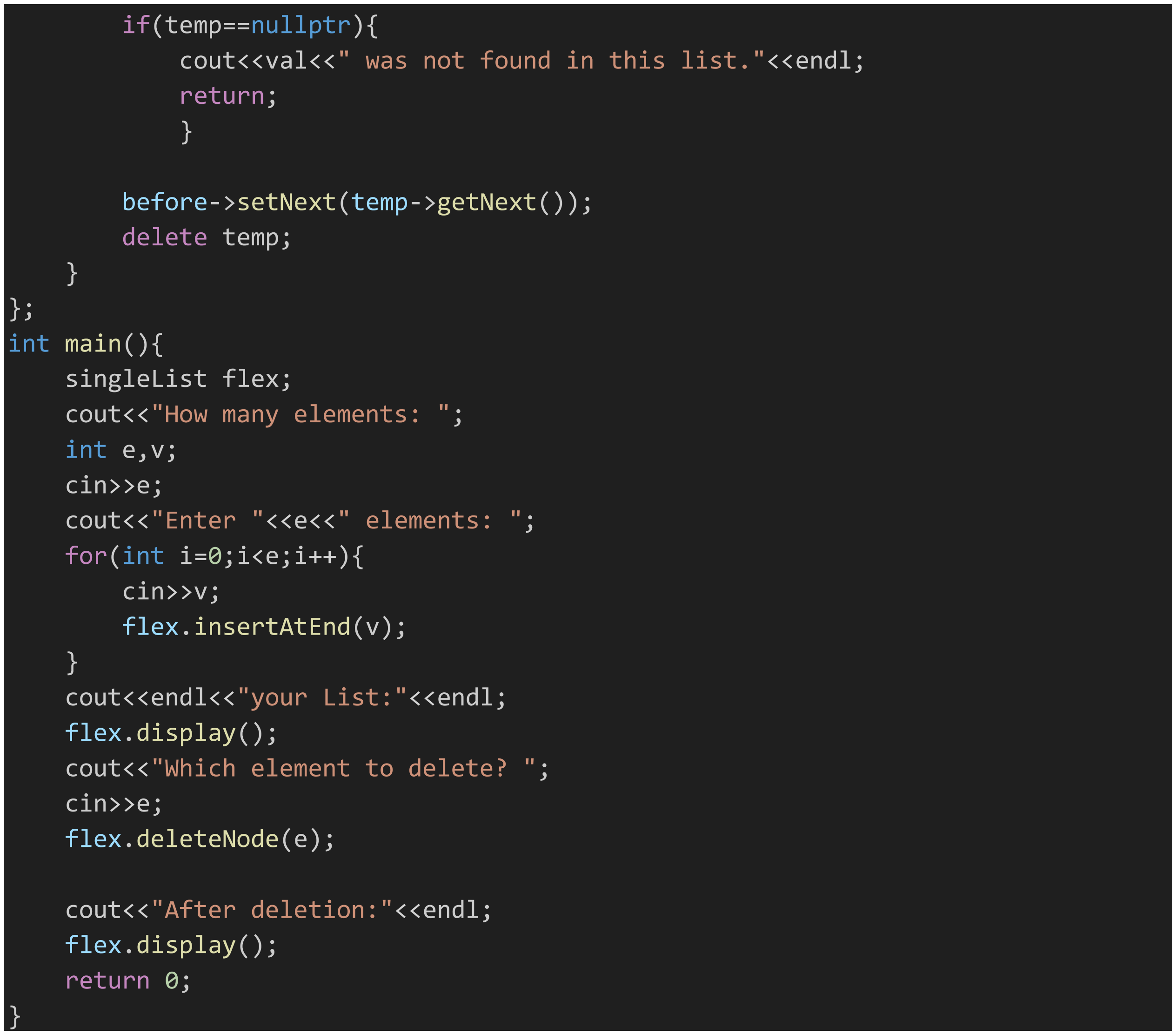
```cpp
            }
            cout<<"This list is a palindrome."<<endl;
        }
};
int main(){
    singleList flex;
    cout<<"How many elements: ";
    int e;
    char v;
    cin>>e;
    cout<<"Enter "<<e<<" elements: ";
    for(int i=0;i<e;i++){
        cin>>v;
        flex.insertAtEnd(v);
    }
    cout<<endl<<"your List:"<<endl;
    flex.display();

    flex.checkPalindrome();
    return 0;
}
```



