***Muhammad Mufeez***

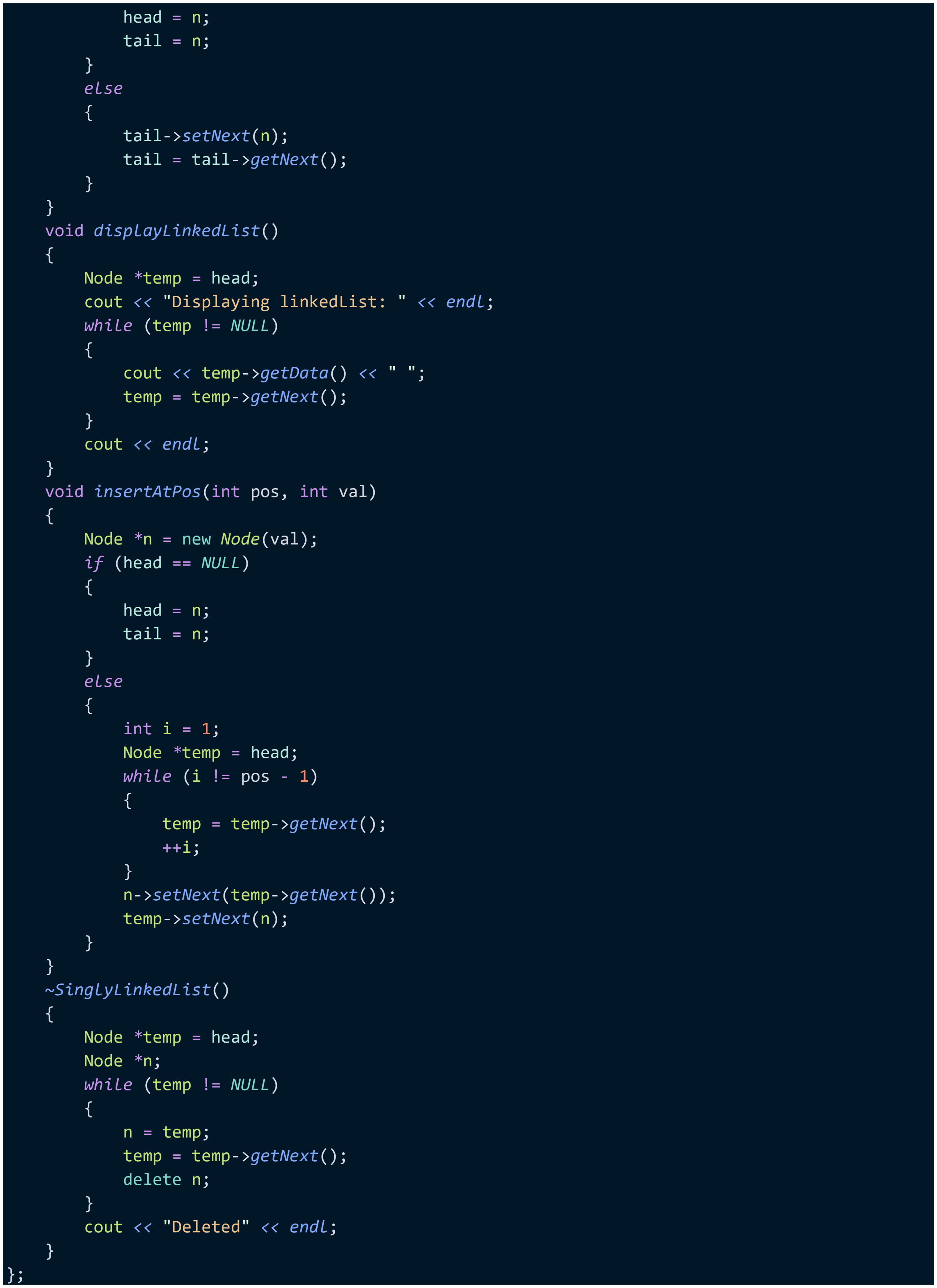
***BCS-3J***

***DS Lab 03***

***Q1:***

*Code:*

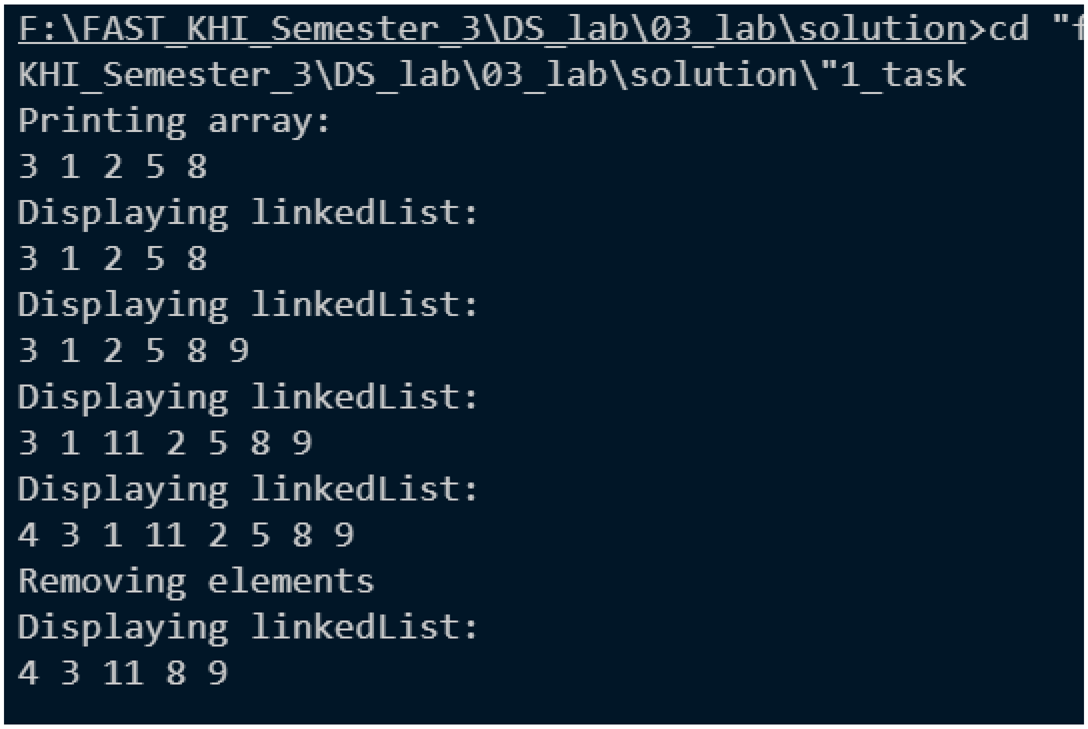
```
#include <iostream>

using namespace std;

class Node
{
private:
    int data;
    Node *next;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    int getData()
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class SinglyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void turnArrToLL(int arr[])
    {
        head = new Node(arr[0]);
        Node *temp = head;
        for (int i = 1; i < 5; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
        tail = temp;
    }

    void insertAtFront(int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
        }
        else
        {
            n->setNext(head);
            head = n;
        }
    }
    void removeElements(int val)
    {
        Node *temp = head;
        while (temp->getNext()->getData() != val)
        {
            temp = temp->getNext();
        }
        Node *temp1 = temp->getNext();
        temp->setNext(temp->getNext()->getNext());
        delete temp1;
    }
    void insertAtTail(int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
```



```cpp
            head = n;
            tail = n;
        }
        else
        {
            tail->setNext(n);
            tail = tail->getNext();
        }
    }
    void displayLinkedList()
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }
    void insertAtPos(int pos, int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
        }
        else
        {
            int i = 1;
            Node *temp = head;
            while (i != pos - 1)
            {
                temp = temp->getNext();
                ++i;
            }
            n->setNext(temp->getNext());
            temp->setNext(n);
        }
    }
    ~SinglyLinkedList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
    }
};
```

```cpp
int main()
{
    int arr[5] = {3, 1, 2, 5, 8};
    cout << "Printing array: " << endl;
    for (int i = 0; i < 5; ++i)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    SinglyLinkedList *list1 = new SinglyLinkedList();
    list1->turnArrToLL(arr);
    list1->displayLinkedList();
    list1->insertAtTail(9);
