4)Write C++ programs to implement the dequeue (double ended queue) ADT using a doubly linked list.

Program:

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

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

class Deque {
private:
   Node* front;
   Node* rear;

public:
   Deque() : front(nullptr), rear(nullptr) {}

   // Insert at the front
   void insertFront(int val) {
      Node* newNode = new Node(val);
      if (!front) front = rear = newNode;
      else {
         newNode->next = front;
         front->prev = newNode;
```

```
        front = newNode;
    }
}

// Insert at the rear
void insertRear(int val) {
    Node* newNode = new Node(val);
    if (!rear) front = rear = newNode;
    else {
        newNode->prev = rear;
        rear->next = newNode;
        rear = newNode;
    }
}

// Delete from the front
void deleteFront() {
    if (!front) return;
    Node* temp = front;
    if (front == rear) front = rear = nullptr;
    else {
        front = front->next;
        front->prev = nullptr;
    }
    delete temp;
}

// Delete from the rear
void deleteRear() {
    if (!rear) return;
    Node* temp = rear;
```

```
        if (front == rear) front = rear = nullptr;
        else {
            rear = rear->prev;
            rear->next = nullptr;
        }
        delete temp;
    }

    // Display the deque
    void display() {
        Node* temp = front;
        while (temp) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    Deque dq;
    dq.insertFront(10);
    dq.insertRear(20);
    dq.insertFront(5);
    dq.display();  // Output: 5 10 20
    dq.deleteFront();
    dq.display();  // Output: 10 20
    dq.deleteRear();
    dq.display();  // Output: 10
    return 0;
}
```