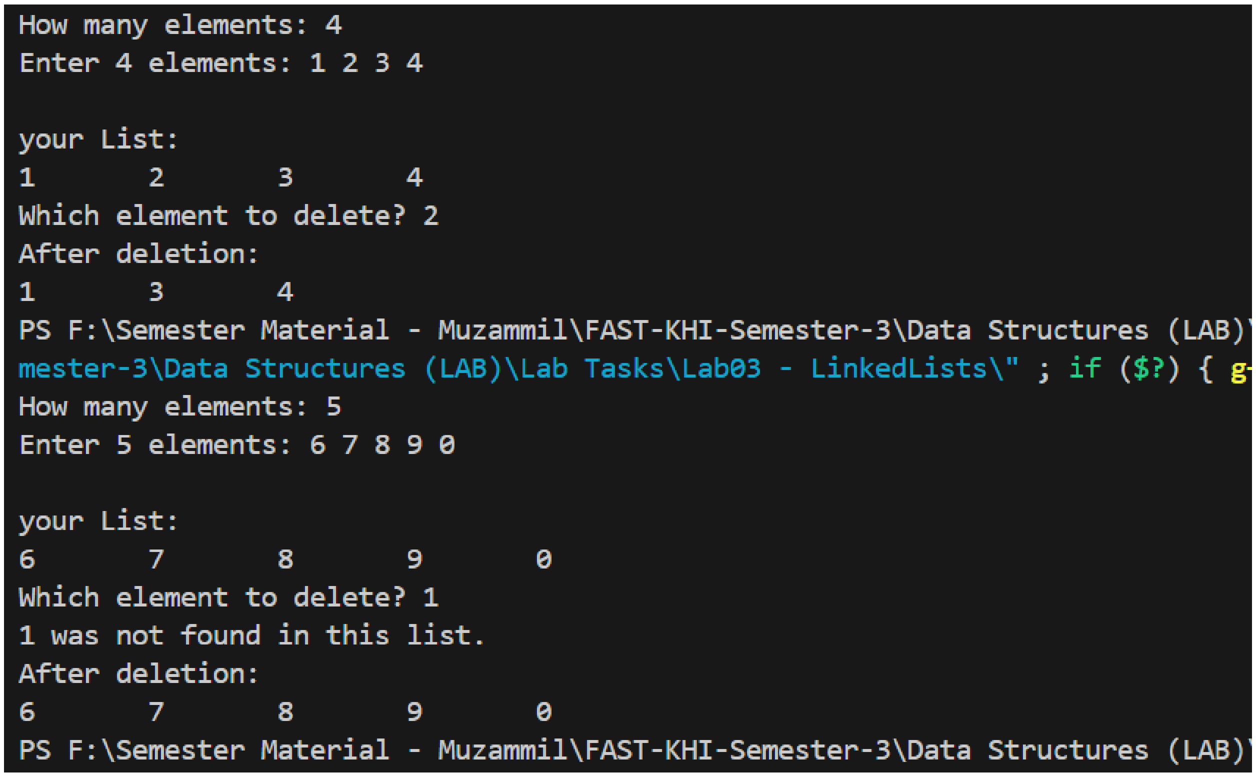
```
How many elements: 11
Enter 11 elements: B O R R O W O R R O B

your List:
B       O       R       R       O       W       O       R       R       O       B
This list is a palindrome.
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists> cd '
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q5_23K2001.cpp -o Q5_23K2001 } ;
How many elements: 5
Enter 5 elements: 1 0 2 0 1

your List:
1       0       2       0       1
This list is a palindrome.
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists> cd '
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q5_23K2001.cpp -o Q5_23K2001 } ;
How many elements: 5
Enter 5 elements: 1 2 3 4 5

your List:
1       2       3       4       5
This list is NOT a palindrome.
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists> cd '
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g++ Q5_23K2001.cpp -o Q5_23K2001 } ;
How many elements: 1
Enter 1 elements: 0

your List:
0
This list is a palindrome.
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists> 
```

Q6:

```cpp
//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();
            }
```



```cpp
        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;
}
```



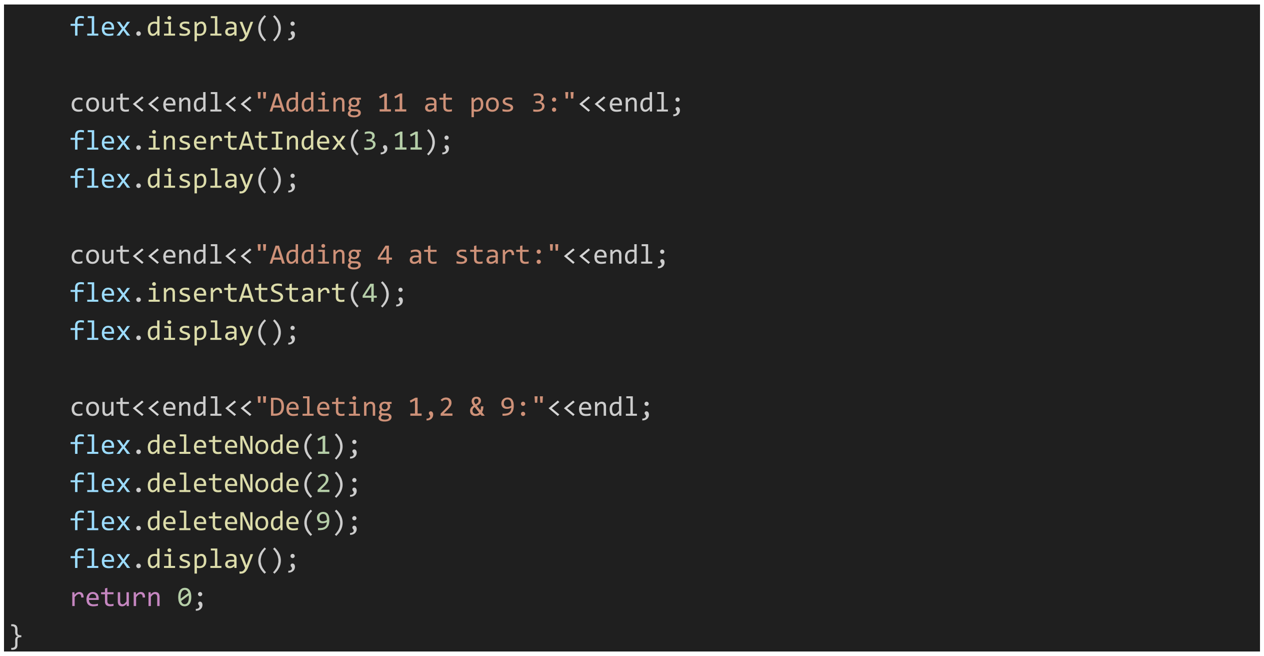
```
How many elements: 4
Enter 4 elements: 1 2 3 4

your List:
1       2       3       4
Which element to delete? 2
After deletion:
1       3       4
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g
How many elements: 5
Enter 5 elements: 6 7 8 9 0

your List:
6       7       8       9       0
Which element to delete? 1
1 was not found in this list.
After deletion:
6       7       8       9       0
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)
```

Q7:

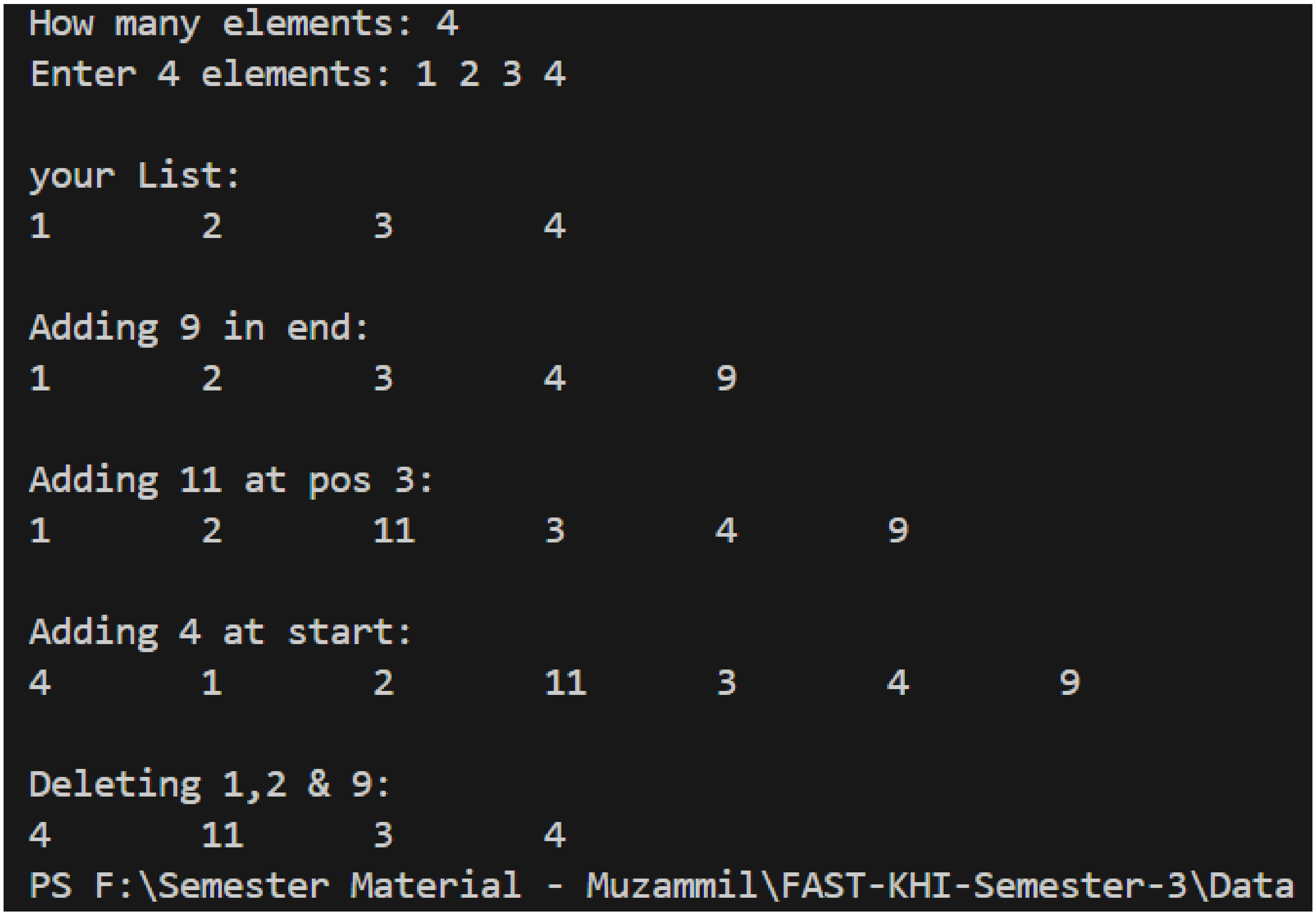
```cpp
//23K2001 Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        int data;
        node* next;
    public:
        node(){next = nullptr;}
        node(int val){
            data = val;
            next = nullptr;
        }

        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){
```

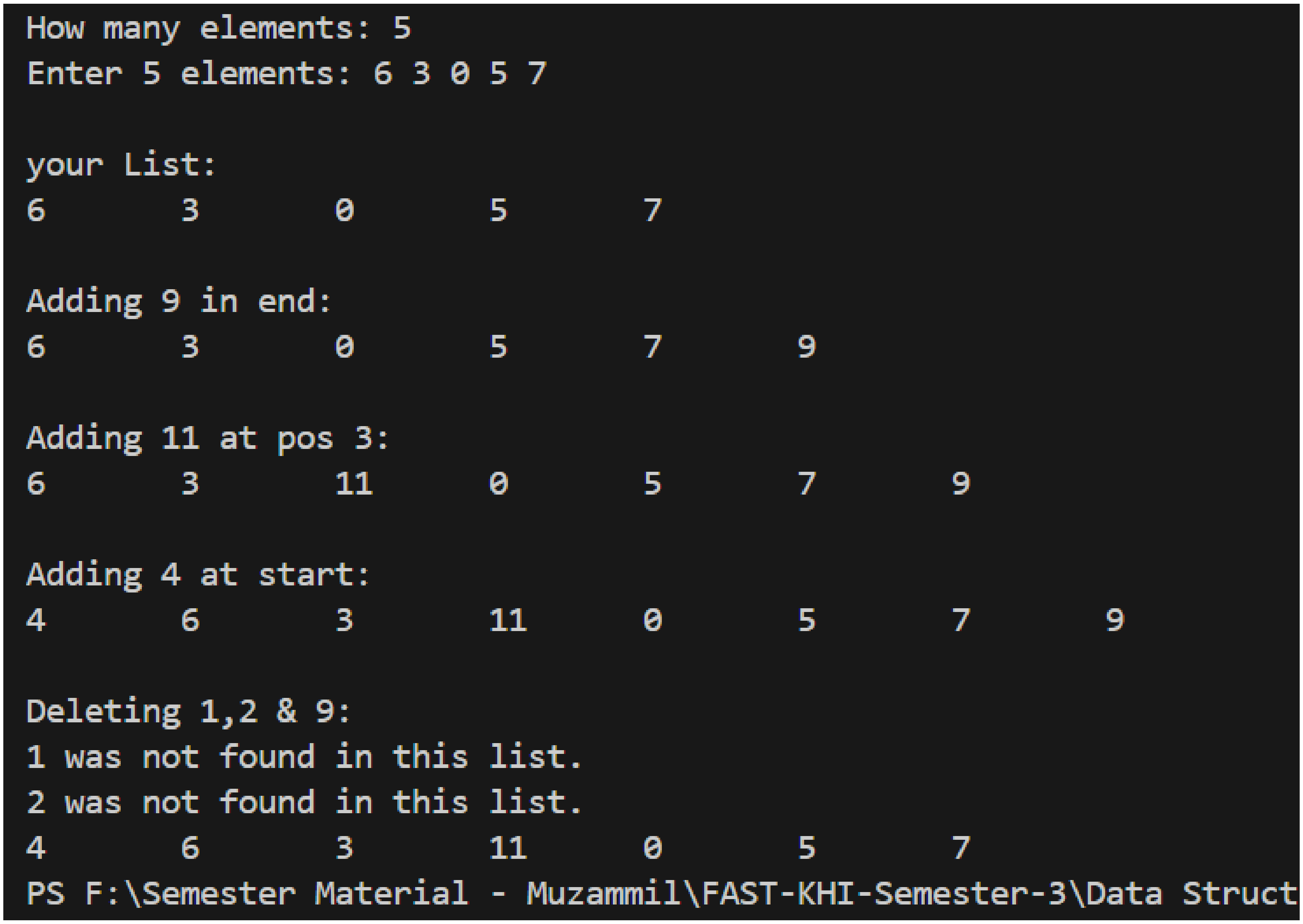
```cpp
                    head = nullptr;
                    tail = nullptr;
                }
                else{
                    head = head->getNext();
                    tail->setNext(head);
                }

                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;
}
```



```
How many elements: 4
Enter 4 elements: 1 2 3 4

your List:
1       2       3       4

Adding 9 in end:
1       2       3       4       9

Adding 11 at pos 3:
1       2       11      3       4       9

Adding 4 at start:
4       1       2       11      3       4       9

Deleting 1,2 & 9:
4       11      3       4
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data
```