    list1->displayLinkedList();
    list1->insertAtPos(3, 11);
    list1->displayLinkedList();
    list1->insertAtFront(4);
    list1->displayLinkedList();
    cout << "Removing elements" << endl;
    list1->removeElements(1);
    list1->removeElements(2);
    list1->removeElements(5);
    list1->displayLinkedList();

    return 0;
}
```

*Output:*



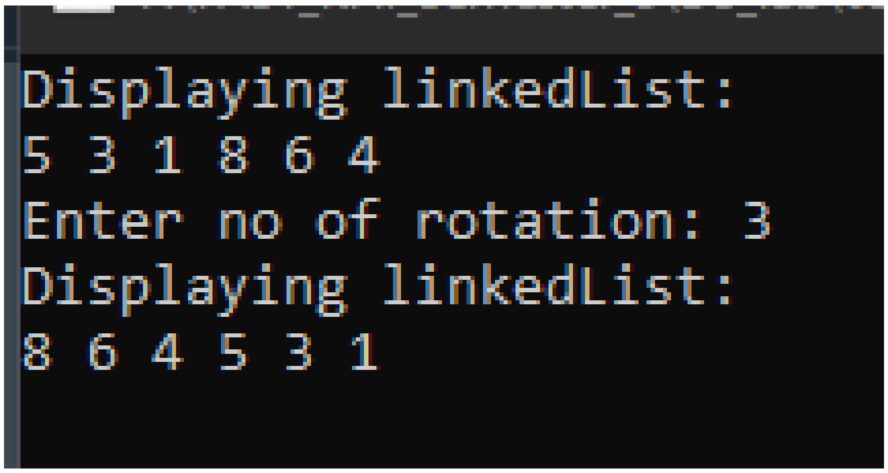
```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f
KHI_Semester_3\DS_lab\03_lab\solution\"1_task
Printing array:
3 1 2 5 8
Displaying linkedList:
3 1 2 5 8
Displaying linkedList:
3 1 2 5 8 9
Displaying linkedList:
3 1 11 2 5 8 9
Displaying linkedList:
4 3 1 11 2 5 8 9
Removing elements
Displaying linkedList:
4 3 11 8 9
```

***Q2:***

*Code:*

```cpp
#include <iostream>

using namespace std;

class Node
{
private:
    int data;
    Node *next;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    int getData()
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class SinglyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    void turnArrToLL(int arr[], int size)
    {
        head = new Node(arr[0]);
        Node *temp = head;
        for (int i = 1; i < size; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
        tail = temp;
    }
    void rotateList(int n)
    {
        // considering that n is less than the length of linked list
        int i = 0;
        Node *temp = head;
        while (i < n - 1)
        {
            temp = temp->getNext();
            i++;
        }
        Node *temp1 = head;
        head = temp->getNext();
        temp->setNext(NULL);
        tail->setNext(temp1);
        tail = temp;
    }
    void displayLinkedList()
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }

    ~SinglyLinkedList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
    }
};

int main()
{
    int arr[] = {5, 3, 1, 8, 6, 4, 2};
    int size = 6;
    int n;
    SinglyLinkedList *ptr = new SinglyLinkedList();
    ptr->turnArrToLL(arr, size);
    ptr->displayLinkedList();
    cin >> n;
    ptr->rotateList(n);
    ptr->displayLinkedList();
    return 0;
}
```

*Output:*



```
Displaying linkedList:
5 3 1 8 6 4
Enter no of rotation: 3
Displaying linkedList:
8 6 4 5 3 1
```

