Emam saimon

Here's a clear explanation and implementation of **insertion in a circular singly linked list** in C++. A circular singly linked list is like a singly linked list, except the last node points back to the head.

We will cover the following insertion cases:

1. Insert at the beginning.
2. Insert at the end.
3. Insert after a given node.

☑ Circular Singly Linked List Node Structure

#include <iostream>

using namespace std;

struct Node {

  int data;

  Node* next;

};

🔁 Insert at the End

void insertEnd(Node** head, int value) { Node* newNode = new Node(); newNode->data = value;

```
if (*head == nullptr) {
    newNode->next = newNode; // Point to itself
    *head = newNode;
    return;
}

Node* temp = *head;
```

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

```
while (temp->next != *head) {
    temp = temp->next;
}

temp->next = newNode;
newNode->next = *head;


}
```

🔁 Insert at the Beginning

void insertBeginning(Node** head, int value) { Node* newNode = new Node(); newNode->data = value;

```
if (*head == nullptr) {
    newNode->next = newNode;
    *head = newNode;
    return;
}

Node* temp = *head;
while (temp->next != *head) {
    temp = temp->next;
}

temp->next = newNode;
newNode->next = *head;
*head = newNode;


}
```

🔁 Insert After a Given Node

void insertAfter(Node* prevNode, int value) {

if (prevNode == nullptr) {

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

```
        cout << "Previous node cannot be NULL\n";
        return;
    }

    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = prevNode->next;
    prevNode->next = newNode;
}
```

🔁 Display the List

```
void display(Node* head) {
    if (head == nullptr) return;

    Node* temp = head;
    do {
        cout << temp->data << " -> ";
        temp = temp->next;
    } while (temp != head);
    cout << "(head)\n";
}
```

🧪 Example Usage

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

```
int main() {
   Node* head = nullptr;


   insertEnd(&head, 10);
   insertEnd(&head, 20);
   insertBeginning(&head, 5);
   insertAfter(head->next, 15); // Insert 15 after 10


   display(head);
   return 0;
}
```

✅ **Insert at the End – Circular Singly Linked List (C++)**

#include using namespace std;

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

// Insert at the end void insertEnd(Node** head, int value) { Node* newNode = new Node(); newNode->data = value;

```
if (*head == nullptr) {
    newNode->next = newNode; // First node points to itself
    *head = newNode;
    return;
}

Node* temp = *head;
```

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

```
while (temp->next != *head) {
    temp = temp->next;
}

temp->next = newNode;
newNode->next = *head;


}
```

// Display the circular list void display(Node* head) { if (head == nullptr) { cout << "List is empty\n"; return; }

```
Node* temp = head;
do {
    cout << temp->data << " -> ";
    temp = temp->next;
} while (temp != head);
cout << "(head)\n";


}
```

int main() { Node* head = nullptr;

```
insertEnd(&head, 10);
insertEnd(&head, 20);
insertEnd(&head, 30);
insertEnd(&head, 40);

cout << "Circular Singly Linked List:\n";
display(head);

return 0;


}
```

✅ **Function to Delete the First Node**

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

void deleteFirst(Node** head) { if (*head == nullptr) { cout << "List is empty.\n"; return; }

```
Node* temp = *head;

// If there's only one node
if (temp->next == *head) {
    delete temp;
    *head = nullptr;
    return;
}

// Find the last node
Node* last = *head;
while (last->next != *head) {
    last = last->next;
}

// Update last node's next pointer and head
last->next = (*head)->next;
*head = (*head)->next;

delete temp;


}
```

✅ **Function to Delete the Last Node**

void deleteLast(Node** head) { if (*head == nullptr) { cout << "List is empty.\n"; return; }

```
Node* temp = *head;

// If only one node
if (temp->next == *head) {
    delete temp;
    *head = nullptr;
```

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-

```
    return;
}

Node* prev = nullptr;
while (temp->next != *head) {
    prev = temp;
    temp = temp->next;
}

prev->next = *head;
delete temp;


}
```

https://github.com/EmamSaimon592/Circular-Singly-Linked-List-