DS LAB-3

23K-0634

3a

Q1

#include<iostream>

using namespace std;

class node

{

public:

    int data;

    node\* next;

    node(int val): data(val), next(NULL) {}

};

void insert\_end(node\* &head, int data)

{

    node\* current = head;

    node\* insertNode = new node(data);

    while(current->next != NULL){

        current = current->next;

    }

    current->next = insertNode;

}

void displayLL(node\* head)

{

    node\* current = head;

    while (current != NULL){

        cout<<current->data<<endl;

        current = current->next;

    }

    cout<<endl;

}

void delete\_specific\_data(node\* &head, int data)

{

    node\* current = head;

    node\* prev = NULL;

    if (head == NULL) return;

    if (head->data == data) {  //if head node needs to be deleted

        node\* temp = head;

        head = head->next;

        delete temp;

        return;

    }

    while (current != NULL && current->data != data) {

        prev = current;

        current = current->next;

    }

    if (current == NULL) return;  // If data not found

    prev->next = current->next;

    delete current;

}

node\* insert\_start(node\* &head, int data)

{

    node\* temp = new node(data);

    temp->next = head;

    head = temp;

    return head;

}

void EvenOdd(node\* &head)

{

    node\* current = head;

    node\* nextNode = nullptr;

    while (current != NULL) {

        nextNode = current->next;  //store next node before modifying the list

        if (current->data%2==0) {

            int tempData = current->data;

            delete\_specific\_data(head, current->data);

            head = insert\_start(head, tempData);

        }

        current = nextNode;  //move to the next node

    }

}

int main()

{

    node\* node1 = new node(10);

    node\* head = node1;

    // Constructing linked list

    insert\_end(head, 1);

    insert\_end(head, 2);

    insert\_end(head, 3);

    insert\_end(head, 4);

    insert\_end(head, 5);

    cout << "Original Linked List: "<<endl;

    displayLL(head);

    EvenOdd(head);

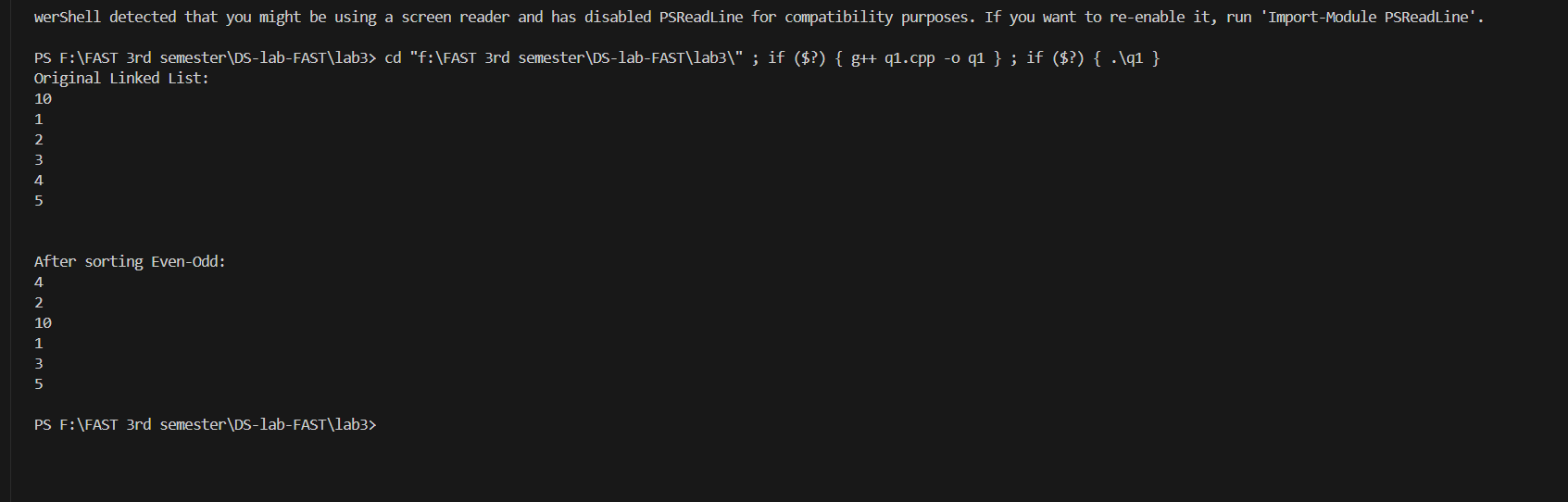
    cout<<endl;

    cout << "After sorting Even-Odd: "<<endl;

    displayLL(head);

    return 0;