***Q3:***

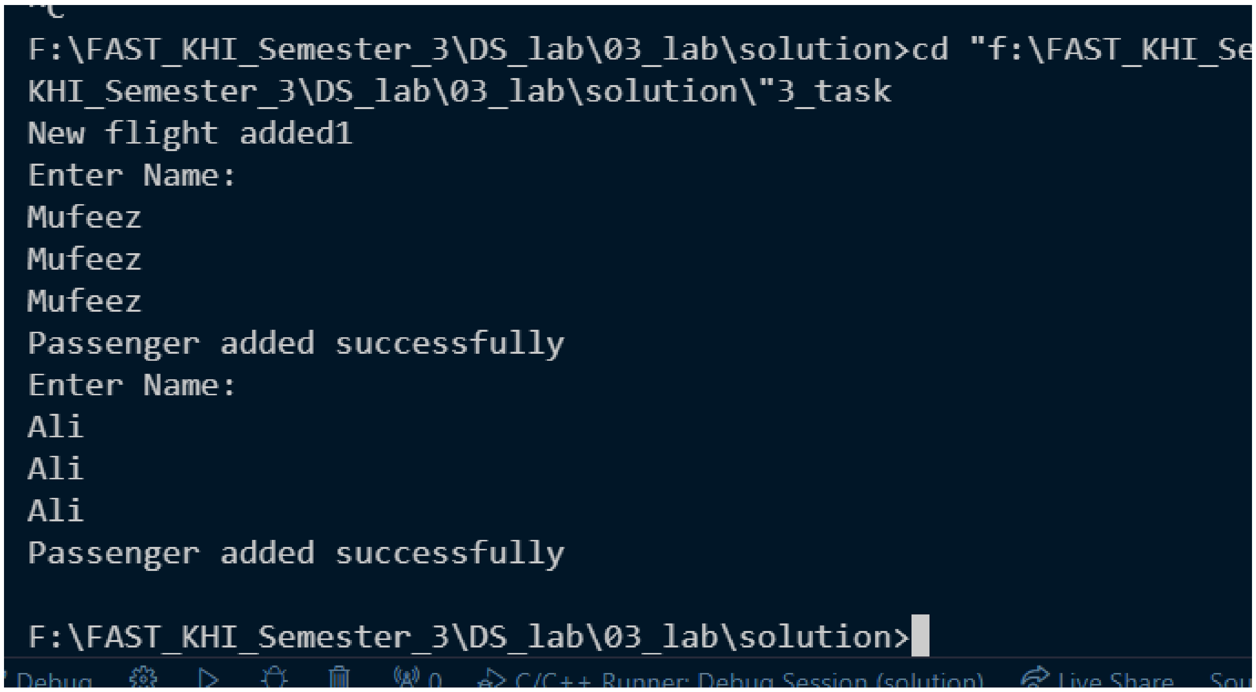
*Code:*

```cpp
#include <bits/stdc++.h>

using namespace std;
template <typename T>
class Node
{
public:
    T data;
    Node *next;

public:
    Node(T data)
    {
        this->data = data;
        this->next = NULL;
    }
    Node()
    {
        next = NULL;
    }

    void setNext(Node<T> *next)
    {
        this->next = next;
    }
    Node<T> *getNext()
    {
        return next;
    }
    T getData()
    {
        return data;
    }
    void setData(T data)
    {
        this->data = data;
    }
};
template <typename T>
class SinglyLinkedList
{
public:
    Node<T> *listHead;
    Node<T> *listTail;
    SinglyLinkedList<T> *flightHead;
    SinglyLinkedList<T> *flightNext;

    SinglyLinkedList<T> *flightTail;
    SinglyLinkedList<T> *passengers;
    int N; // no of flights

public:
    SinglyLinkedList()
    {
        listHead = NULL;
        listTail = NULL;
        flightHead = NULL;
        flightTail = NULL;
        passengers = NULL;
        flightNext = NULL;
        N = 0;
    }
    void addNewFlight()
    {
        SinglyLinkedList<T> *n = new SinglyLinkedList<T>();
        SinglyLinkedList<T> *mover;
        if (flightHead == NULL)
        {
            flightHead = n;
            flightTail = n;
        }
        else
        {
            mover = flightHead;
            while (mover->flightNext != NULL)
            {
                mover = mover->flightNext;
            }
            mover->flightNext = n;
            flightTail = n;
        }
        ++N;
        cout << "New flight added" << N << endl;
    }
    void reserveTicket(int n)
    {

        if (n > N)
        {
            cout << "flight of this number doesn't exist" << endl;
            return;
        }
        SinglyLinkedList<T> *mover = flightHead;
        if (n != 1)
            while (n--)
            {
                mover = mover->flightNext;
            }

        string name;
        cout << "Enter Name: " << endl;
        getline(cin, name);
        Node<T> *newNode = new Node<T>(name);
        cout << name << endl;
        if (passengers == NULL)
        {
            passengers = new SinglyLinkedList<T>();
            passengers->listHead = newNode;
            passengers->listTail = newNode;
            cout << name << endl;
        }
        else
        {
            cout << name << endl;
            passengers->listTail->setNext(newNode);
            passengers->listTail = passengers->listTail->getNext();
        }
        cout << "Passenger added successfully" << endl;
    }
```

```
    bool checkSeat(T val)
    {

        // assuming that user will enter the correct flight number
        cout << "Enter flight number: ";
        int n;
        cin >> n;
        if (n > N)
        {
            cout << "This number of flight does not exist" << endl;
            return false;
        }
        SinglyLinkedList<T> *mover = flightHead;
        if (n != 1)
            while (n--)
            {
                mover = mover->flightNext;
            }

        if (mover->passengers->listHead == NULL)
        {
            cout << "passenger not found" << endl;
            return false;
        }
        else
        {
            Node<T> *listMover = mover->passengers->listHead;
            while (listMover->next != NULL)
            {
                if (listMover->data == val)
                {
                    cout << "passenger found " << endl;
                    return true;
                }
                listMover = listMover->next;
            }
            return false;
        }
    }
    void displaySystem()
    {
        SinglyLinkedList<T> *mover = flightHead;
        while (mover != NULL)
        {
            cout << "\tDisplaying next flight passengers: " << endl;
            mover->displayPeople(mover);
            mover = mover->flightNext;
        }
    }
    void displayPeople(SinglyLinkedList<T> *temp)
    {
        Node<T> *mover = temp->passengers->listHead;
        cout << "\t\tDisplaying Passengers of current flight: " << endl;
        while (mover != NULL)
```

```cpp
        {
            cout << mover->data << " ";
            mover = mover->next;
        }
        cout << endl;
    }
    void cancelSeat(T val)
    {
        int n;
        cout << "Enter you flight number: ";
        cin >> n;

        SinglyLinkedList<T> *mover = flightHead;
        if (mover == NULL)
            return;
        if (n != 1)
            while (n--)
            {
                mover = mover->flightNext;
            }

        Node<T> *temp = mover->passengers->listHead;
        if (temp == NULL)
        {
            cout << "No passengers in this flight" << endl;
            return;
        }
        while (temp->getNext()->getData() != val && temp->getNext() != NULL)
        {
            temp = temp->getNext();
        }
        Node<T> *temp1 = temp->getNext();
        temp->setNext(temp->getNext()->getNext());
        delete temp1;
        cout << "Passenger's seat has been cancelled" << endl;
    }
};

int main()
{
    SinglyLinkedList<string> *system1 = new SinglyLinkedList<string>();
    system1->addNewFlight();
    system1->reserveTicket(1);
    system1->reserveTicket(1);
    return 0;
}
```

