## **LEC-5 Circular Linked List**

using namespace std;

#include <iostream>

class circularlinklist; // Forward declaration

class Node {

    friend class circularlinklist;

    int data;

    Node\* next;

};

class circularlinklist {

private:

    Node\* head;

public:

    circularlinklist() {

        head = nullptr;

    }

    ~circularlinklist() {

        // TODO: Implement destructor to free memory used by the list nodes

    }

    bool insertatstart(int val) {

        Node\* newNode = new Node();

        newNode->data = val;

        newNode->next = newNode; // Circular connection

        if (head == nullptr) {

            head = newNode;

        } else {

            Node\* current = head;

            while (current->next != head) {

                current = current->next;

            }

            current->next = newNode;

            newNode->next = head;

            head = newNode;

        }

        return true;

    }

    bool insertatend(int val) {

        Node\* newNode = new Node();

        newNode->data = val;

        newNode->next = newNode; // Circular connection

        if (head == nullptr) {

            head = newNode;

        } else {

            Node\* current = head;

            while (current->next != head) {

                current = current->next;

            }

            current->next = newNode;

            newNode->next = head;

        }

        return true;

    }

    bool deletefromstart() {

        if (head == nullptr) {

            return false;

        }

        Node\* temp = head;

        if (head->next == head) { // Only one node

            head = nullptr;

        } else {

            Node\* current = head;

            while (current->next != head) {

                current = current->next;

            }

            current->next = head->next;

            head = head->next;

        }

        delete temp;

        return true;

    }

    bool deletefromend() {

        if (head == nullptr) {

            return false;

        }

        if (head->next == head) { // Only one node

            delete head;

            head = nullptr;

            return true;

        }

        Node\* prev = nullptr;

        Node\* current = head;

        while (current->next != head) {

            prev = current;

            current = current->next;

        }

        prev->next = head;

        delete current;

        return true;

    }

    void display() {

        if (head == nullptr) {

            cout << "List is empty." << endl;

            return;

        }

        Node\* current = head;

        do {

            cout << current->data << " ";

            current = current->next;

        } while (current != head);

        cout << endl;

    }

};

int main() {

    circularlinklist list;

    list.insertatstart(3);

    list.insertatstart(2);

    list.insertatstart(1);

    list.insertatend(4);

    list.insertatend(5);

    cout << "Original List: ";

    list.display();

    list.deletefromstart();

    list.deletefromend();

    cout << "Modified List: ";

    list.display();

    return 0;

}