Experiment-4

Student Name: Gurveer Singh Mangat **UID:** 23BCS11074
Branch: B.E-C.S.E **Section/Group:** 23KRG-1A
Semester: 5th **Date of Performance:** 25/08/2025
Subject Name: DAA **Subject Code:** 23CSH-301

1. **Aim:** Apply the concept of Linked list and write code to Insert and Delete an element at the beginning and atend in Doubly and Circular Linked List.

2. **Objective:** To understand doubly and circular linked list

3. **Input/Apparatus Used:** Doubly and circular Linked List is used.

4. **Procedure/Algorithm: Pseudocode:**

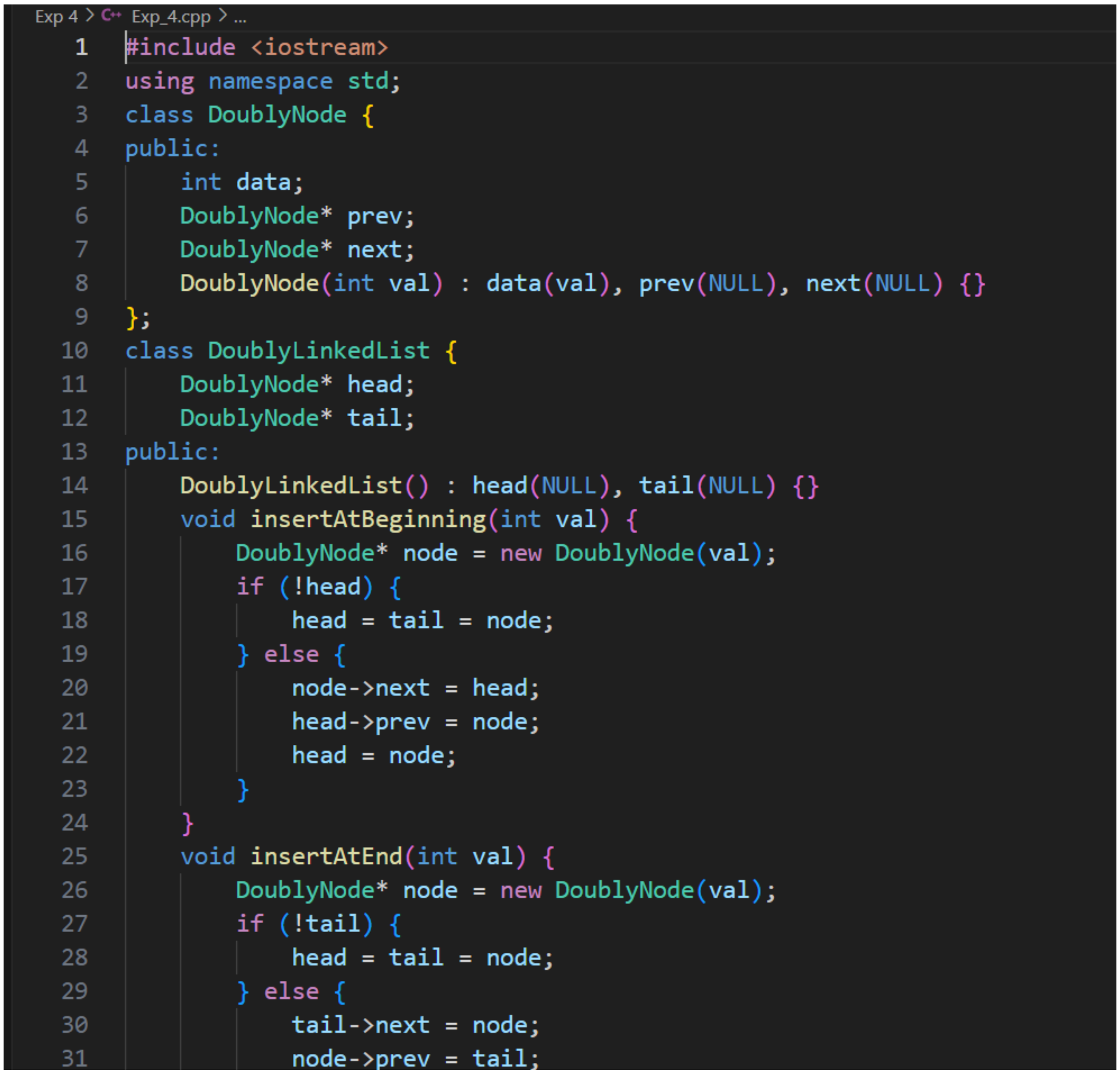
Procedure for beginning of circular linked list:

Step1. Create the new node
Step2. Set the new node‟s next to itself (circular) Step3. If the list is empty,return new node.
Step4. Set our new node‟s next to the front. Step5. Set tail‟s next to our new node.
Step6. Return the end of the list.

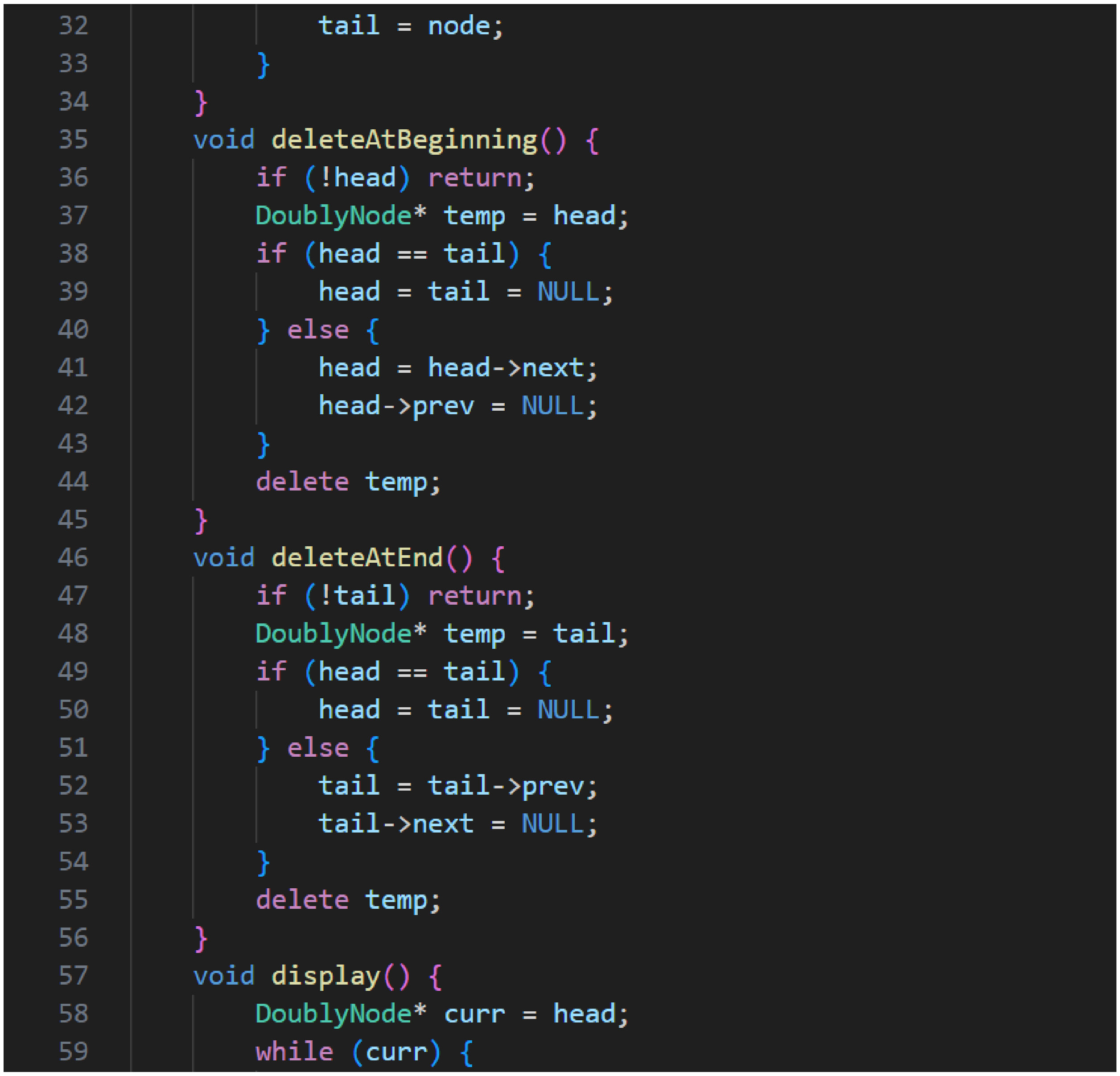
Procedure for end of circular linked list:

Step1. Create the new node
Step2. Set the new node‟s next to itself (circular) Step3. If the list is empty,return new node.
Step4. Set our new node‟s next to the front. Step5. Set tail‟s next to our new node.
Step6. Return the end of the list.

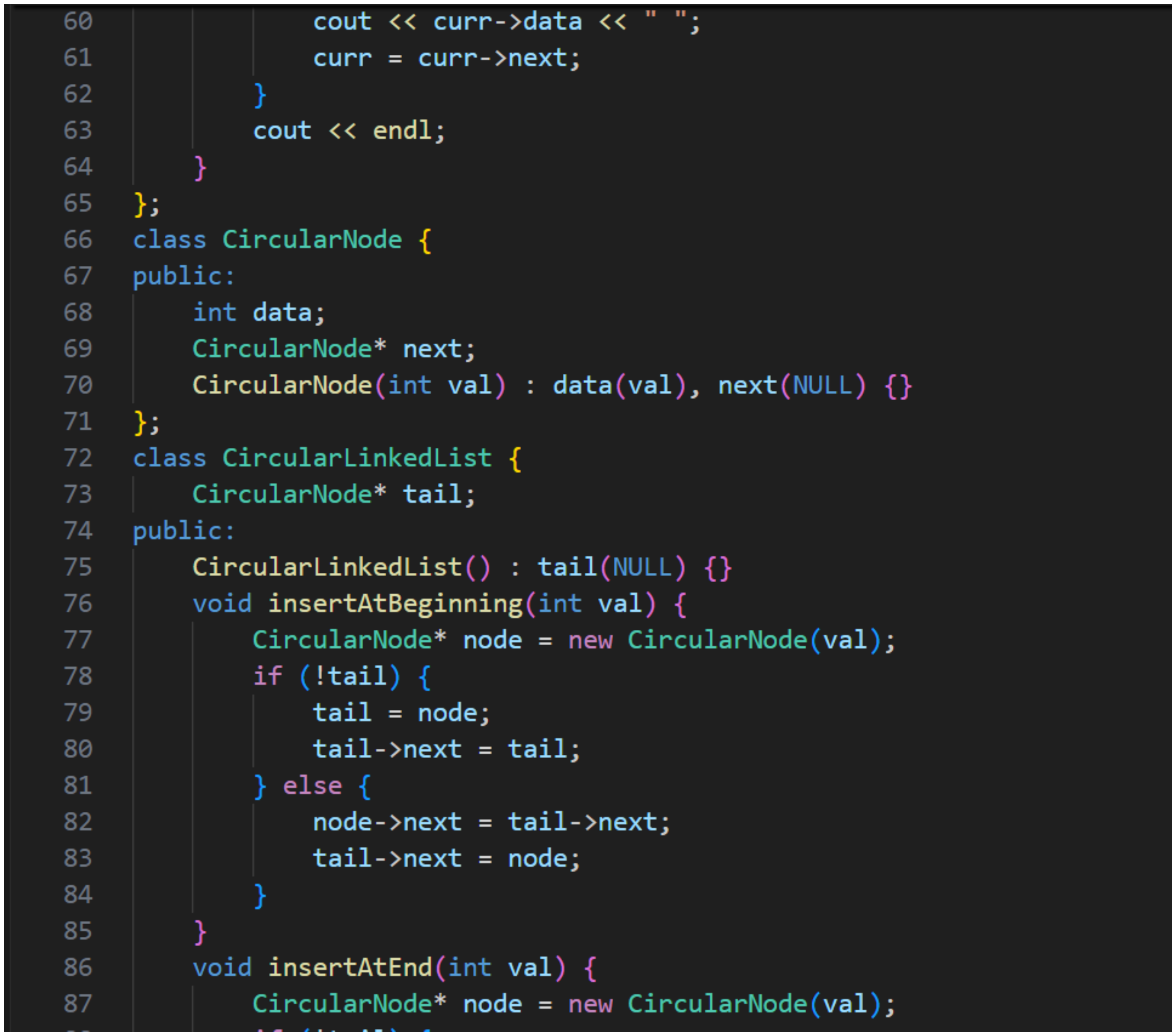
5. **Code:**


Exp 4 > Exp_4.cpp > ...

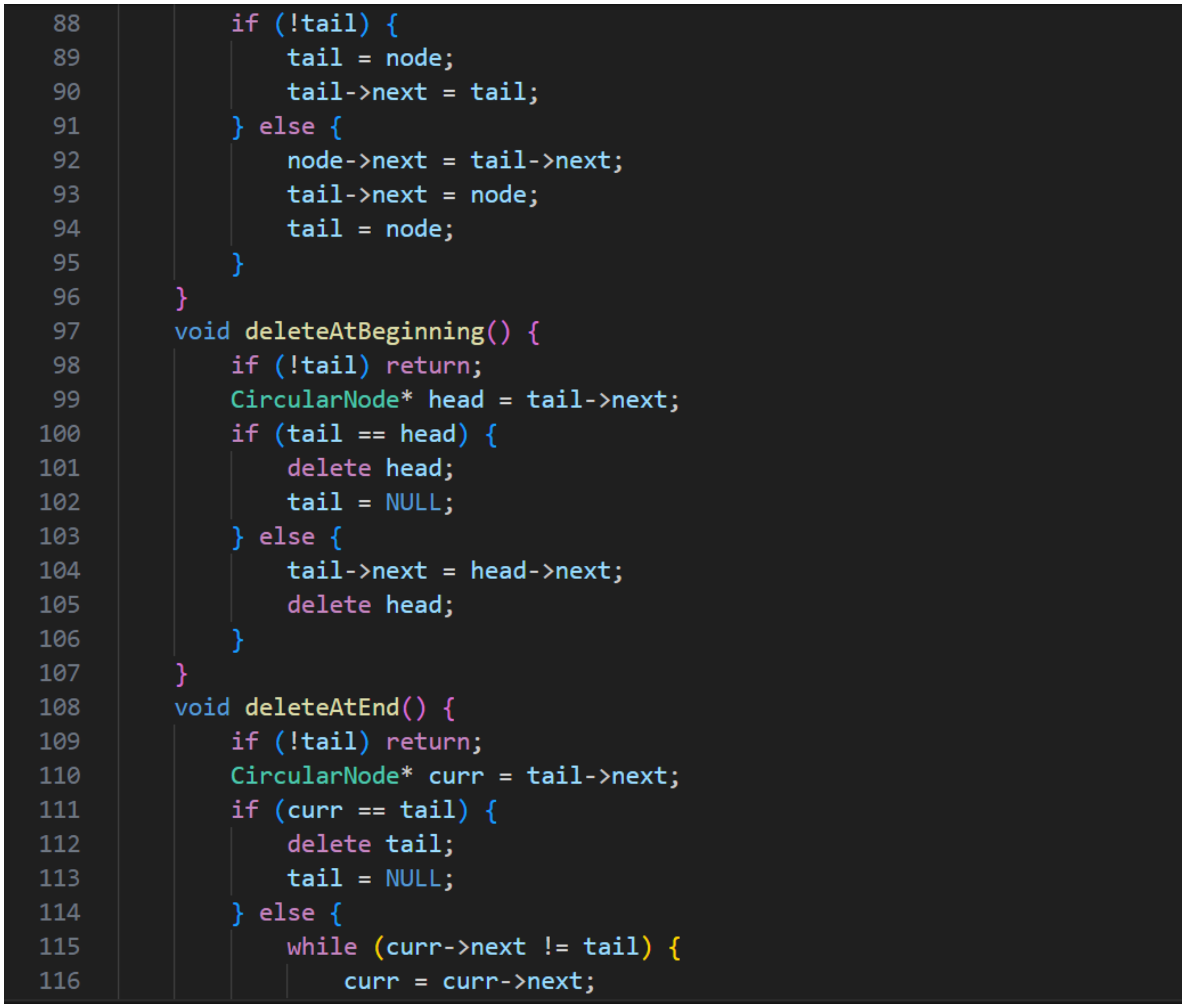
```cpp
#include <iostream>
using namespace std;
class DoublyNode {
public:
    int data;
    DoublyNode* prev;
    DoublyNode* next;
    DoublyNode(int val) : data(val), prev(NULL), next(NULL) {}
};
class DoublyLinkedList {
    DoublyNode* head;
    DoublyNode* tail;
public:
    DoublyLinkedList() : head(NULL), tail(NULL) {}
    void insertAtBeginning(int val) {
        DoublyNode* node = new DoublyNode(val);
        if (!head) {
            head = tail = node;
        } else {
            node->next = head;
            head->prev = node;
            head = node;
        }
    }
    void insertAtEnd(int val) {
        DoublyNode* node = new DoublyNode(val);
        if (!tail) {
            head = tail = node;
        } else {
            tail->next = node;
            node->prev = tail;
```