*Output:*



```
^C
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\FAST_KHI_Se
KHI_Semester_3\DS_lab\03_lab\solution\"3_task
New flight added1
Enter Name:
Mufeez
Mufeez
Mufeez
Passenger added successfully
Enter Name:
Ali
Ali
Ali
Passenger added successfully

F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>
```

**//This print name three times , I placed cout for debugging forgot to remove them**

***Q4:***

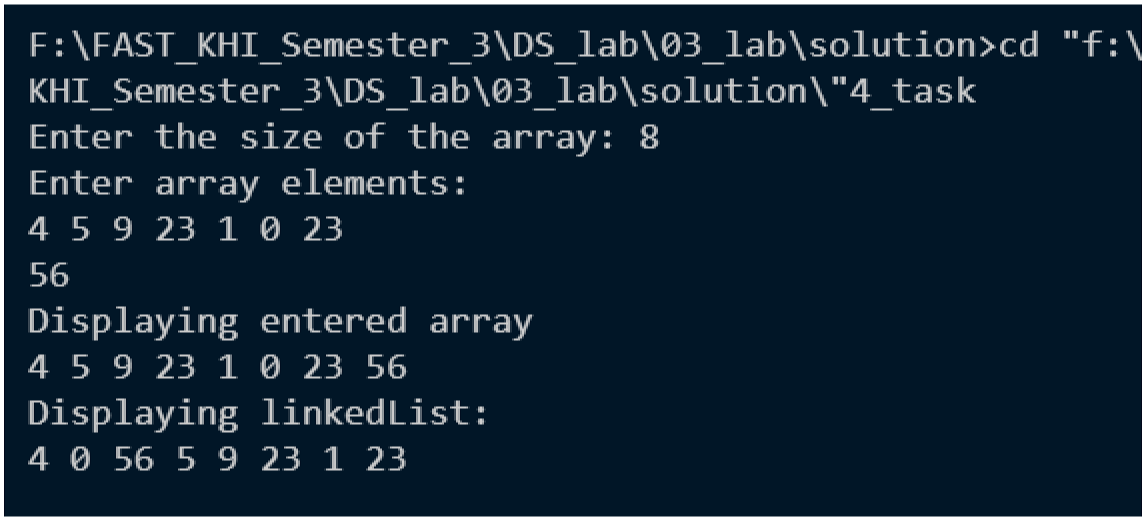
*Code:*

```
#include <iostream>

using namespace std;

class Node
{
private:
    int data;
    Node *next;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    int getData()
```

```cpp
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class SinglyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    void turnArrToLL(int *arr, int size)
    {
        head = new Node(arr[0]);
        Node *temp = head;
        for (int i = 1; i < size; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
        delete arr;
        tail = temp;
    }
    void insertAtFront(int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
        }
        else
        {
            n->setNext(head);
            head = n;
        }
    }
    void sortEvenFirst()
    {
        Node *evenHead = nullptr;
        Node *oddHead = nullptr;
        Node *evenTail = nullptr;
        Node *oddTail = nullptr;
        if (head == NULL || head->getNext() == NULL)
        {
            return;
        }
        Node *curr = head;
        while (curr != NULL)
        {
            if (curr->getData() % 2 == 0)
            {
                if (evenHead == NULL)
                {
                    evenHead = evenTail = curr;
                }
                else
                {
                    evenTail->setNext(curr);
                    evenTail = evenTail->getNext();
                }
            }
            else
            {
                if (oddHead == NULL)
                {
                    oddHead = oddTail = curr;
                }
                else
                {
                    oddTail->setNext(curr);
                    oddTail = oddTail->getNext();
                }
            }
            curr = curr->getNext();
        }
        if (evenHead == NULL)
        {
            head = oddHead;
        }
        if (evenTail)
            evenTail->setNext(nullptr);
        if (oddTail)
            oddTail->setNext(nullptr);

        evenTail->setNext(oddHead);
        head = evenHead;
    }
    void displayLinkedList()
```

```cpp
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }

    ~SinglyLinkedList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
    }
};

int main()
{
    int size;
    cout << "Enter the size of the array: ";
    cin >> size;
    int *arr = new int[size];
    cout << "Enter array elements: " << endl;
    for (int i = 0; i < size; ++i)
    {
        cin >> arr[i];
    }
    cout << "Displaying entered array" << endl;
    for (int i = 0; i < size; ++i)
    {
        cout << arr[i] << " ";
    }
    cout << endl;

    SinglyLinkedList *l1 = new SinglyLinkedList();
    l1->turnArrToLL(arr, size);
    l1->sortEvenFirst();
    l1->displayLinkedList();
    return 0;
}
```