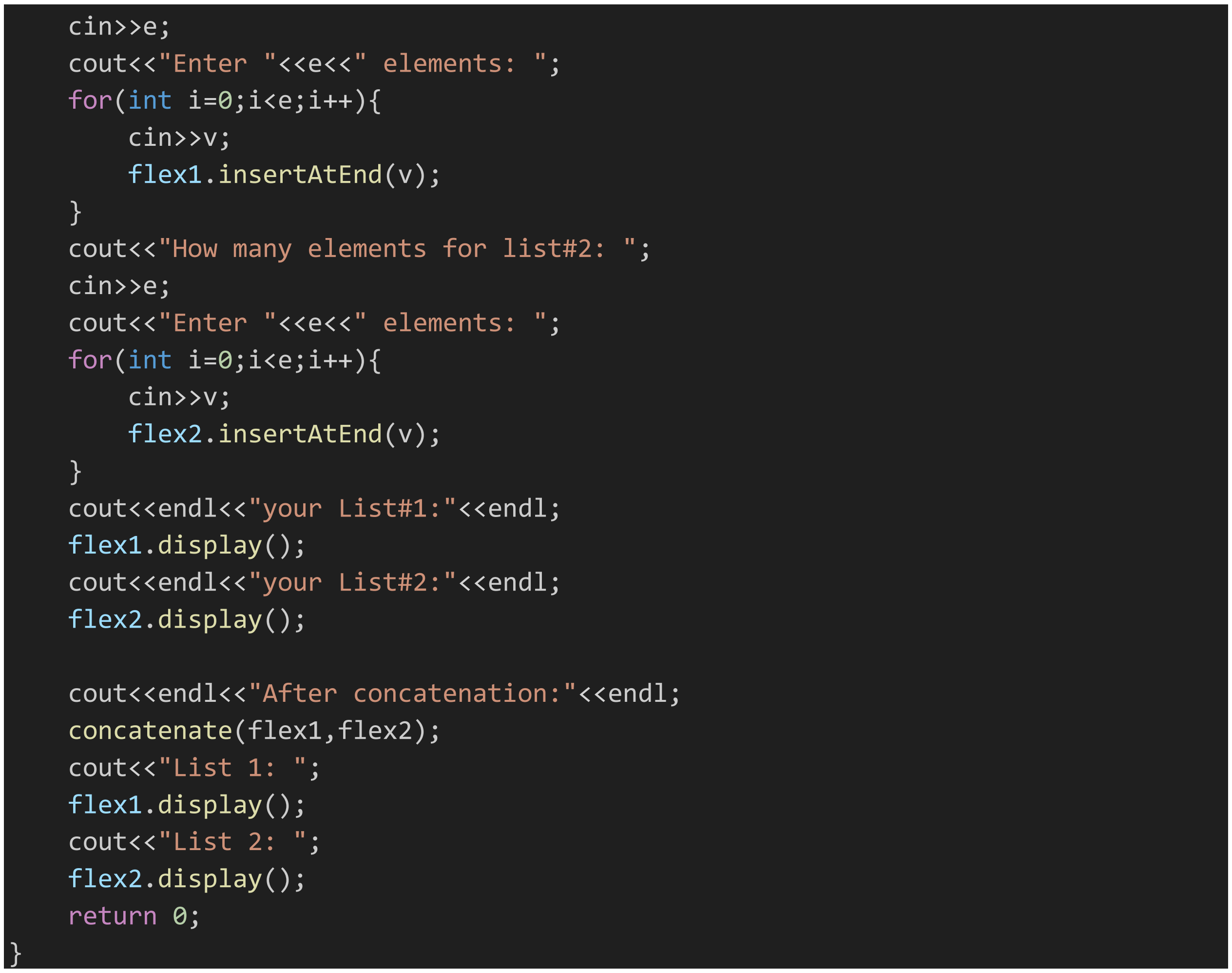


```
How many elements: 5
Enter 5 elements: 6 3 0 5 7

your List:
6       3       0       5       7

Adding 9 in end:
6       3       0       5       7       9

Adding 11 at pos 3:
6       3       11      0       5       7       9

Adding 4 at start:
4       6       3       11      0       5       7       9

Deleting 1,2 & 9:
1 was not found in this list.
2 was not found in this list.
4       6       3       11      0       5       7
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Struct
```

