



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment - 4

Student Name: Ayush Ranjan
Branch: BE-CSE
Semester: 5th
Subject Name: DAA

UID: 23BCS10187
Section/Group: KRG-2-B
Date of Performance: 30/7/25
Subject Code: 23CSH-301

1. Aim: To implement insertion and deletion operations at the beginning and end of a Doubly Linked List and a Circular Linked List in C++

2. Procedure:

- Define a node structure with `data` and appropriate pointer fields (`next, prev`).
- For **Doubly Linked List**:
 - Insert at beginning: create node, adjust head and links.
 - Insert at end: traverse to last, link new node.
 - Delete at beginning: update head and free first node.
 - Delete at end: traverse to last, adjust links, free node.
- For **Circular Linked List**:
 - Insert at beginning: create node, link to head and tail.
 - Insert at end: adjust tail to point to new node, link back to head.
 - Delete at beginning: shift head pointer, adjust tail link.
 - Delete at end: traverse to second last, adjust to head, free last node.
 - Display list after each operation to verify correctness.

3. Code:

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

// ----- Doubly Linked List -----
struct DNode {
    int data;
    DNode* prev;
    DNode* next;
};

class DoublyLinkedList {
    DNode* head;
public:
    DoublyLinkedList() : head(NULL) {}

    void insertAtBegin(int val) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
DNode* n = new DNode{val, NULL, head};  
if (head) head->prev = n;  
head = n;  
}  
  
void insertAtEnd(int val) {  
DNode* n = new DNode{val, NULL, NULL};  
if (!head) {  
head = n;  
return;  
}  
DNode* temp = head;  
while (temp->next) temp = temp->next;  
temp->next = n;  
n->prev = temp;  
}  
  
void deleteAtBegin() {  
if (!head) return;  
DNode* temp = head;  
head = head->next;  
if (head) head->prev = NULL;  
delete temp;  
}  
  
void deleteAtEnd() {  
if (!head) return;  
if (!head->next) {  
delete head;  
head = NULL;  
return;  
}  
DNode* temp = head;  
while (temp->next) temp = temp->next;  
temp->prev->next = NULL;  
delete temp;  
}  
  
void display() {  
DNode* temp = head;  
while (temp) {  
cout << temp->data << " ";
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        temp = temp->next;
    }
    cout << endl;
}

// ----- Circular Linked List -----
struct CNode {
    int data;
    CNode* next;
};

class CircularLinkedList {
    CNode* tail;
public:
    CircularLinkedList() : tail(NULL) {}

    void insertAtBegin(int val) {
        CNode* n = new CNode{val, NULL};
        if (!tail) {
            tail = n;
            tail->next = tail;
        } else {
            n->next = tail->next;
            tail->next = n;
        }
    }

    void insertAtEnd(int val) {
        CNode* n = new CNode{val, NULL};
        if (!tail) {
            tail = n;
            tail->next = tail;
        } else {
            n->next = tail->next;
            tail->next = n;
            tail = n;
        }
    }

    void deleteAtBegin() {
        if (!tail) return;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
CNode* head = tail->next;
if (head == tail) {
    delete head;
    tail = NULL;
} else {
    tail->next = head->next;
    delete head;
}
}

void deleteAtEnd() {
if (!tail) return;
CNode* head = tail->next;
if (head == tail) {
    delete tail;
    tail = NULL;
} else {
    CNode* temp = head;
    while (temp->next != tail) temp = temp->next;
    temp->next = head;
    delete tail;
    tail = temp;
}
}

void display() {
if (!tail) {
    cout << endl;
    return;
}
CNode* temp = tail->next;
do {
    cout << temp->data << " ";
    temp = temp->next;
} while (temp != tail->next);
cout << endl;
}
};

// ----- Main Function -----
int main() {
DoublyLinkedList dll;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
dll.insertAtBegin(10);
dll.insertAtEnd(20);
dll.insertAtBegin(5);
dll.display();
dll.deleteAtBegin();
dll.display();
dll.deleteAtEnd();
dll.display();
```

```
CircularLinkedList cll;
cll.insertAtBegin(10);
cll.insertAtEnd(20);
cll.insertAtBegin(5);
cll.display();
cll.deleteAtBegin();
cll.display();
cll.deleteAtEnd();
cll.display();
```

```
return 0;
}
```

4. Output:

```
Doubly Linked List:
5 10 20
10 20
10

Circular Linked List:
5 10 20
10 20
10
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

5. Learning Outcomes:

- Learnt how to create and manage **doubly and circular linked lists**.
- Gained understanding of insertion and deletion at both ends of linked lists.
- Practised pointer manipulation for prev and next links.
- Understood differences between doubly linked and circular linked list structures.
- Developed confidence in handling dynamic data structures in C++.