}



Q2

#include<iostream>

using namespace std;

class node {

public:

    int data;

    node\* next;

    node(int val) : data(val), next(NULL) {}

};

void insert\_start(node\* &head, int data) {

    node\* newnode = new node(data);

    newnode->next = head;

    head = newnode;

}

void insert\_end(node\* &head, int data) {

    if (head == NULL) {

        head = new node(data);

        return;

    }

    node\* current = head;

    while (current->next != NULL) {

        current = current->next;

    }

    current->next = new node(data);

}

void displayLL(node\* head) {

    node\* current = head;

    while (current != NULL) {

        cout << current->data << " ";

        current = current->next;

    }

    cout << endl;

}

void reversing(node\* &newHead, node\* head) {

    node\* current = head;

    while (current != NULL) {

        insert\_start(newHead, current->data);

        current = current->next;

    }

}

bool checkPalindrome(node\* head) {

    // reverse

    node\* reversedHead = NULL;

    reversing(reversedHead, head);

    // Compare both

    node\* current1 = head;

    node\* current2 = reversedHead;

    while (current1 != NULL && current2 != NULL) {

        if (current1->data != current2->data) {

            return false;

        }

        current1 = current1->next;

        current2 = current2->next;

    }

    return true;

}

int main() {

    int val;

    node\* head = NULL;

    cout << "Enter numbers: (-1 to stop): ";

    cin >> val;

    while (val != -1) {

        insert\_end(head, val);

        cin >> val;

    }

    displayLL(head);

    if (checkPalindrome(head)) {

        cout << "The linked list is a palindrome." << endl;

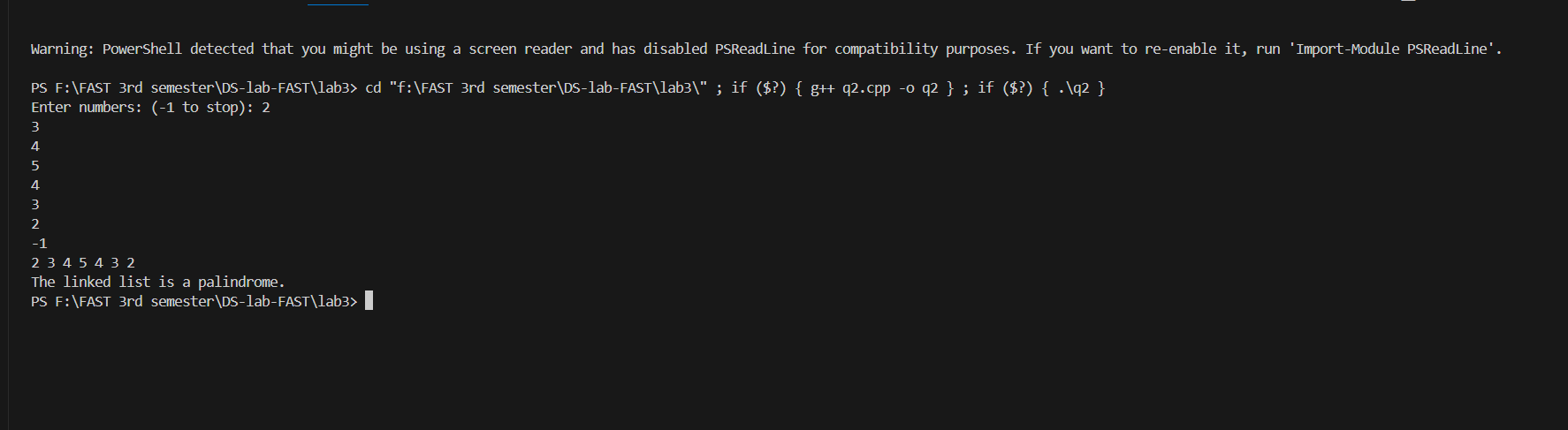
    } else {

        cout << "The linked list is not a palindrome." << endl;