*Output:*



```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\
KHI_Semester_3\DS_lab\03_lab\solution\"4_task
Enter the size of the array: 8
Enter array elements:
4 5 9 23 1 0 23
56
Displaying entered array
4 5 9 23 1 0 23 56
Displaying linkedList:
4 0 56 5 9 23 1 23
```

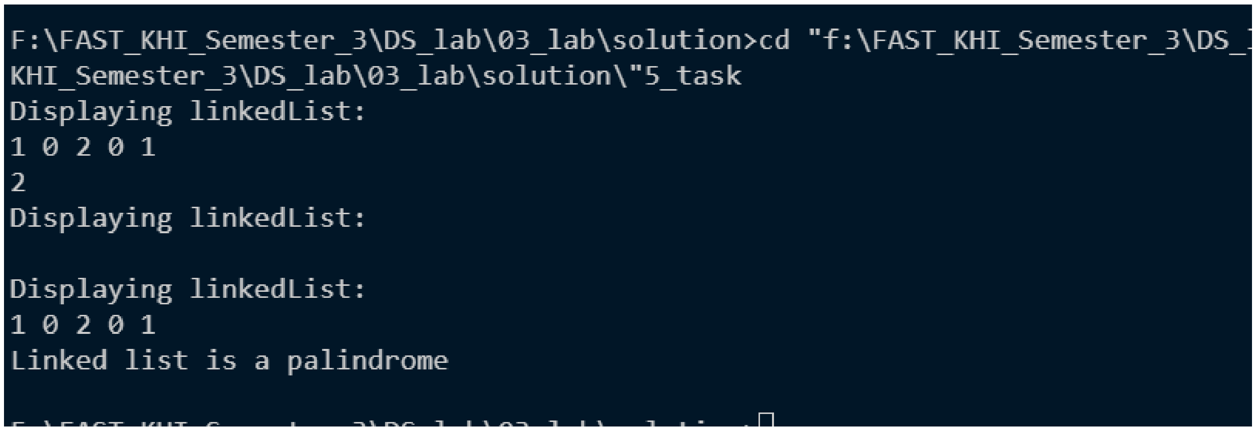
***Q5:***

*Code:*

```cpp
#include <iostream>

using namespace std;
template <typename T>
class Node
{
public:
    T data;
    Node<T> *next;

public:
    Node(T data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node<T> *next)
    {
        this->next = next;
    }
    Node<T> *getNext()
    {
        return next;
    }
    T getData()
    {
        return data;
    }
    void setData(T data)
    {
        this->data = data;
    }
```

```cpp
};
template <typename T>
class SinglyLinkedList
{
public:
    Node<T> *head;
    Node<T> *tail;

public:
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void setHead(Node<T> *head)
    {
        this->head = head;
    }
    void setTail(Node<T> *tail)
    {
        this->tail = tail;
    }
    Node<T> *getHead()
    {
        return head;
    }
    Node<T> *getTail()
    {
        return tail;
    }
    void turnArrToLL(int arr[], int size)
    {
        head = new Node<T>(arr[0]);
        Node<T> *temp = head;
        for (int i = 1; i < size; ++i)
        {
            Node<T> *n = new Node<T>(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
        tail = temp;
    }
    void displayLinkedList()
    {
        Node<T> *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }
    Node<T> *reverse(Node<T> *link)
    {
        if (link == NULL || link->getNext() == NULL)
        {
            return link;
        }
        Node<T> *curr = link;
        Node<T> *prev = NULL;
        Node<T> *next;
        while (!curr)
        {
            next = curr->getNext();
            curr->setNext(prev);
            prev = curr;
            curr = next;
        }
        // below four lines are to debug to check whether the link list has been reversed or
not
        curr = head;
        head = prev;
        displayLinkedList();
        head = curr;
        displayLinkedList();
        return prev;
    }
    void checkPalindrome()
    {
        Node<T> *slow = head, *fast = head;
        while (fast != NULL && fast->getNext() != NULL)
        {
            slow = slow->getNext();
            fast = fast->getNext()->getNext();
        }
        cout << slow->getData() << endl;
        Node<T> *rev = reverse(slow->getNext());
        Node<T> *temp = head;
        bool flag = true;
        while (rev != NULL && temp != rev)
        {
            if (rev != temp)
            {
                flag = false;
                break;
            }
            rev = rev->getNext();
            temp = temp->getNext();
        }
        if (flag)
            cout << "Linked list is a palindrome" << endl;
        else
            cout << "Linked list is not a palindrome" << endl;
    }
    ~SinglyLinkedList()
    {
        Node<T> *temp = head;
        Node<T> *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
    }
};

int main()
{
    int arr[] = {1, 0, 2, 0, 1};
    int size = 5;
    SinglyLinkedList<int> *list = new SinglyLinkedList<int>();
    list->turnArrToLL(arr, size);
    list->displayLinkedList();
    list->checkPalindrome();
    return 0;
}
```

*Output:*



```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\FAST_KHI_Semester_3\DS_
KHI_Semester_3\DS_lab\03_lab\solution\"5_task
Displaying linkedList:
1 0 2 0 1
2
Displaying linkedList:

Displaying linkedList:
1 0 2 0 1
Linked list is a palindrome

F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>
```

