# EXPERIMENT-17

# Aim: Write a menu driven program to show Doubly linked list implementation. It should consist of delete from head , delete from end & delete from middle.

## Pseudo code

## Initialize an empty doubly linked list with a "head" pointer set to NULL.

## Insert a node with value 2 at the tail:

## Create a new node with value 2 and a "next" pointer set to NULL.

## If the list is empty (head is NULL):

## Set the head to the new node.

## Otherwise:

## Traverse the list to the last node.

## Update the "next" pointer of the last node to point to the new node.

## Set the "prev" pointer of the new node to point to the last node.

## Insert a node with value 1 at the head:

## Create a new node with value 1, "next" pointer pointing to the current head, and "prev" set to NULL.

## Update the "prev" pointer of the old head to point to the new node.

## Update the head to point to the new node.

## Delete the head node:

## Update the head to point to the next node.

## Set the "prev" pointer of the new head to NULL.

## Delete the tail node:

## Traverse the list to the last node.

## Update the "next" pointer of the second-to-last node to NULL.

## Delete the last node.

## Delete a node at a specific position:

## Traverse the list to the node at the given position.

## Update the "next" pointer of the previous node to point to the next node.

## Update the "prev" pointer of the next node to point to the previous node.

## Delete the node to be removed.

## 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;

}

// DELETION

void deleteathead(node\* &head){       // FUNCTION

    head=head->next;                  // TO DELETE

    head->prev=NULL;                  //  AT HEAD

    return;

}

void deleteatmiddle(node\* &head , int pos){     //  FUNCTION

    int count=1;                                //  TO DELETE

    if (head==NULL || pos == 1){                // FROM MIDDLE

        deleteathead(head);

        return;

    }

    node\* temp=head;

    while (temp!=NULL && count!=pos){

        temp=temp->next;

        count++;

    }

    temp->prev->next=temp->next;

    if (temp->next=NULL){

    temp->next->prev=temp->prev;

}

}

void deleteatend(node\* &head){

    if (head==NULL){               // FUNCTION

        return;                    // TO DELETE

    }                              //  AT END

    node\* temp = head ;

    while(temp->next!=NULL){

        temp=temp->next;

    }

   if (temp->prev->next!=NULL){

    temp->prev->next=NULL;

   }

   delete 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);

    deleteathead(head);

    display(head);

    deleteatend(head);

    display(head);

    deleteatmiddle(head,2);

    display(head);

}

## Output:

## 1 2 3 4 5

## 2 3 4 5

## 2 3 4

## 2 4

## Learning from experiment

* **Doubly Linked List**: This code demonstrates the implementation of a doubly linked list, where each node has both next and previous pointers, allowing for easier traversal in both directions.
* **Insertion and Deletion**: The code shows how to insert nodes at the head, tail, and middle of the linked list, as well as how to delete nodes from the head, tail, and middle, illustrating basic operations on a doubly linked list.