# Experiment - 16

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

## Pseudo code

Define a structure for a doubly linked list node:

- data: integer value

- next: pointer to the next node

- prev: pointer to the previous node

Function to insert a node at the head of the list:

Input: head (pointer to the list head), val (value to insert)

Create a new node 'n' with the given value

Make 'n' point to the current head node as its next

If the head is not null, make the previous of the current head point to 'n'

Update the head to point to 'n'

Function to insert a node at the tail of the list:

Input: head (pointer to the list head), val (value to insert)

Create a new node 'n' with the given value

Traverse the list to find the current tail node

Make the next of the current tail point to 'n'

Make the previous of 'n' point to the current tail

Update the tail to point to 'n'

Function to insert a node at a specified position in the list:

Input: head (pointer to the list head), val (value to insert), position

Create a new node 'n' with the given value

Traverse the list to find the node at the specified position

Update pointers to insert 'n' in the middle

- Set the next of 'n' to the next of the current node

- Set the previous of 'n' to the current node

- Set the next of the current node to 'n'

- Set the previous of the next node to 'n'

Function to display the elements of the doubly linked list:

Input: head (pointer to the list head)

Initialize a temporary pointer 'temp' to the head

Loop until 'temp' is not null:

Print the data value at 'temp'

Move 'temp' to the next node

End loop

Main program:

Initialize a pointer 'head' to null

Insert elements at the tail, head, and middle of the list

Display the elements of the list after each insertion

## Source code:

#include<iostream>

using namespace std;

struct node{

    int data;

    node\* next;

    node\* prev;

    node(int val){

        data = val;

        next = NULL;

    }

};

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

   node\* n= new node(val);                   // TO INSERT

    n->next=head;                            //  AT HEAD

    if(head!=NULL){

    head->prev=n;

    }

    head=n;

}

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

    node\* n=new node(val);

    if (head==NULL){                         //  FUNCTION

       insertathead(head,val);               //  TO INSERT

       return;                               //   AT TAIL

    }

    node\* temp =head;

    while (temp->next!=NULL){

        temp=temp->next;

    }

    temp->next=n;

    n->prev=temp;

}

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

    int count=1;

    node\* n = new node(val);

    if (head==NULL || position == 1){            //  FUNCTION

        insertathead(head,val);                  //  TO INSERT

        return;                                  //  AT MIDDLE

    }

    node\* temp = head;

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

        temp=temp->next;

        count++;

    }

    n->next=temp->next;

    temp->next=n;

    n->next->prev=temp->next;

    n->prev =temp;

}

void display(node\* head){

    node\* temp=head;

    while (temp!=NULL){

        cout<<temp->data<<" ";

        temp=temp->next;

    }

    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

* In this code, a doubly linked list is implemented. Each node in the list contains a "prev" pointer in addition to the "next" pointer, allowing for easier traversal in both directions.
* The code demonstrates the insertion of nodes at the head, tail, and middle of the doubly linked list, and it correctly maintains the "prev" and "next" pointers to ensure proper linkage between nodes during insertion.