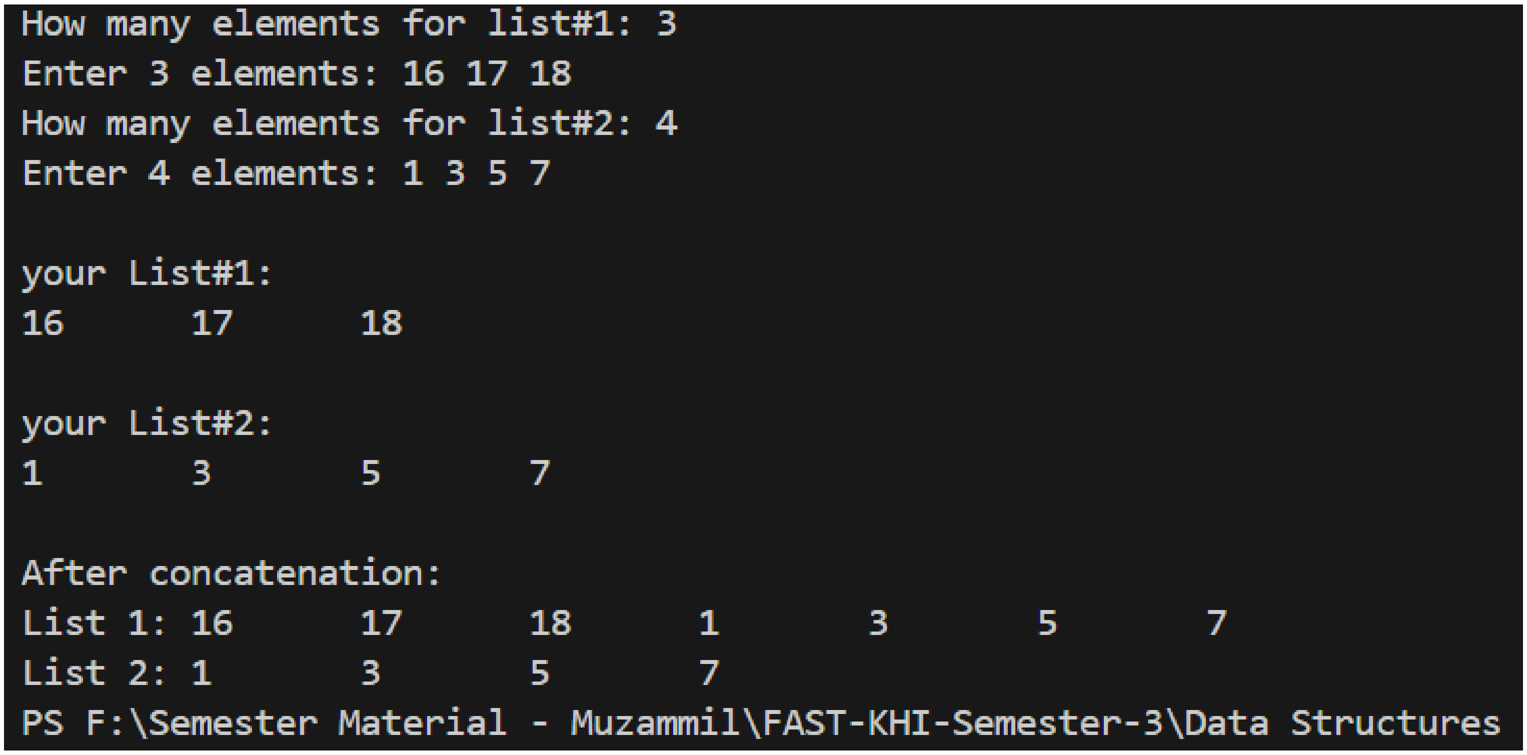
Q8:

```
//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; }
```

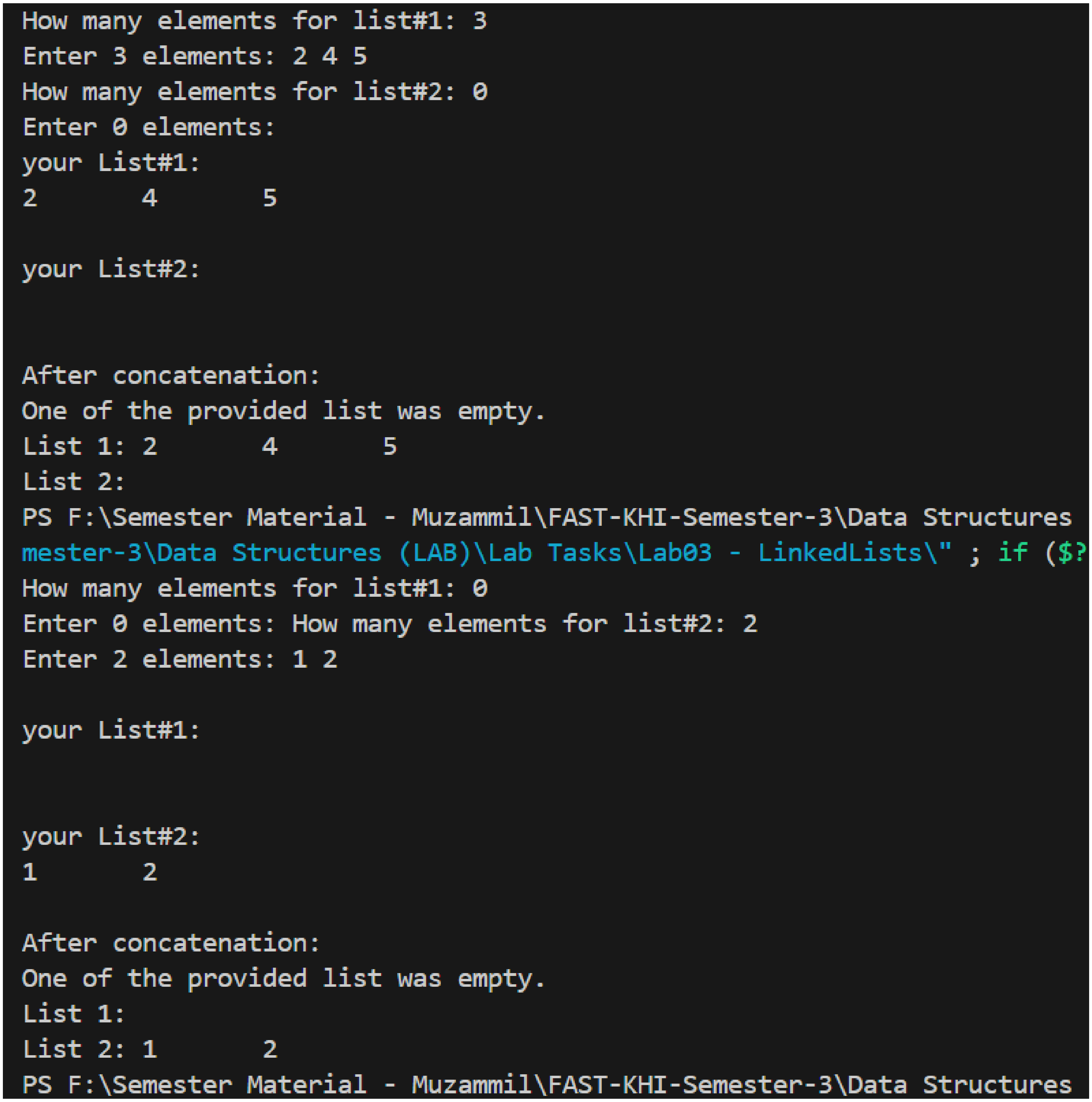
```cpp
        node* getTail(){ return tail; }
        void setHead(node* update){head = update;}
        void setTail(node* update){tail = update;}

        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;
}
```



