Experiment - 4

Student Name: Mandeep Kaur **UID:** 23BCS10854
Branch: BE-CSE **Section/Group:** KRG-2B
Semester: 5th **Date of Performance:** 11/8/25
Subject Name: Design and Analysis of Algorithms
Subject Code: 23CSH-301

1. **Aim:** To apply the concept of Linked List and write code to insert and delete an element at the **beginning** and **end** in **Doubly** and **Circular Linked List**.

2. **Objective:** The main objective is to :
   1. Understand the working of Doubly Linked List(DLL) and Circular Linked List(CLL).
   2. Perform Insertion and Deletion operations at both beginning and end of the lists.
   3. Analyze advantages of circular and doubly linked lists over singly linked lists.

3. **Input/ Apparatus Used:**
   Concepts of pointers and dynamic memory allocation.

4. **Algorithm:**

A. Doubly Linked List:

1. Insertion at Beginning:

```
I) Create a new node.
II) If list is empty->head=new node.
III) Else set newNode->next=head and head->prev=newNode.
IV) Update head=NewNode.
```

2. Insertion at end:

```
I) Create a new node.
II) If list is empty → head = new node.
III) Else traverse to last node.
```

IV) Set last->next = newNode and newNode->prev = last.

3. Deletion at Beginning:

I) If list empty → return.
II) Move head to next node, free old head.
III) Update new head's prev = NULL.

4. Deletion at end:

I) If list empty->return.
II) Traverse to last node.
III) Set(last->prev)->next = NULL.
IV) Free last node.

B. Circular Linked List -

1. Insertion at Beginning:

I) Create a new node.
II) If list empty → point new node to itself.
III) Else → newNode->next = head, and last->next = newNode.
IV) Update head = newNode.

2. Insertion at end:

I) Create a new node.
II) If list empty → newNode->next = newNode (self loop).
III) Else → traverse to last node.
IV) Set last->next = newNode, newNode->next = head.

3. Deletion at Beginning:

I) If list empty->return.
II) If only 1 node ->delete it, head=NULL.
III) Else-> find last node.
IV) Set last->next=head->next.
V) Free old head, update head=head->next.

4. Deletion at End:

I) If list empty->return.
II) If only 1 node->delete it,head=NULL.
III) Else->traverse to second last node.

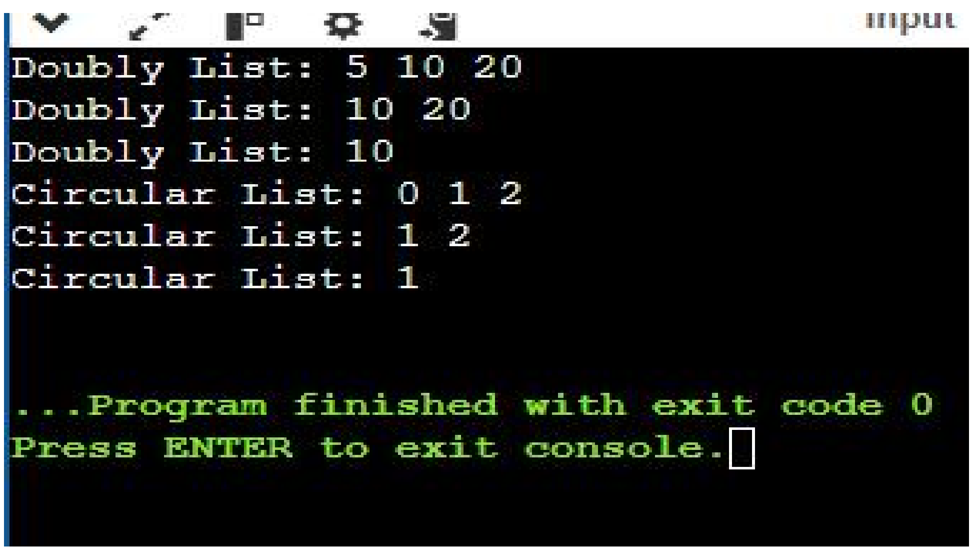
IV) Set secondLast->next=head.
V) Free last node.