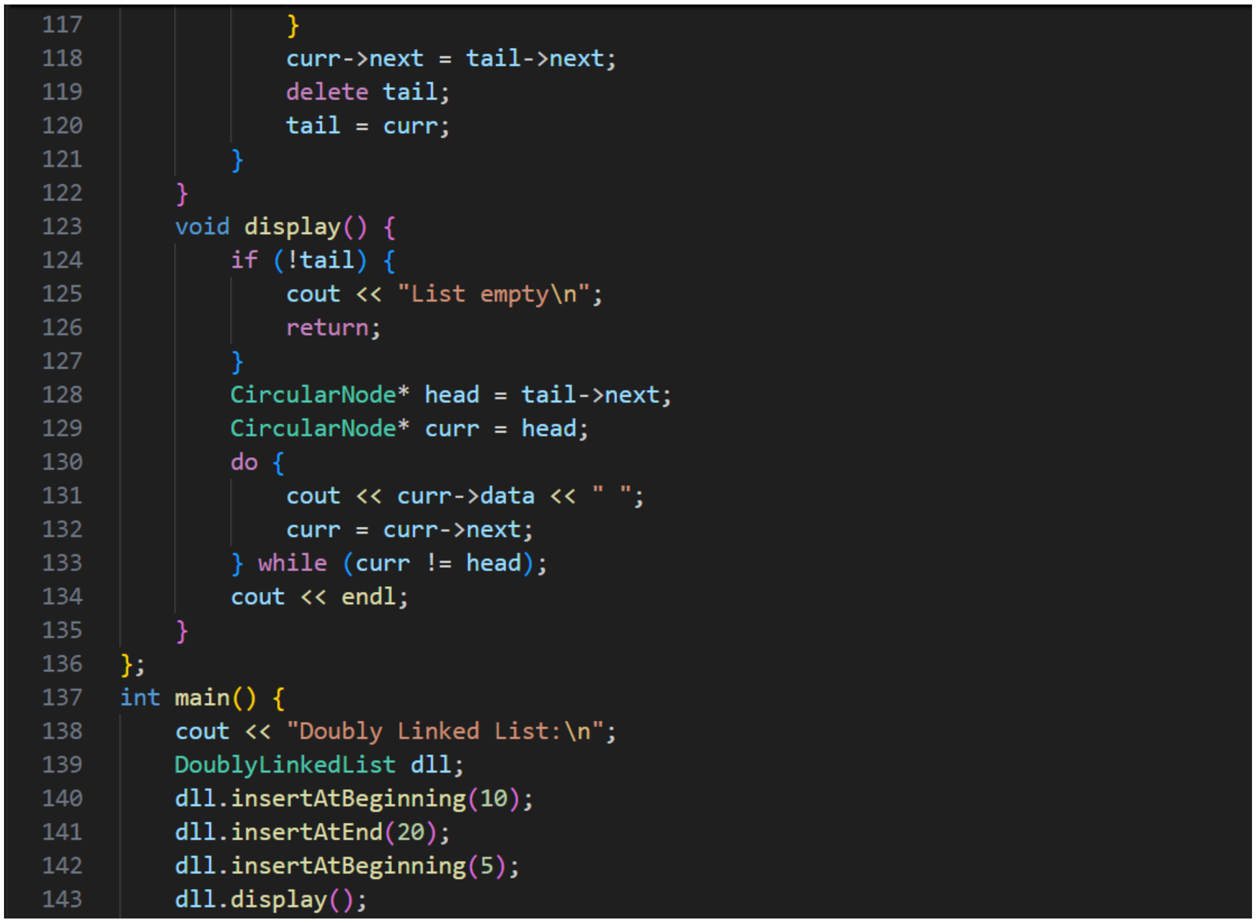
```
            tail = node;
        }
    }
    void deleteAtBeginning() {
        if (!head) return;
        DoublyNode* temp = head;
        if (head == tail) {
            head = tail = NULL;
        } else {
            head = head->next;
            head->prev = NULL;
        }
        delete temp;
    }
    void deleteAtEnd() {
        if (!tail) return;
        DoublyNode* temp = tail;
        if (head == tail) {
            head = tail = NULL;
        } else {
            tail = tail->prev;
            tail->next = NULL;
        }
        delete temp;
    }
    void display() {
        DoublyNode* curr = head;
        while (curr) {
```