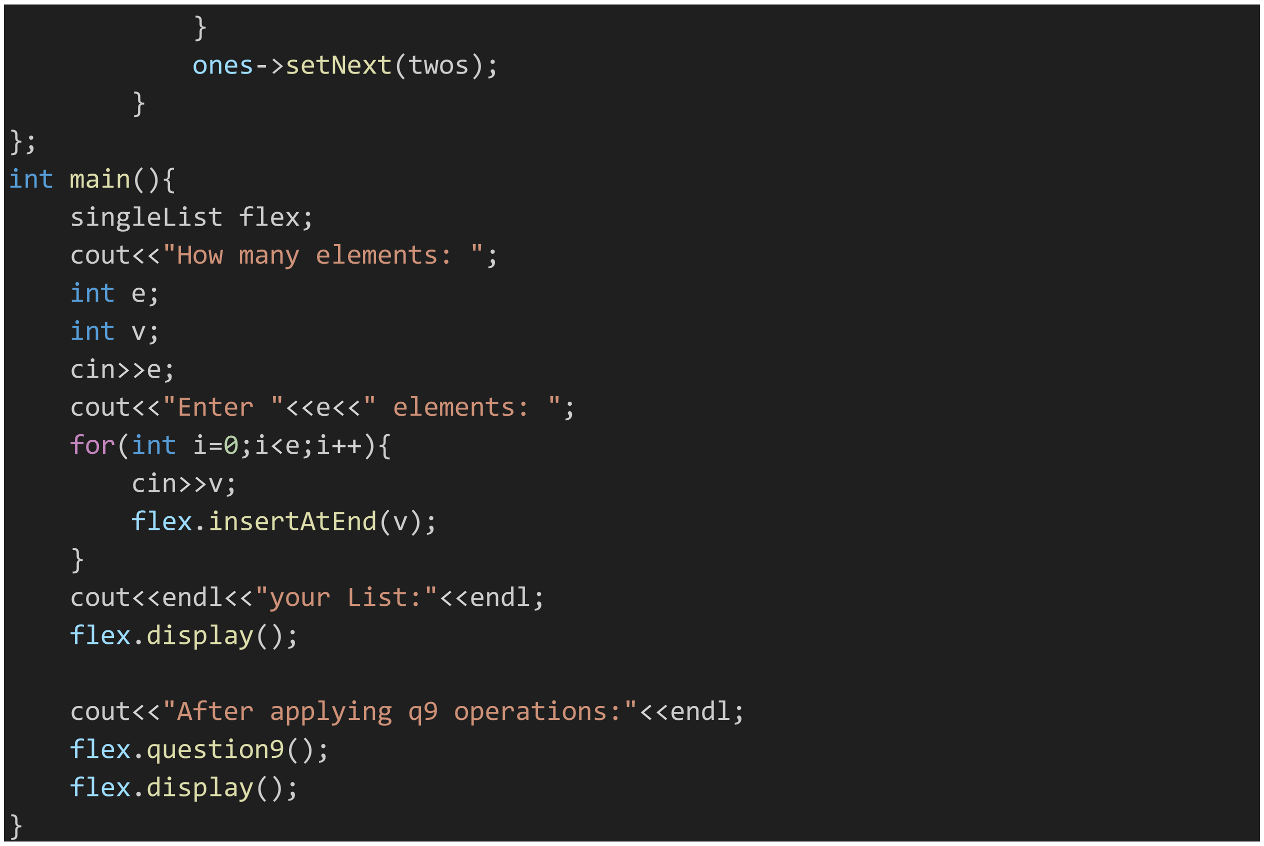
```
How many elements for list#1: 3
Enter 3 elements: 16 17 18
How many elements for list#2: 4
Enter 4 elements: 1 3 5 7

your List#1:
16      17      18

your List#2:
1       3       5       7

After concatenation:
List 1: 16      17      18      1       3       5       7
List 2: 1       3       5       7
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```



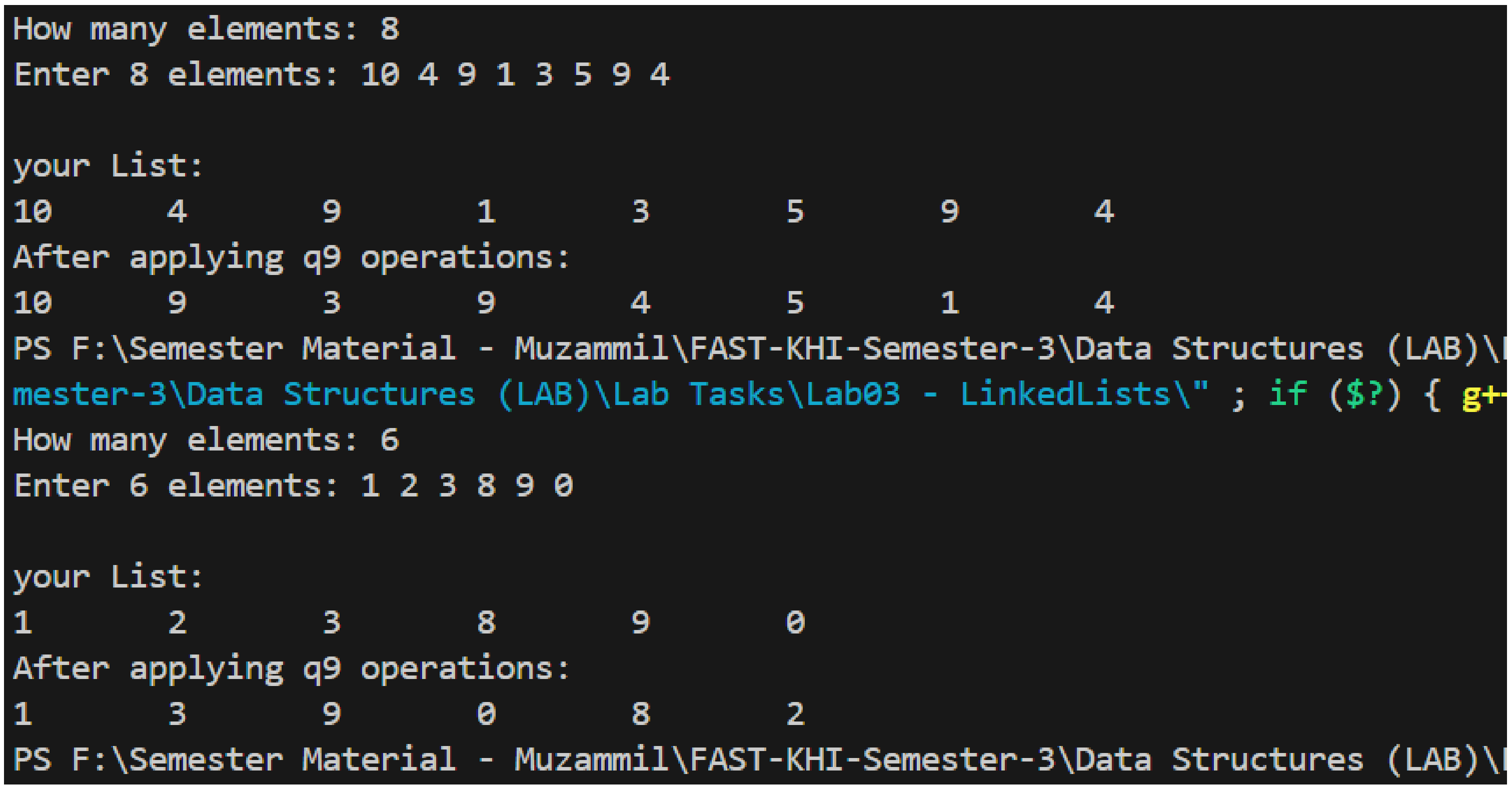
```
How many elements for list#1: 3
Enter 3 elements: 2 4 5
How many elements for list#2: 0
Enter 0 elements:
your List#1:
2       4       5