5. Code and output:

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

// ---------------- DOUBLY LINKED LIST ----------------
struct DNode {
    int data;
    DNode* prev;
    DNode* next;
};

DNode* dHead = NULL;

void insertAtBeginDLL(int val) {
    DNode* newNode = new DNode{val, NULL, dHead};
    if (dHead != NULL) dHead->prev = newNode;
    dHead = newNode;
}

void insertAtEndDLL(int val) {
    DNode* newNode = new DNode{val, NULL, NULL};
    if (dHead == NULL) { dHead = newNode; return; }
    DNode* temp = dHead;
    while (temp->next) temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}

void deleteAtBeginDLL() {
    if (dHead == NULL) return;
    DNode* temp = dHead;
    dHead = dHead->next;
    if (dHead) dHead->prev = NULL;
    delete temp;
}

void deleteAtEndDLL() {
    if (dHead == NULL) return;
    if (dHead->next == NULL) { delete dHead; dHead = NULL; return; }
    DNode* temp = dHead;
    while (temp->next) temp = temp->next;
    temp->prev->next = NULL;
    delete temp;
}
```

```
void displayDLL() {
    DNode* temp = dHead;
    cout << "Doubly List: ";
    while (temp) { cout << temp->data << " "; temp = temp->next; }
    cout << endl;
}

// ---------------- CIRCULAR LINKED LIST ----------------
struct CNode {
    int data;
    CNode* next;
};

CNode* cHead = NULL;

void insertAtBeginCLL(int val) {
    CNode* newNode = new CNode{val, NULL};
    if (cHead == NULL) { newNode->next = newNode; cHead = newNode; return; }
    CNode* temp = cHead;
    while (temp->next != cHead) temp = temp->next;
    newNode->next = cHead;
    temp->next = newNode;
    cHead = newNode;
}
```

```
void insertAtEndCLL(int val) {
    CNode* newNode = new CNode{val, NULL};
    if (cHead == NULL) { newNode->next = newNode; cHead = newNode; return; }
    CNode* temp = cHead;
    while (temp->next != cHead) temp = temp->next;
    temp->next = newNode;
    newNode->next = cHead;
}

void deleteAtBeginCLL() {
    if (cHead == NULL) return;
    if (cHead->next == cHead) { delete cHead; cHead = NULL; return; }
    CNode* temp = cHead;
    while (temp->next != cHead) temp = temp->next;
    CNode* delNode = cHead;
    temp->next = cHead->next;
    cHead = cHead->next;
    delete delNode;
}

void deleteAtEndCLL() {
    if (cHead == NULL) return;
    if (cHead->next == cHead) { delete cHead; cHead = NULL; return; }
    CNode* temp = cHead;
    while (temp->next->next != cHead) temp = temp->next;
    delete temp->next;
    temp->next = cHead;
}

void displayCLL() {
    if (cHead == NULL) { cout << "Circular List: Empty\n"; return; }
    CNode* temp = cHead;
    cout << "Circular List: ";
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != cHead);
    cout << endl;
}

// ---------------- MAIN ----------------
int main() {
    // Doubly linked list demo
    insertAtBeginDLL(10);
    insertAtEndDLL(20);
    insertAtBeginDLL(5);
    displayDLL();
    deleteAtBeginDLL();
    displayDLL();
    deleteAtEndDLL();
    displayDLL();

    // Circular linked list demo
    insertAtEndCLL(1);
    insertAtEndCLL(2);
    insertAtBeginCLL(0);
    displayCLL();
    deleteAtBeginCLL();
    displayCLL();
    deleteAtEndCLL();
    displayCLL();
    return 0;
}
```



```
Doubly List: 5 10 20
Doubly List: 10 20
Doubly List: 10
Circular List: 0 1 2
Circular List: 1 2
Circular List: 1


...Program finished with exit code 0
Press ENTER to exit console.
```