***Q6:***

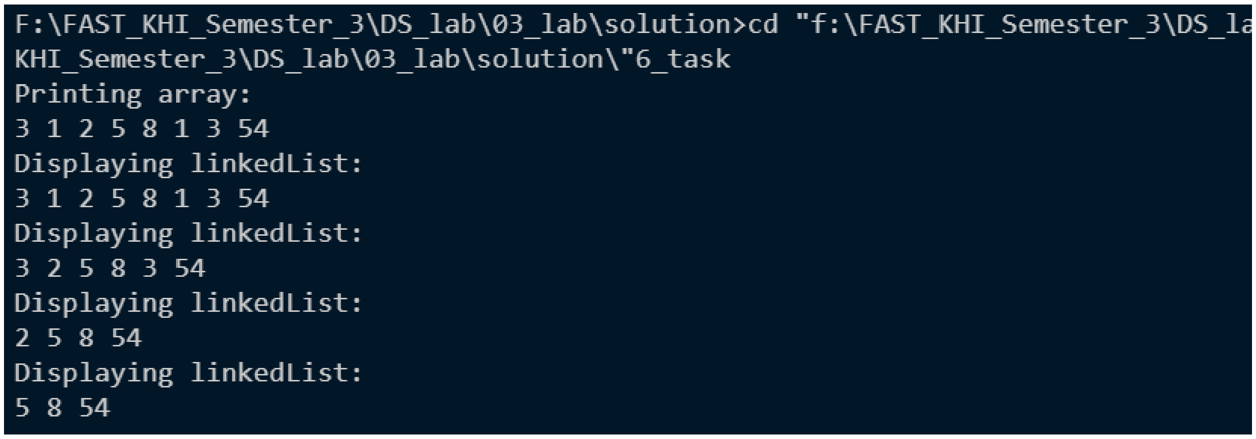
*Code:*

```
#include <bits/stdc++.h>

using namespace std;
//DONE
class Node
{
private:
    int data;
    Node *next;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    int getData()
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class SinglyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void turnArrToLL(int arr[], int size)
    {
        head = new Node(arr[0]);
        Node *temp = head;
        for (int i = 1; i < size; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
```

```cpp
        tail = temp;
    }
    void removeElements(int val)
    {
        Node *temp = head, *prev = NULL;
        if(temp->getData()==val){
            prev = head;
            head = head->getNext();
            temp = head;
            delete prev;
        }
        while (temp != NULL)
        {
            if (temp->getData() == val)
            {
                // Node *temp1 = temp;
                prev->setNext(temp->getNext());
                prev= temp->getNext();
                delete temp;
                temp = prev;
                // delete temp1;
            }
            else
            {
                prev = temp;
                temp = temp->getNext();
            }
        }
            displayLinkedList();
    }
    void displayLinkedList()
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }
    ~SinglyLinkedList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
    }
};
```

```
int main()
{
    int size = 8;
    int arr[size] = {3, 1, 2, 5, 8, 1, 3, 54};
    cout << "Printing array: " << endl;
    for (int i = 0; i < size; ++i)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
    SinglyLinkedList *list1 = new SinglyLinkedList();
    list1->turnArrToLL(arr, size);
    list1->displayLinkedList();
    list1->removeElements(1);
    list1->removeElements(3);
    list1->removeElements(2);

    return 0;
}
```

*Output:*



```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\FAST_KHI_Semester_3\DS_la
KHI_Semester_3\DS_lab\03_lab\solution\"6_task
Printing array:
3 1 2 5 8 1 3 54
Displaying linkedList:
3 1 2 5 8 1 3 54
Displaying linkedList:
3 2 5 8 3 54
Displaying linkedList:
2 5 8 54
Displaying linkedList:
5 8 54
```

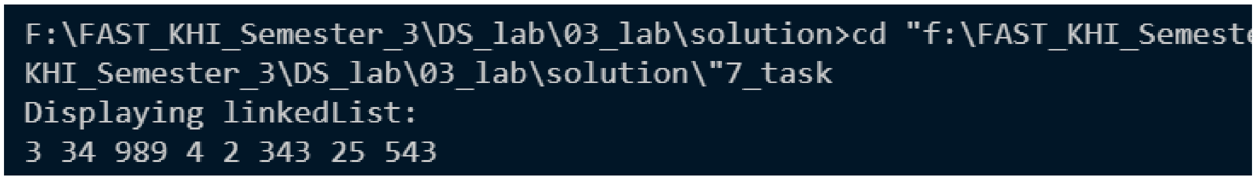
***Q7:***

*Code:*

```
#include <iostream>

using namespace std;

class Node
{
private:
    int data;
    Node *next;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    int getData()
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class circularList
{
private:
    Node *head;
    Node *tail;

public:

    circularList()
    {
        head = NULL;
        tail = NULL;
    }
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    void turnArrToLL(int arr[])
    {
        head = new Node(arr[0]);
        Node *temp = head;
        for (int i = 1; i < 7; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
```

```cpp
        tail = temp;
        tail->setNext(head);
    }

    void insertAtFront(int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
            tail->setNext(head);
        }
        else
        {
            n->setNext(head);
            head = n;
            tail->setNext(head);
        }
    }
    void removeElements(int val)
    {
        Node *temp = head;
        while (temp->getNext()->getData() != val)
        {
            temp = temp->getNext();
        }
        Node *temp1 = temp->getNext();
        temp->setNext(temp->getNext()->getNext());
        delete temp1;
    }
    void insertAtTail(int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
            tail->setNext(head);
        }
        else
        {
            tail->setNext(n);
            tail = tail->getNext();
            tail->setNext(head);
        }
    }
    void displayLinkedList()
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp->getNext() != head)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }
    void insertAtPos(int pos, int val)
    {
        Node *n = new Node(val);
        if (head == NULL)
        {
            head = n;
            tail = n;
            tail->setNext(head);
        }
        else
        {
            int i = 1;
            Node *temp = head;
            while (i != pos - 1)
            {
                temp = temp->getNext();
                ++i;
            }
            n->setNext(temp->getNext());
            temp->setNext(n);
        }
    }

    ~circularList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
    }
};

int main()
{
    circularList *cl = new circularList();
    int arr[] = {34,4,2,55,343,25,543};
    cl->turnArrToLL(arr);
    cl->insertAtTail(43);
    cl->insertAtFront(3);
    cl->insertAtPos(3,989);
    cl->removeElements(55);
    cl->displayLinkedList();
    return 0;
}
```