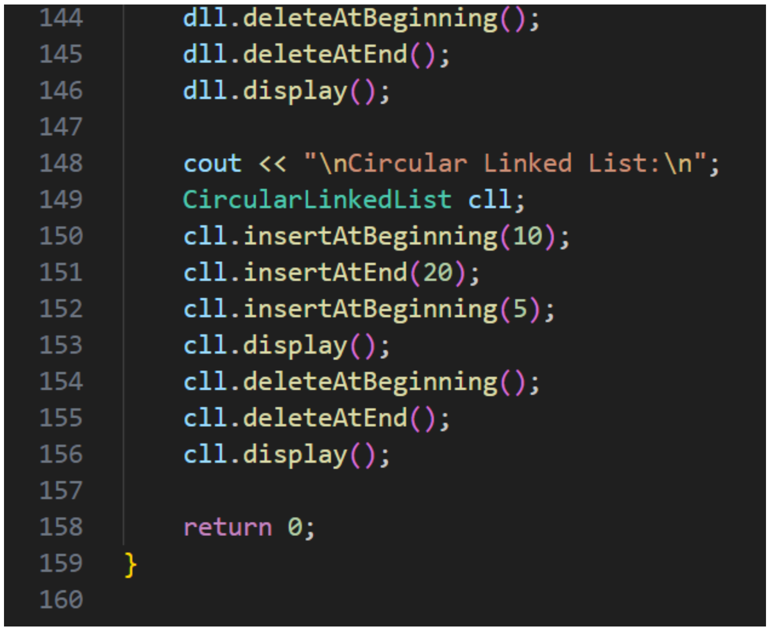
```cpp
                cout << curr->data << " ";
                curr = curr->next;
            }
            cout << endl;
        }
};
class CircularNode {
public:
    int data;
    CircularNode* next;
    CircularNode(int val) : data(val), next(NULL) {}
};
class CircularLinkedList {
    CircularNode* tail;
public:
    CircularLinkedList() : tail(NULL) {}
    void insertAtBeginning(int val) {
        CircularNode* node = new CircularNode(val);
        if (!tail) {
            tail = node;
            tail->next = tail;
        } else {
            node->next = tail->next;
            tail->next = node;
        }
    }
    void insertAtEnd(int val) {
        CircularNode* node = new CircularNode(val);
```



```
        if (!tail) {
            tail = node;
            tail->next = tail;
        } else {
            node->next = tail->next;
            tail->next = node;
            tail = node;
        }
    }
    void deleteAtBeginning() {
        if (!tail) return;
        CircularNode* head = tail->next;
        if (tail == head) {
            delete head;
            tail = NULL;
        } else {
            tail->next = head->next;
            delete head;
        }
    }
    void deleteAtEnd() {
        if (!tail) return;
        CircularNode* curr = tail->next;
        if (curr == tail) {
            delete tail;
            tail = NULL;
        } else {
            while (curr->next != tail) {
                curr = curr->next;
```



```cpp
            }
            curr->next = tail->next;
            delete tail;
            tail = curr;
        }
    }
    void display() {
        if (!tail) {
            cout << "List empty\n";
            return;
        }
        CircularNode* head = tail->next;
        CircularNode* curr = head;
        do {
            cout << curr->data << " ";
            curr = curr->next;
        } while (curr != head);
        cout << endl;
    }
};
int main() {
    cout << "Doubly Linked List:\n";
    DoublyLinkedList dll;
    dll.insertAtBeginning(10);
    dll.insertAtEnd(20);
    dll.insertAtBeginning(5);
    dll.display();
```

