# EXPERIMENT-14

# Aim: Write a menu driven program to show Circular linked list implementation. It should consist of insert at head , insert at tail & insert at middle.

## Pseudo code

// Create a struct for a linked list node with 'data' and 'next' fields

// Function to insert a node at the tail of the circular linked list

insertattail(head, val):

Create a new node 'n' with 'val'

if head is NULL:

Set head to 'n'

Set n's next pointer to 'n'

return

Initialize a temporary node 'temp' to head

while temp's next is not head:

Move temp to temp's next

Set temp's next to 'n'

Set n's next to head

// Function to insert a node at the head of the circular linked list

insertathead(head, val):

Create a new node 'n' with 'val'

if head is NULL:

Set head to 'n'

Set n's next pointer to 'n'

return

Initialize a temporary node 'temp' to head

while temp's next is not head:

Move temp to temp's next

Set temp's next to 'n'

Set n's next to head

Set head to 'n'

// Function to insert a node at the middle of the circular linked list

insertatmiddle(head, val, position):

Create a new node 'n' with 'val'

if head is NULL or position is 1:

Call insertathead(head, val)

return

Initialize count to 1

Initialize a temporary node 'temp' to head

while temp's data is not val and count is less than position - 1:

Move temp to temp's next

Increment count

Set n's next to temp's next

Set temp's next to 'n'

// Function to display the circular linked list

display(head):

Initialize a temporary node 'temp' to head

Loop:

Print temp's data

Move temp to temp's next

if temp is equal to head, break the loop

Print a new line

## Source code:

//CIRCULAR LINKED LIST

#include<iostream>

using namespace std;

struct node{

    int data;

    node\* next;

    node(int val){

        data = val;

        next = NULL;

    }

};

void insertattail(node\* &head, int val){

    node\* n = new node(val);

    if (head==NULL){                     // FUNCTION

        head=n;                          // TO INSERT

        n->next=n;                       // AT TAIL

        return;

    }

    node\* temp = head;

    while(temp->next!=head){

        temp=temp->next;

    }

    temp->next=n;

    n->next=head;

}

void insertathead(node\* &head,int val){

    node\* n = new node(val);

    if (head==NULL){                           // FUNCTION

        head=n;                                // TO INSERT

        n->next=n;                             //   HEAD

        return;

    }

    node\* temp = head;

    while(temp->next!=head){

        temp=temp->next;

    }

    temp->next=n;

    n->next=head;

    head=n;

}

void insertatmiddle(node\* &head,int val,int position){

    int count=1;

    node\* n = new node(val);

    if (head==NULL ){                          // FUNCTION

        head=n;                                // TO INSERT

        n->next=n;                             // AT MIDDLE

        return;

    }

    if (position == 1){

        insertathead(head,val);

        return;

    }

    node\* temp =head;

    while(temp->data!=val && count < position-1){

        temp=temp->next;

        count++;

    }

    n->next=temp->next;

    temp->next=n;

}

void display(node\* head){

    node\* temp =head;

    do {

        cout<<temp->data<<" ";

        temp=temp->next;

    }

    while (temp!=head);

    cout<<endl;

}

int main(){

    node\* head = NULL;

    insertattail(head,2);

    insertattail(head,3);

    insertattail(head,4);

    insertattail(head,5);

    display(head);

    insertathead(head,1);

    display(head);

    insertatmiddle(head,9,3);

    display(head);

}

## Output:

**2 3 4 5**

**1 2 3 4 5**

**1 2 9 3 4 5**

## Learning from experiment

* Circular linked list insertion methods.
* Efficient middle insertion in circular linked list.