LAB SESSION 09

Circular Linked List – Insertion and Traversal

THEORY

A Circular Linked List is a variation of a linked list in which the last node points back to the first node, forming a continuous loop. This structure can be implemented using either singly linked or doubly linked nodes.
The primary difference from a regular linked list is that there is no NULL pointer in a CLL — traversal will continue indefinitely unless a stopping condition is specified.
Types of Circular Linked Lists:

1. **Singly Circular Linked List (SCLL)** – Each node has a single pointer to the next node, and the last node links to the first node.
2. **Doubly Circular Linked List (DCLL)** – Each node has two pointers, next and prev, and the last node's next points to the first, while the first node's prev points to the last.

Advantages:

- Can traverse from any node to any other node without going back to the head.
- Useful for applications that require continuous looping (e.g., multiplayer turn-based games).

Disadvantages:

- Slightly more complex pointer management.
- Requires careful loop termination during traversal to avoid infinite loops.

Applications:

- CPU process scheduling (Round Robin algorithm).
- Multiplayer games where turns cycle repeatedly.
- Playlist looping in music applications.

PROCEDURE

1. Define a Node structure with data and next pointer.
2. Implement:
3. insertAtBeginning()
4. insertAtEnd()
5. Update links to maintain circular connection after each insertion.
6. Implement display() with a termination condition to avoid infinite loops.
7. Test both insertion methods and display function.

CODE

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

struct Node {
   int data;
   Node* next;
};
```

```
Node* head = NULL;

void insertAtBeginning(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    if (head == NULL) {
        head = newNode;
        head->next = head;
        return;
    }
    Node* temp = head;
    while (temp->next != head) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->next = head;
    head = newNode;
}

void insertAtEnd(int value) {
    Node* newNode = new Node();
    newNode->data = value;
    if (head == NULL) {
        head = newNode;
        head->next = head;
        return;
    }
    Node* temp = head;
    while (temp->next != head) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->next = head;
}

void display() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }
    Node* temp = head;
    do {
        cout << temp->data << " ";
        temp = temp->next;
    } while (temp != head);
    cout << endl;
}

int main() {
    insertAtBeginning(10);
    insertAtEnd(20);
    insertAtEnd(30);
    insertAtBeginning(5);
    display();
    return 0;
}
```

EXPECTED OUTPUT

```
5 10 20 30
```

EXERCISE

1. Implement an insertAtPosition() function for a CLL.
2. Modify the code to work with a **doubly circular linked list**.
3. Write a function to search for a value in the CLL.
4. Implement a delete function for the first node in the CLL.