    }

    return 0;

}



Q3

#include <iostream>

using namespace std;

class node

{

    public:

    int data;

    node \*next;

    node(int val):data(val),next(NULL){};

};

void insert\_start(node\* &head, int data)

{

    node\* temp = new node(data);

    node\* current = head;

    while(current->next != head){

        current = current->next;

    }

    temp->next = head;

    head = temp;

    current->next = head;

}

void insert\_end(node\* &head, int data)

{

    node\* current = head;

    node\* temp = new node(data);

    if(current->next == NULL){

        current->next = temp;

        temp->next = head;

    }

    else{

        while(current->next != head){

            current = current->next;

        }

        current->next = temp;

        temp->next = head;

    }

}

void insert\_at\_position(node\* &head, int index, int data)

{

    node\* current = head;

    node\* temp = new node(data);

    int count=1;

    if (index == 0){

        insert\_start(head,data);

    }

    else{

        while(count < index){

            current = current->next;

            count++;

        }

        temp->next = current->next;

        current->next = temp;

    }

}

void delete\_node(node\* head, int index)

{

    node\* current = head;

    node\* temp = NULL;

    int counter = 0;

    while(counter != index){

        temp = current;

        current = current->next;

        counter++;

    }

    temp->next = current->next;

    delete current;

}

void display(node\* head)

{

    node\* current = head;

    cout<<current->data<<endl;

    current = current->next;

    while(current != head){

        cout<<current->data<<endl;

        current = current->next;

    }

}

int main()

{

    node\* node1 = new node(10);

    node\* head = node1;

    insert\_end(head, 20);

    insert\_end(head, 30);

    insert\_end(head, 40);

    insert\_start(head,5);

    insert\_at\_position(head,2,25);

    display(head);

    cout<<endl;

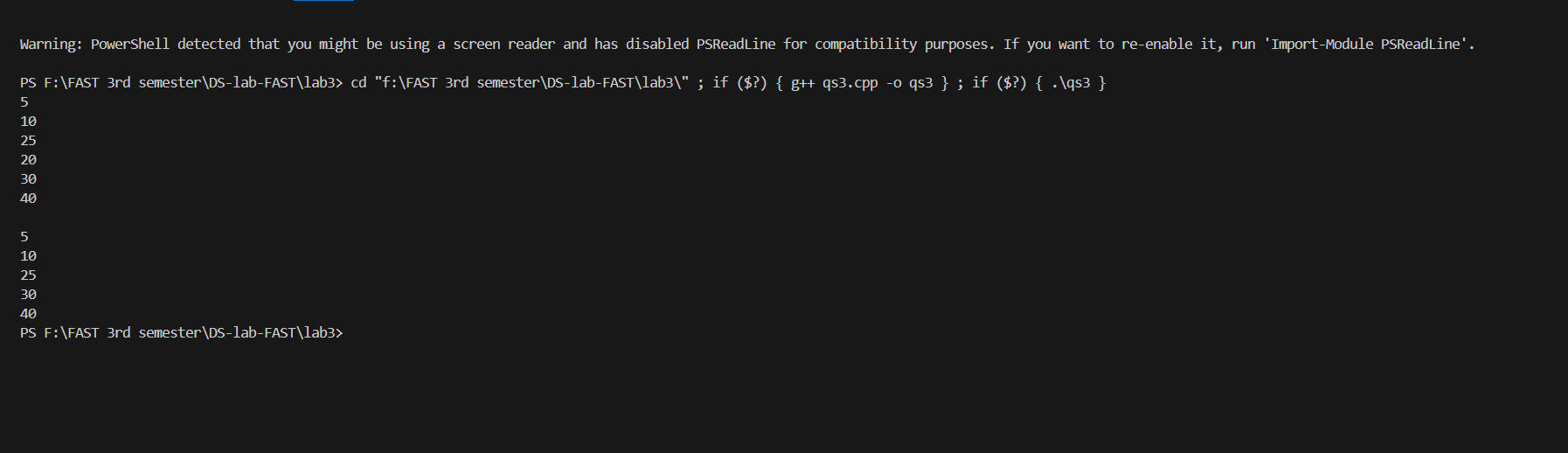
    delete\_node(head,3);

    display(head);