*Output:*



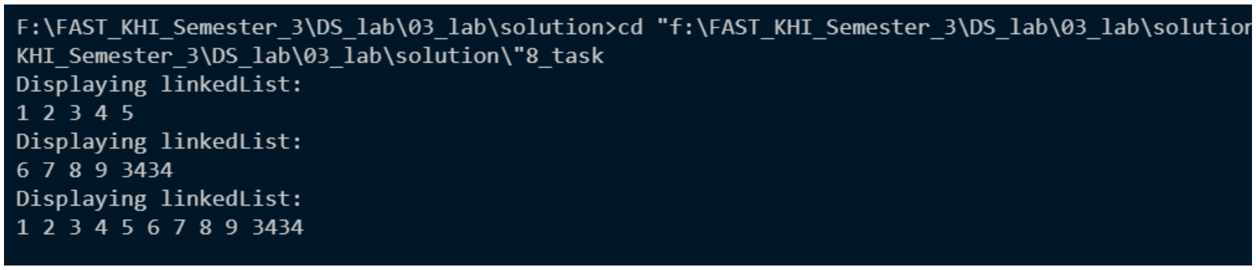
```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\FAST_KHI_Semeste
KHI_Semester_3\DS_lab\03_lab\solution\"7_task
Displaying linkedList:
3 34 989 4 2 343 25 543
```

***Q8:***

*Code:*

```cpp
#include <bits/stdc++.h>

using namespace std;
class Node
{
private:
    int data;
    Node *next;
    Node *prev;

public:
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
        this->prev = NULL;
    }

    void setNext(Node *next)
    {
        this->next = next;
    }
    Node *getNext()
    {
        return next;
    }
    void setPrev(Node *prev)
    {
        this->prev = prev;
    }
    Node *getPrev()
    {
        return prev;
    }
    int getData()
    {
        return data;
    }
    void setData(int data)
    {
        this->data = data;
    }
};

class DoublyLinkedList
{
private:
    Node *head;
    Node *tail;

public:
    void setHead(Node *head)
    {
        this->head = head;
    }
    void setTail(Node *tail)
    {
        this->tail = tail;
    }
    Node *getHead()
    {
        return head;
    }
    Node *getTail()
    {
        return tail;
    }
    DoublyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void turnArrToLL(int arr[], int size)
    {
        head = new Node(arr[0]);
        Node *temp = head;

        Node *prev = NULL;
        for (int i = 1; i < size; ++i)
        {
            Node *n = new Node(arr[i]);
            temp->setNext(n);
            temp->setPrev(prev);
            prev = temp;
            temp = temp->getNext();
        }
        tail = temp;
        displayLinkedList();
    }
    DoublyLinkedList *concatenateLL(DoublyLinkedList *M)
    {
        if (M == NULL)
        {
            return this;
        }
        Node *Mhead = M->getHead();
        Node *Mtail = M->getTail();
        Node *lmover = head;
        Node *Mmover = Mhead;
        while (Mmover != NULL)
        {
            int val = Mmover->getData();
            Node *n = new Node(val), *prev = NULL;
            if (head == NULL)
            {
                head = n;
                tail = n;
            }
            else
            {
                tail->setNext(n);
                prev = tail;
                tail = tail->getNext();
                tail->setPrev(prev);
            }
            Mmover = Mmover->getNext();
        }
        return this;
    }
    void displayLinkedList()
    {
        Node *temp = head;
        cout << "Displaying linkedList: " << endl;
        while (temp != NULL)
        {
            cout << temp->getData() << " ";
            temp = temp->getNext();
        }
        cout << endl;
    }
    ~DoublyLinkedList()
    {
        Node *temp = head;
        Node *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
    }
};

int main()
{
    DoublyLinkedList *L = new DoublyLinkedList();
    DoublyLinkedList *M = new DoublyLinkedList();
    int arr[] = {1, 2, 3, 4, 5};
    int arr1[] = {6, 7, 8, 9, 3434};
    L->turnArrToLL(arr, 5);
    M->turnArrToLL(arr1, 5);
    L->concatenateLL(M);
    L->displayLinkedList();
    return 0;
}
```

*Output:*

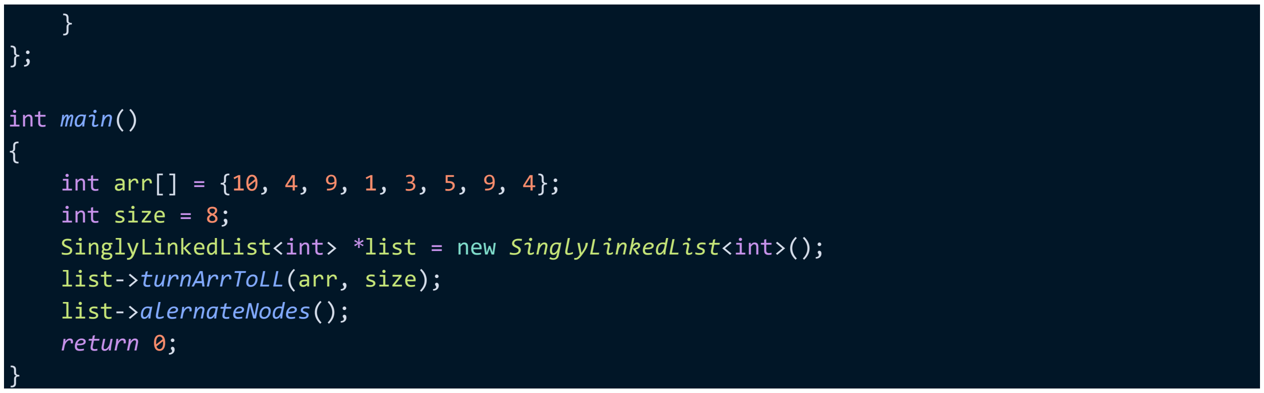


```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solution>cd "f:\FAST_KHI_Semester_3\DS_lab\03_lab\solutior
KHI_Semester_3\DS_lab\03_lab\solution\"8_task
Displaying linkedList:
1 2 3 4 5
Displaying linkedList:
6 7 8 9 3434
Displaying linkedList:
1 2 3 4 5 6 7 8 9 3434
```