your List#2:


After concatenation:
One of the provided list was empty.
List 1: 2       4       5
List 2:
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?
How many elements for list#1: 0
Enter 0 elements: How many elements for list#2: 2
Enter 2 elements: 1 2

your List#1:


your List#2:
1       2

After concatenation:
One of the provided list was empty.
List 1:
List 2: 1       2
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```

Q9:

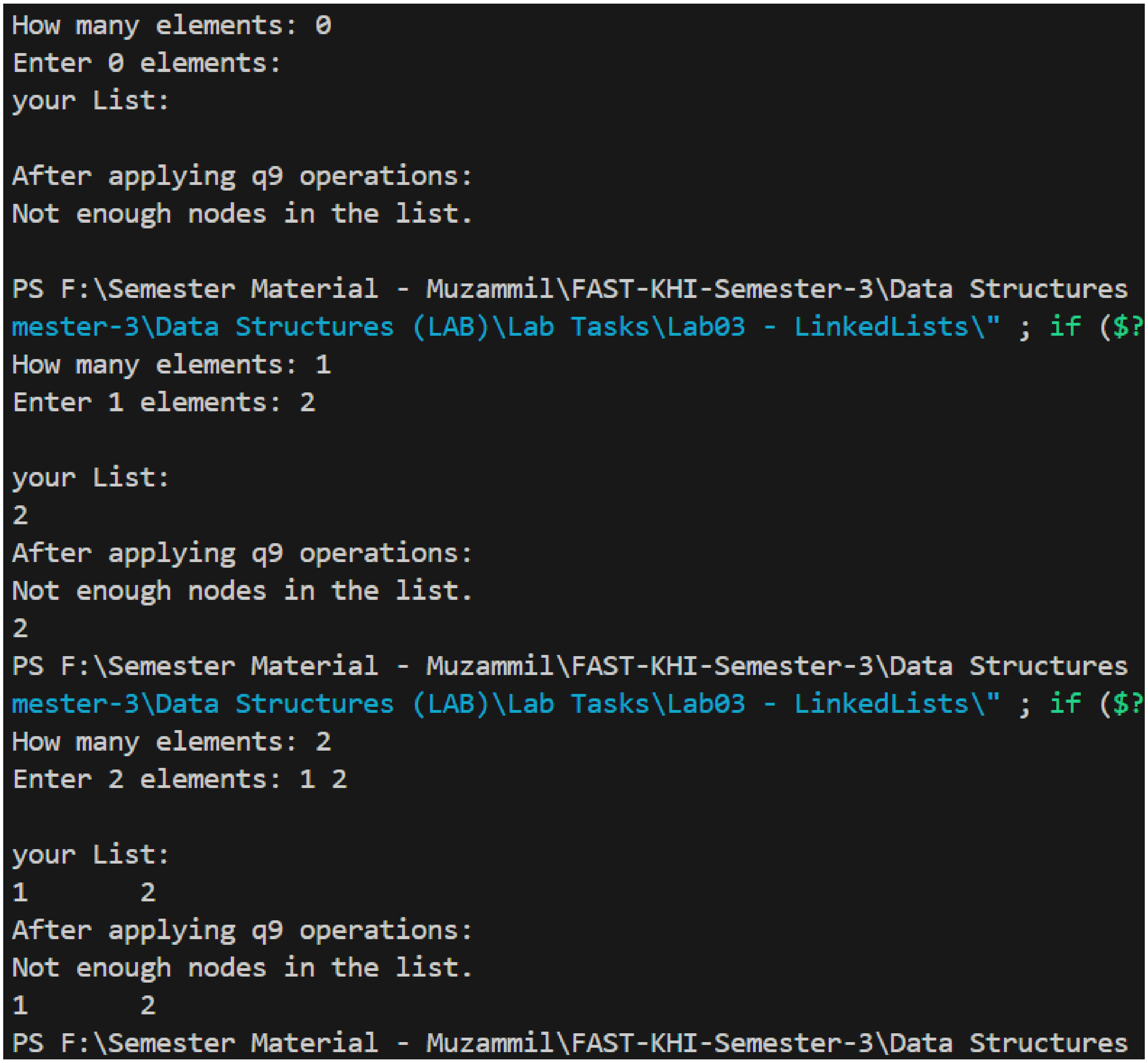
```
//23K2001 Muzammil
#include<iostream>
using namespace std;
class node{
    private:
        int data;
        node* next;
    public:
        node(){next = nullptr;}
        node(int val){
            data = val;
            next = nullptr;
        }

        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;
```