    // can check all functions one by one by uncommenting

    return 0;

}



3b

Q1

#include<iostream>

using namespace std;

class node

{

    public:

    int data;

    node\* next = NULL;

    node\* prev = NULL;

    node(int val):data(val){};

};

void insert\_end(node\* head, int val)

{

    node\* temp = new node(val);

    node\* current = head;

    if(current->next == NULL){

        current->next = temp;

        temp->prev = current;

        temp->next = head;

    }

    else{

        while(current->next != head){

            current = current->next;

        }

        current->next = temp;

        temp->prev = current;

        temp->next = head;

    }

}

void insert\_start(node\* &head, int data)

{

    node\* temp = new node(data);

    node\* current = head;

    while(current->next != head){

        current = current->next;

    }

    temp->next = head;

    temp->prev = current;

    head = temp;

    current->next = head;

}

void insert\_at\_position(node\* &head, int index, int data)

{

    node\* current = head;

    node\* temp = new node(data);

    int count=1;

    if (index == 0){

        insert\_start(head,data);

    }

    else{

        while(count < index){

            current = current->next;

            count++;

        }

        temp->next = current->next;

        current->next->prev = temp;

        current->next = temp;

        temp->prev = current;

    }

}

void display(node\* head)

{

    node\* current = head;

    cout<<current->data<<endl;

    current = current->next;

    while(current != head){

        cout<<current->data<<endl;

        current = current->next;

    }

}

void delete\_node(node\* head, int index)

{

    node\* current = head;

    node\* temp = NULL;

    int counter = 0;

    while(counter != index){

        temp = current;

        current = current->next;

        counter++;

    }

    temp->next = current->next;

    current->next->prev = temp;

    delete current;

}

int main()

{

    node\* node1 = new node(10);

    node\* head = node1;

    insert\_end(head,20);

    insert\_end(head,30);

    insert\_end(head,40);

    insert\_start(head,5);

    insert\_at\_position(head,3,25);

    display(head);

    cout<<endl;

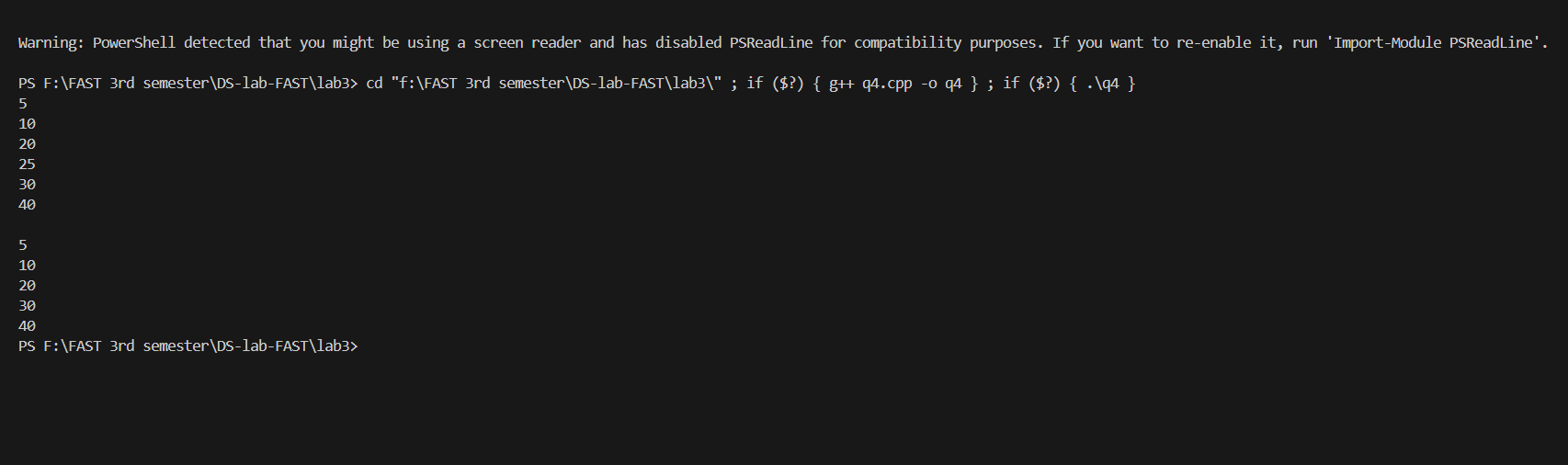
    delete\_node(head,3);

