

|  |
| --- |
| **LAB 9** of DSA LAB |

**Submitted by:**

**Name:** Muhammad shuraim

**Roll no:** SU92-S24-BSSEM-103

**Section:** 3A

**Submitted to**

Sir Rasikh Ali

**Question 1**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int d) {

data = d;

next = nullptr;

}

};

class CircularSinglyLinkedList {

public:

Node\* head;

CircularSinglyLinkedList() {

head = nullptr;

}

void insertAtStart(int d) {

Node\* newnode = new Node(d);

if (head == nullptr) {

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 d) {

if (head == nullptr) {

insertAtStart(d);

return;

}

Node\* newnode = new Node(d);

Node\* temp = head;

while (temp->next != head) {

temp = temp->next;

}

temp->next = newnode;

newnode->next = head;

}

void insertAtSpecLoc(int pos, int d) {

if (pos < 1) {

cout << "INVALID POSITION!" << endl;

return;

}

if (pos == 1) {

insertAtStart(d);

return;

}

Node\* newnode = new Node(d);

Node\* temp = head;

int count = 1;

while (count < pos - 1 && temp->next != head) {

temp = temp->next;

count++;

}

if (count < pos - 1) {

cout << "INVALID POSITION!" << endl;

return;

}

newnode->next = temp->next;

temp->next = newnode;

}

void insertAtCenter(int d) {

if (head == nullptr) {

insertAtStart(d);

return;

}

Node\* slow = head;

Node\* fast = head;

do {

if (fast->next == head || fast->next->next == head) break;

slow = slow->next;

fast = fast->next->next;

} while (fast != head && fast->next != head);

Node\* newnode = new Node(d);

newnode->next = slow->next;

slow->next = newnode;

}

void display() {

if (head == nullptr) return;

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << "(HEAD)" << endl;

}

void displayReverse(Node\* temp) {

if (temp == nullptr || temp->next == head) {

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

return;

}

displayReverse(temp->next);

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

}

void displayReverse() {

if (head == nullptr) return;

displayReverse(head);

cout << "(HEAD)" << endl;

}

};

int main() {

CircularSinglyLinkedList obj;

obj.insertAtStart(10);

obj.insertAtStart(20);

obj.insertAtEnd(30);

obj.insertAtEnd(40);

obj.insertAtSpecLoc(3, 25);

obj.insertAtCenter(15);

cout << "Circular Singly Linked List in Order: " << endl;

obj.display();

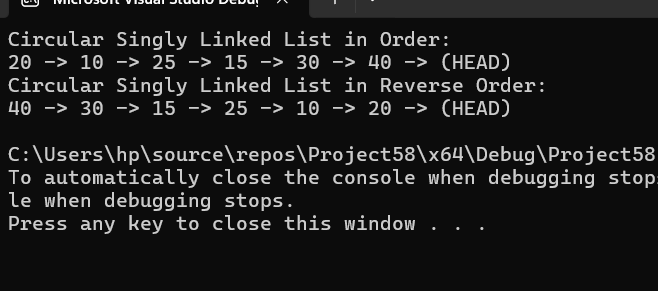
cout << "Circular Singly Linked List in Reverse Order: " << endl;

obj.displayReverse();

return 0;

}

**OUTPUT**

****

Explanation

* The Node class represents an individual node in the linked list.
* Each node has:
* An integer data to store the value.
* A pointer next to point to the next node in the list.
* The constructor initializes a new node with given data (d) and sets next to nullptr.
* The CircularSinglyLinkedList class manages the linked list.
* It has a pointer head, which always points to the first node.
* The constructor initializes head to nullptr (empty list).
* A new node is created with the given data.
* If the list is empty, the node links to itself.
* Otherwise, the function finds the last node (temp) and updates its next pointer.
* The new node is set as head.
* If the list is empty, it calls insertAtStart().
* Otherwise, it finds the last node and inserts the new node after it, maintaining the circular structure.
* If pos == 1, it calls insertAtStart().
* Otherwise, it traverses the list to the (pos-1)th node.
* If pos is invalid, it prints an error message.
* Otherwise, it inserts the new node at the desired position.
* If the list is empty, it inserts at the start.
* Uses slow and fast pointers to find the middle of the list:
* slow moves one step at a time.
* fast moves two steps at a time.
* Inserts the new node after slow, which is at the center.