**NAME : MALIHA SHAHID**

**ROLL NO:SU92-BSSEM-S24-071**

**LAB 9**

#include <iostream>

using namespace std;

class Node {

public:

int value;

Node\* next;

Node\* prev;

Node(int v) {

value = v;

next = prev = NULL;

}

};

class CircularLinkedList {

public:

Node\* head;

CircularLinkedList() {

head = NULL;

}

void insert\_AT\_First(int val) {

Node\* newNode = new Node(val);

if (!head) {

head = newNode;

head->next = head;

head->prev = head;

} else {

Node\* tail = head->prev;

newNode->next = head;

newNode->prev = tail;

tail->next = newNode;

head->prev = newNode;

head = newNode;

}

}

void insert\_AT\_Last(int val) {

if (!head) {

insert\_AT\_First(val);

return;

}

Node\* newNode = new Node(val);

Node\* tail = head->prev;

newNode->next = head;

newNode->prev = tail;

tail->next = newNode;

head->prev = newNode;

}

void insert\_At\_Nth(int val, int pos) {

if (pos == 1 || !head) {

insert\_AT\_First(val);

return;

}

Node\* temp = head;

for (int i = 1; i < pos - 1 && temp->next != head; i++) {

temp = temp->next;

}

Node\* newNode = new Node(val);

newNode->next = temp->next;

newNode->prev = temp;

temp->next->prev = newNode;

temp->next = newNode;

}

void insert\_AT\_Center(int val) {

if (!head || head->next == head) {

insert\_AT\_First(val);

return;

}

Node\* slow = head;

Node\* fast = head;

while (fast->next != head && fast->next->next != head) {

slow = slow->next;

fast = fast->next->next;

}

insert\_At\_Nth(val, (fast->next == head) ? 2 : 3);

}

void displayForward() {

if (!head) {

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

return;

}

Node\* temp = head;

do {

cout << temp->value << " -> ";

temp = temp->next;

} while (temp != head);

cout << "(head)" << endl;

}

void displayReverse() {

if (!head) {

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

return;

}

Node\* tail = head->prev;

Node\* temp = tail;

do {

cout << temp->value << " <- ";

temp = temp->prev;

} while (temp != tail);

cout << "(head)" << endl;

}

};

int main() {

CircularLinkedList cll;

cll.insert\_AT\_Last(10);

cll.insert\_AT\_Last(20);

cll.insert\_AT\_First(5);

cll.insert\_At\_Nth(15, 3);

cll.insert\_AT\_Center(12);

cout << "Circular Linked List (Forward): ";

cll.displayForward();

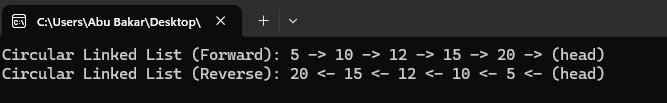
cout << "Circular Linked List (Reverse): ";

cll.displayReverse();

return 0;

}

**OUTPUT**

****

**EXPLANATION**

* **Node Class**: Represents a node with value, next (points to next node), and prev (points to previous node).
* **CircularLinkedList Class**: Manages a circular doubly linked list with head pointing to the first node.
* **insert\_AT\_First(val)**: Adds a node at the beginning and updates links properly.
* **insert\_AT\_Last(val):** Adds a node at the end and maintains circular links.
* **insert\_At\_Nth(val, pos):** Inserts a node at the given position in the list.
* **insert\_AT\_Center(val):** Finds the middle of the list and inserts a node there.
* **displayForward():** Prints the list in forward order.
* **displayReverse():** Prints the list in reverse order.