    display(head);

    return 0;

    // can check all functions one by one by uncommenting

}



Q2

#include<iostream>

using namespace std;

class node

{

    public:

    int data;

    node\* next = NULL;

    node\* prev = NULL;

    node(int val):data(val){};

};

void insert(node\* &head, int data)

{

    node\* newnode = new node(data);

    node\* current = head;

    while(current != NULL && current->next != nullptr){

        current = current->next;

    }

    current->next = newnode;

    newnode->prev = current;

}

void display(node\* &head)

{

    node\* current = head;

    while (current!=NULL){

        cout<<current->data<<" ";

        current = current->next;

    }

}

void merge(node\* &Lhead, node\* &Mhead)

{

    node\* current = Lhead;

    while(current->next != NULL){

        current = current->next;

    }

    current->next = Mhead;

}

int main()

{

    // creating LL 'L'

    node\* node1 = new node(10);

    node\* Lhead = node1;

    insert(Lhead,20);

    insert(Lhead,30);

    insert(Lhead,40);

    // creating LL 'M'

    node\* node2 = new node(50);

    node\* Mhead = node2;

    insert(Mhead,60);

    insert(Mhead,70);

    insert(Mhead,80);

    cout<<"Linked List L:"<<endl;

    display(Lhead);

    cout<<endl;

    cout<<"Linked List M:"<<endl;

    display(Mhead);

    cout<<endl;

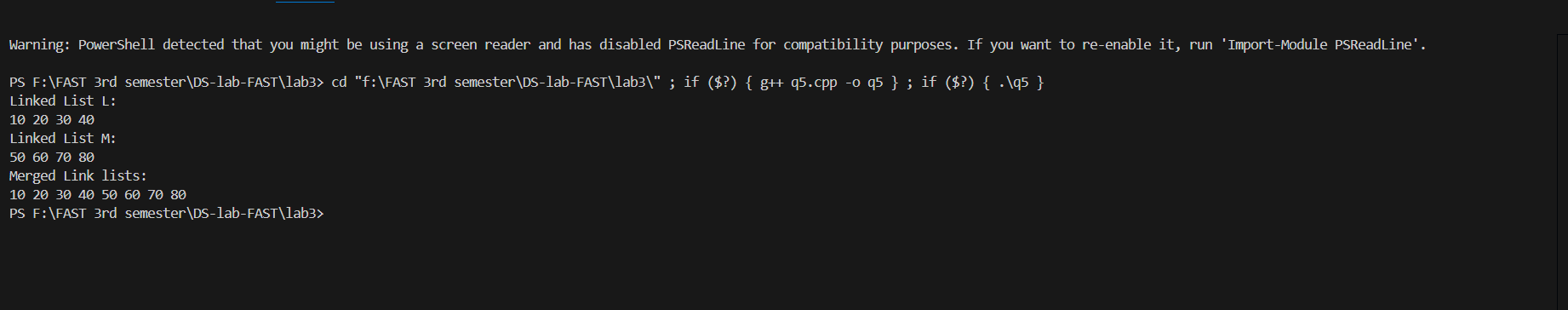
    merge(Lhead,Mhead);

    cout<<"Merged Link lists:"<<endl;

    display(Lhead);

    return 0;

}



Q3

#include<iostream>

using namespace std;

class node

{

    public:

    int data;

    node\* next = NULL;

    node\* prev = NULL;

    node(int val):data(val){};