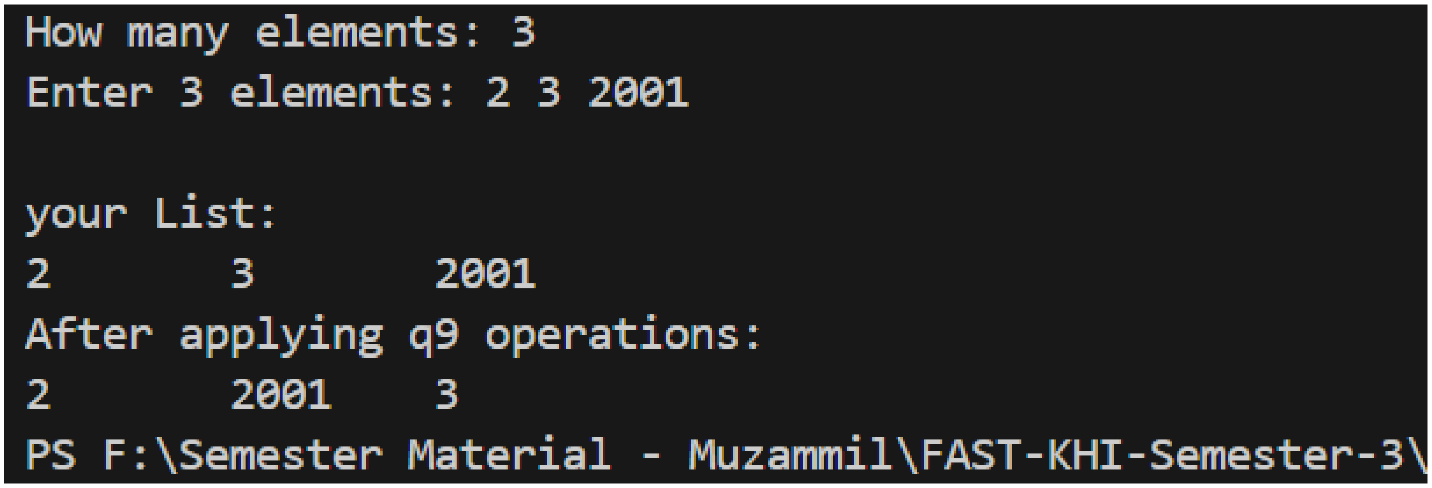
```cpp
            }
            ones->setNext(twos);
        }
};
int main(){
    singleList flex;
    cout<<"How many elements: ";
    int e;
    int v;
    cin>>e;
    cout<<"Enter "<<e<<" elements: ";
    for(int i=0;i<e;i++){
        cin>>v;
        flex.insertAtEnd(v);
    }
    cout<<endl<<"your List:"<<endl;
    flex.display();

    cout<<"After applying q9 operations:"<<endl;
    flex.question9();
    flex.display();
}
```



```
How many elements: 8
Enter 8 elements: 10 4 9 1 3 5 9 4

your List:
10      4       9       1       3       5       9       4
After applying q9 operations:
10      9       3       9       4       5       1       4
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?) { g+
How many elements: 6
Enter 6 elements: 1 2 3 8 9 0

your List:
1       2       3       8       9       0
After applying q9 operations:
1       3       9       0       8       2
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures (LAB)\
```



```
How many elements: 0
Enter 0 elements:
your List:

After applying q9 operations:
Not enough nodes in the list.

PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?
How many elements: 1
Enter 1 elements: 2

your List:
2
After applying q9 operations:
Not enough nodes in the list.
2
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
mester-3\Data Structures (LAB)\Lab Tasks\Lab03 - LinkedLists\" ; if ($?
How many elements: 2
Enter 2 elements: 1 2

your List:
1       2
After applying q9 operations:
Not enough nodes in the list.
1       2
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\Data Structures
```



```
How many elements: 3
Enter 3 elements: 2 3 2001

your List:
2       3       2001
After applying q9 operations:
2       2001    3
PS F:\Semester Material - Muzammil\FAST-KHI-Semester-3\
```