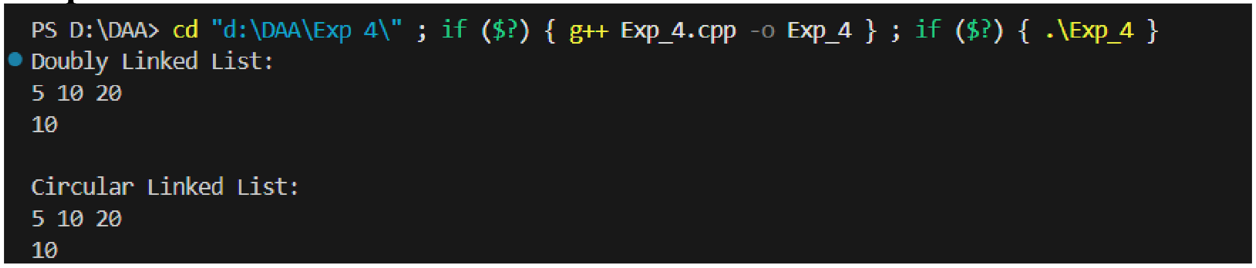


```cpp
    dll.deleteAtBeginning();
    dll.deleteAtEnd();
    dll.display();

    cout << "\nCircular Linked List:\n";
    CircularLinkedList cll;
    cll.insertAtBeginning(10);
    cll.insertAtEnd(20);
    cll.insertAtBeginning(5);
    cll.display();
    cll.deleteAtBeginning();
    cll.deleteAtEnd();
    cll.display();

    return 0;
}
```

6. Output:



```
PS D:\DAA> cd "d:\DAA\Exp 4\" ; if ($?) { g++ Exp_4.cpp -o Exp_4 } ; if ($?) { .\Exp_4 }
Doubly Linked List:
5 10 20
10

Circular Linked List:
5 10 20
10
```