};

void insert\_end(node\* head, int val)

{

    node\* temp = new node(val);

    node\* current = head;

    if(current->next == NULL){

        current->next = temp;

        temp->prev = current;

        temp->next = head;

    }

    else{

        while(current->next != head){

            current = current->next;

        }

        current->next = temp;

        temp->prev = current;

        temp->next = head;

    }

}

void display(node\* head)

{

    node\* current = head;

    cout<<current->data<<endl;

    current = current->next;

    while(current != head){

        cout<<current->data<<endl;

        current = current->next;

    }

}

void swap(node\* &head, int val1, int val2)

{

    node\* current = head;

    node\* temp1 = NULL;

    node\* temp2 = NULL;

    node\* temp3 = NULL;

    int count = 0;

    while(current->next != head){

        if(count == val1){

            temp1 = current;

        }

        if(count == val2){

            temp2 = current;

        }

        current = current->next;

        count++;

    }

    // Swap next pointers

    temp3 = temp1->next;

    temp1->next = temp2->next;

    temp2->next = temp3;

    // Swap prev pointers

    temp3 = temp1->prev;

    temp1->prev = temp2->prev;

    temp2->prev = temp3;

    if (temp1->next != nullptr) temp1->next->prev = temp1;

    if (temp2->next != nullptr) temp2->next->prev = temp2;

    if (temp1->prev != nullptr) temp1->prev->next = temp1;

    if (temp2->prev != nullptr) temp2->prev->next = temp2;

}

int main()

{

    node\* node1 = new node(1);

    node\* head = node1;

    insert\_end(head,7);

    insert\_end(head,4);

    insert\_end(head,2);

    insert\_end(head,6);

    insert\_end(head,4);

    insert\_end(head,5);

    insert\_end(head,3);

    insert\_end(head,9);

    insert\_end(head,8);

    int num1,num2;

    cout<<"Enter two nodes keys: ";

    cin>>num1>>num2;

    cout<<"Initial linked list = ";

    display(head);

    cout<<endl;

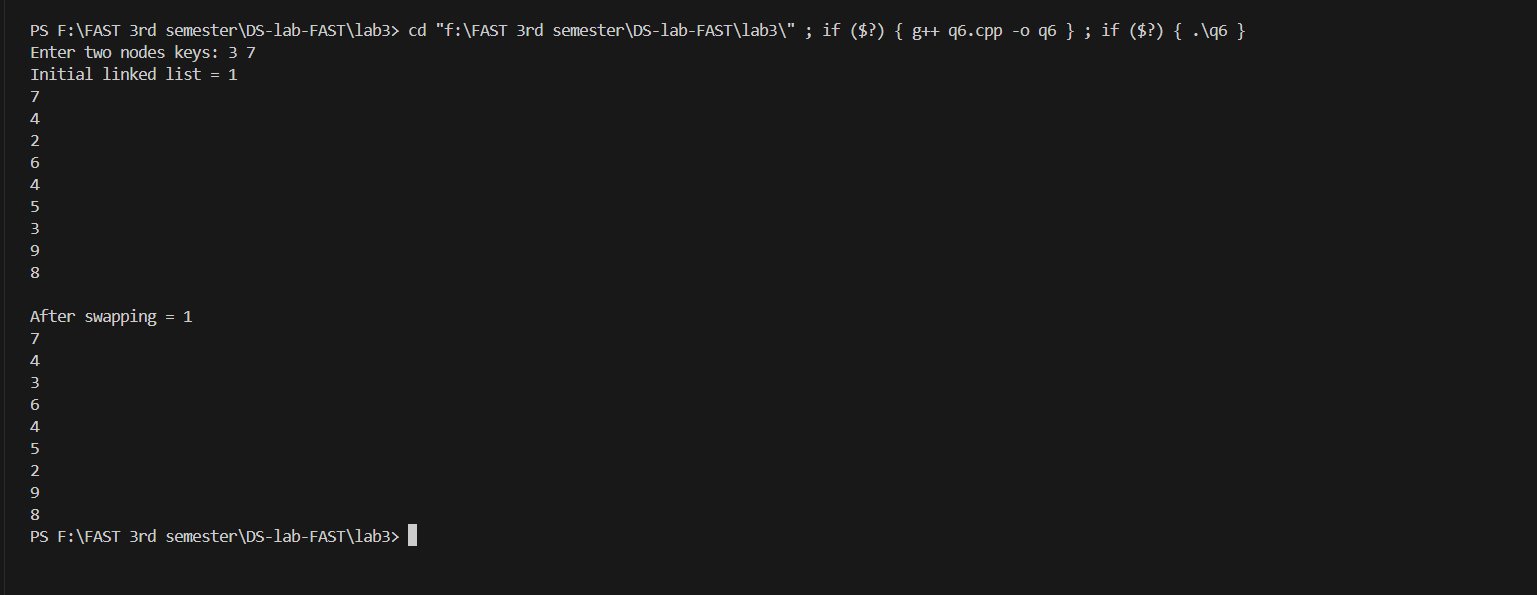
    cout<<"After swapping = ";

