Name:

Wajiha Zahid

Roll No:

S24-040

Subject:

"DSA LAB "

Section

BSSE-3A

Resource Person:

Sir Rasikh Ali

(Lab Task 7)

Q1: Doubly Linked List (Insert & Display Nodes)

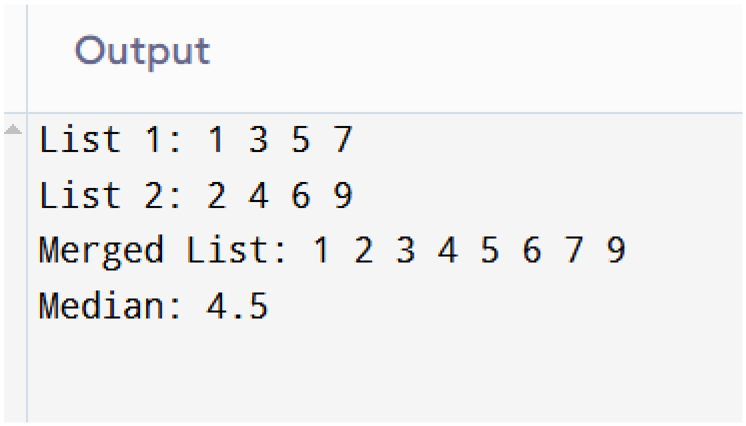
Task: Implement functions to insert node at first, last, Nth location, and centre of a doubly linked list. And display in order and display in reverse order.

Answer:

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* prev;
    Node* next;
    Node(int d) : data(d), prev(nullptr), next(nullptr) {}
};

class DoublyLinkedList {
    Node* head;
    Node* tail;
public:
    DoublyLinkedList() : head(nullptr), tail(nullptr) {}

    void insertLast(int data) {
        Node* newNode = new Node(data);
        if (!head) head = tail = newNode;
        else { tail->next = newNode; newNode->prev = tail; tail = newNode; }
    }

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

    static DoublyLinkedList* mergeLists(DoublyLinkedList* list1,
DoublyLinkedList* list2) {
        DoublyLinkedList* mergedList = new DoublyLinkedList();
        Node *n1 = list1->head, *n2 = list2->head;
        while (n1 || n2) {
            if (!n2 || (n1 && n1->data <= n2->data)) { mergedList-
>insertLast(n1->data); n1 = n1->next; }
            else { mergedList->insertLast(n2->data); n2 = n2->next; }
        }
        return mergedList;
    }

    double findMedian() {
        if (!head) return 0;
        Node* slow = head, *fast = head;
        while (fast && fast->next) { slow = slow->next; fast = fast->next->next;
}
        return (fast ? slow->data : (slow->data + slow->prev->data) / 2.0);
    }
};

int main() {
    DoublyLinkedList list1, list2;
    for (int n : {1, 3, 5, 7}) list1.insertLast(n);
    for (int n : {2, 4, 6, 9}) list2.insertLast(n);

    cout << "List 1: "; list1.display();
    cout << "List 2: "; list2.display();

    DoublyLinkedList* mergedList = DoublyLinkedList::mergeLists(&list1,
&list2);
    cout << "Merged List: "; mergedList->display();

    cout << "Median: " << mergedList->findMedian() << endl;
    delete mergedList;
    return 0;
```

```
}
```

Output:



```
List 1: 1 3 5 7
List 2: 2 4 6 9
Merged List: 1 2 3 4 5 6 7 9
Median: 4.5
```