**Question No:09**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int value) : data(value), next(NULL) {}

};

class CircularLinkedList {

private:

Node\* head;

public:

CircularLinkedList() : head(NULL) {}

void insertAtFirst(int value) {

Node\* newNode = new Node(value);

if (!head) {

head = newNode;

newNode->next = head;

} else {

Node\* temp = head;

while (temp->next != head) {

temp = temp->next;

}

temp->next = newNode;

newNode->next = head;

head = newNode;

}

}

void insertAtLast(int value) {

Node\* newNode = new Node(value);

if (!head) {

head = newNode;

newNode->next = head;

} else {

Node\* temp = head;

while (temp->next != head) {

temp = temp->next;

}

temp->next = newNode;

newNode->next = head;

}

}

void insertAtNth(int value, int position) {

if (position == 0) {

insertAtFirst(value);

return;

}

Node\* newNode = new Node(value);

Node\* temp = head;

for (int i = 0; i < position - 1; i++) {

if (temp->next == head) {

insertAtLast(value);

return;

}

temp = temp->next;

}

newNode->next = temp->next;

temp->next = newNode;

}

void insertAtCenter(int value) {

if (!head) {

insertAtFirst(value);

return;

}

Node\* slow = head;

Node\* fast = head;

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

slow = slow->next;

fast = fast->next->next;

}

Node\* newNode = new Node(value);

newNode->next = slow->next;

slow->next = newNode;

}

void display() {

if (!head) return;

Node\* temp = head;

do {

std::cout << temp->data << " ";

temp = temp->next;

} while (temp != head);

std::cout << std::endl;

}

void displayReverse() {

if (!head) return;

Node\* temp = head;

Node\* prev = NULL;

Node\* nextNode = NULL;

do {

nextNode = temp->next;

temp->next = prev;

prev = temp;

temp = nextNode;

} while (temp != head);

head->next = prev;

head = prev;

Node\* current = head;

do {

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

current = current->next;

} while (current != head);

std::cout << std::endl;

temp = head;

prev = NULL;

nextNode = NULL;

do {

nextNode = temp->next;

temp->next = prev;

prev = temp;

temp = nextNode;

} while (temp != head);

head->next = prev;

head = prev;

}

};

int main() {

CircularLinkedList cll;

cll.insertAtFirst(10);

cll.insertAtLast(20);

cll.insertAtLast(30);

cll.insertAtNth(25, 2);

cll.insertAtCenter(15);

std::cout << "Circular Linked List in order: ";

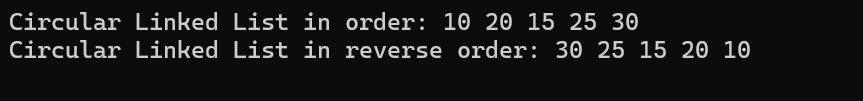
cll.display();

cout << "Circular Linked List in reverse order: ";

cll.displayReverse();

}

**OUTPUT**

****

**Explanation**

In this code the circular Linked List connects elements in a circle. Key functions include **insertAtFirst()**, **insertAtLast()**, **insertAtNth()**, **display()**, and **displayReverse()** for inserting nodes at specific positions and displaying the list in both forward and reverse order. Example usage: Create a list, insert nodes, and print the list in order and reverse.