**Name :M Ijaz**

**Submitted to**

**Sir Rasikh**

**Roll no . 073**

**Subject . Lab Data structure**

**Lab .8**

**Question no**

Create 2 Singly LinkedLists and Merge them and display them.

2. Create 2 Double LinkedLists and Merge them and display them. #include <iostream> using namespace std;

class Node { public:

int data;

Node\* next;

Node(int val) : data(val), next(nullptr) {}

};

class SinglyLinkedList { public:

Node\* head;

SinglyLinkedList() : head(nullptr) {}

void insertLast(int val) {

Node\* newNode = new Node(val);

if (!head) { head = newNode;

} else {

Node\* temp = head; while (temp->next) temp = temp->next; temp->next = newNode;

}

}

void display() { Node\* temp = head; while (temp) { cout << temp->data << " "; temp = temp->next;

}

cout << endl;

}

void merge(SinglyLinkedList& list2) {

if (!head) { head = list2.head; return;

}

Node\* temp = head; while (temp->next) temp = temp->next; temp->next = list2.head;

}

};

int main() { SinglyLinkedList list1, list2; list1.insertLast(1); list1.insertLast(2); list1.insertLast(3);

list2.insertLast(4); list2.insertLast(5); list2.insertLast(6);

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

list1.merge(list2);

cout << "Merged List: "; list1.display(); return 0;

}

Part 2 #include <iostream> using namespace std; class Node { public: int data;

Node\* next;

Node\* prev;

Node(int val) : data(val), next(nullptr), prev(nullptr) {}

};

class DoublyLinkedList { public:

Node\* head;

DoublyLinkedList() : head(nullptr) {}

void insertLast(int val) {

Node\* newNode = new Node(val);

if (!head) { head = newNode;

} else {

Node\* temp = head; while (temp->next) temp = temp->next; temp->next = newNode; newNode->prev = temp;

}

}

void display() {

Node\* temp = head;

while (temp) { cout << temp->data << " "; temp = temp->next;

}

cout << endl;

}

void merge(DoublyLinkedList& list2) {

if (!head) { head = list2.head; return;

}

Node\* temp = head; while (temp->next) temp = temp->next; temp->next = list2.head; if (list2.head) list2.head->prev = temp;

}

};

int main() {

DoublyLinkedList list1, list2; list1.insertLast(1); list1.insertLast(2); list1.insertLast(3);

list2.insertLast(4); list2.insertLast(5); list2.insertLast(6);

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

list1.merge(list2);

cout << "Merged List: "; list1.display(); return 0;



}