    swap(head,num1,num2);

    display(head);

    return 0;

}



Q4

#include<iostream>

using namespace std;

class node

{

    public:

    int data;

    node\* next = NULL;

    node\* prev = NULL;

    node(int data){

        this->data = data;

    }

};

void add\_at\_end(node\* &head, int data)

{

    node\* newnode = new node(data);

    node\* current = head;

    while(current != NULL && current->next != nullptr){

        current = current->next;

    }

    current->next = newnode;

    newnode->prev = current;

}

void displayLL(node\* &head)

{

    node\* current = head;

    while (current!=NULL){

        cout<<current->data<<" ";

        current = current->next;

    }

}

void delete\_index(node\* &head,int index)

{

    node\* current = head;

    int count = 0;

    while (current != nullptr && count < index) {

        current = current->next;

        count++;

    }

    if (current->prev != nullptr) {

        current->prev->next = current->next;

    }

    // Update the prev pointer of the next node

    if (current->next != nullptr) {

        current->next->prev = current->prev;

    }

    // Free the memory of the node to be deleted

    delete current;

}

void alternatives(node\* head, node\* &alt\_head)

{

    node\* current = head;

    int alt = 0;

    node\* alt\_current = alt\_head;

    while(current != NULL){

        if (alt % 2 != 0 && alt != 1){

            add\_at\_end(alt\_head,current->data);

        }

        current = current->next;

        alt++;

    }

    current = head;

    alt = 0;

    //deleting odd indexes..

    while (current != nullptr){

        node\* next\_node = current->next;

        if (alt % 2 != 0){

            if (current->prev != nullptr) {

                current->prev->next = current->next;

            }

            if (current->next != nullptr) {

                current->next->prev = current->prev;

            }

            delete current;

        }

        current = next\_node;

        alt++;

    }

}

void reversing(node\* &alt\_head)

{

    node\* current = alt\_head;

    node\* temp = NULL;

    while (current != nullptr) {

        temp = current->prev;       //swapping

        current->prev = current->next;

        current->next = temp;

        current = current->prev;

    }

    if (temp != nullptr) {

        alt\_head = temp->prev;

    }

}

void append(node\* &head, node\* &alt\_head)

{

    node\* current = head;

    while(current->next != NULL){

        current = current->next;

    }

    current->next = alt\_head;

}

int main()

{

    int val,num;

    cout<<"Enter number of nodes: ";

    cin>>num;

    cout<<"Enter node 1: ";

    cin>>val;

    node\* node1 = new node(val);

    node\* head = node1;

    for (int i=1;i<num;i++){

        cout<<"Enter node "<<i+1<<" :";

        cin>>val;

        add\_at\_end(head,val);

    }

    cout<<"Linked list = ";

    displayLL(head);

    cout<<endl;

    node\* alt\_node = new node(head->next->data);

    node\* alt\_head = alt\_node;

    //alternating, reversing the alt LL, and appending it back, using 3 seperate functions.

    alternatives(head,alt\_head);

    reversing(alt\_head);

    append(head, alt\_head);

    cout<<"Final Linked List: ";

    displayLL(head);

    return 0;

}