***Q9:***

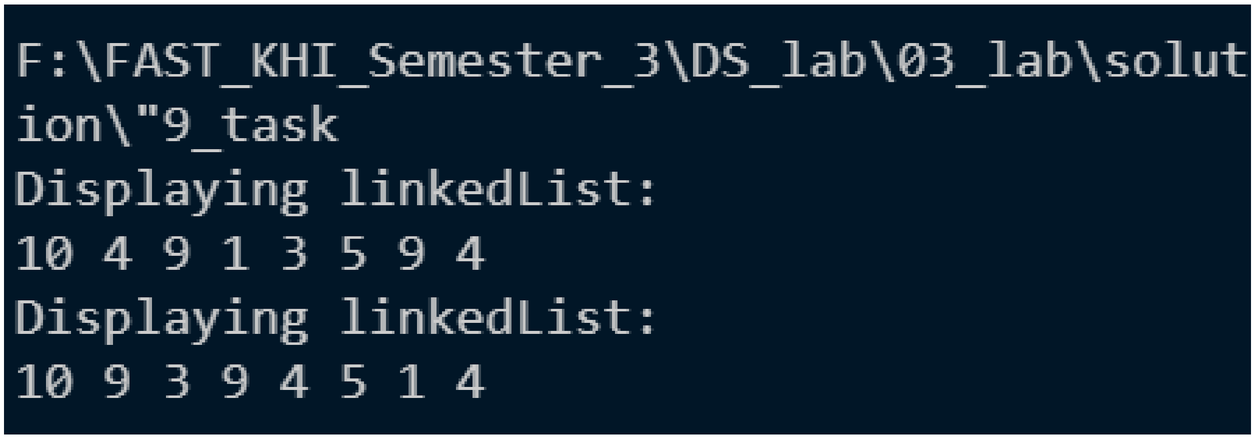
*Code:*

```
#include <iostream>

using namespace std;
template <typename T>
class Node
{
public:
    T data;
    Node<T> *next;

public:
    Node(T data)
    {
        this->data = data;
        this->next = NULL;
    }

    void setNext(Node<T> *next)
    {
        this->next = next;
    }
    Node<T> *getNext()
    {
        return next;
    }
    T getData()
    {
        return data;
    }
    void setData(T data)
    {
        this->data = data;
    }
};
template <typename T>
class SinglyLinkedList
{
public:
    Node<T> *head;
    Node<T> *tail;

public:
    SinglyLinkedList()
    {
        head = NULL;
        tail = NULL;
    }
    void setHead(Node<T> *head)
    {
        this->head = head;
    }
    void setTail(Node<T> *tail)
    {
        this->tail = tail;
    }
    Node<T> *getHead()
    {
        return head;
    }
    Node<T> *getTail()
    {
        return tail;
    }
    void turnArrToLL(int arr[], int size)
    {
        head = new Node<T>(arr[0]);
        Node<T> *temp = head;
        for (int i = 1; i < size; ++i)
        {
            Node<T> *n = new Node<T>(arr[i]);
            temp->setNext(n);
            temp = temp->getNext();
        }
        tail = temp;
        displayLinkedList();
    }
    void displayLinkedList(Node<T> *newHead = NULL)
    {
        if (newHead != NULL)
        {
            Node<T> *temp = newHead;
            cout << "Displaying linkedList: " << endl;
            while (temp != NULL)
            {
                cout << temp->getData() << " ";
                temp = temp->getNext();
            }
            cout << endl;
        }
        else
        {

            Node<T> *temp = head;
            cout << "Displaying linkedList: " << endl;
            while (temp != NULL)
            {
                cout << temp->getData() << " ";
                temp = temp->getNext();
            }
            cout << endl;
        }
    }
    void alernateNodes()
    {
        Node<T> *mover = head, *tempHead = head->getNext(), *alternate = tempHead, *prev =
NULL;

        while (alternate != NULL && alternate->getNext() != NULL)
        {
            mover->setNext(alternate->getNext());
            mover = mover->getNext();
            alternate->setNext(mover->getNext());
            alternate = alternate->getNext();
        }

        alternate = tempHead;
        while (alternate != NULL)
        {
            Node<T> *next = alternate->getNext();
            alternate->setNext(prev);
            prev = alternate;
            alternate = next;
        }
        mover->setNext(prev);
        displayLinkedList();
    }
    ~SinglyLinkedList()
    {
        Node<T> *temp = head;
        Node<T> *n;
        while (temp != NULL)
        {
            n = temp;
            temp = temp->getNext();
            delete n;
        }
        cout << "Deleted" << endl;
```



```
    }
};

int main()
{
    int arr[] = {10, 4, 9, 1, 3, 5, 9, 4};
    int size = 8;
    SinglyLinkedList<int> *list = new SinglyLinkedList<int>();
    list->turnArrToLL(arr, size);
    list->alernateNodes();
    return 0;
}
```

*Output:*



```
F:\FAST_KHI_Semester_3\DS_lab\03_lab\solut
ion\"9_task
Displaying linkedList:
10 4 9 1 3 5 9 4
Displaying linkedList:
10 9 3 9 4 5 1 4
```