

## Source Code

```
#include<iostream>
using namespace std;
class node{
public:
    int data;
    node* prev,*next;
    node(int val){
        data=val;
        prev=NULL;
        next=NULL;
    }
};

void insertAtHead(node* &head,int val){
    node* n=new node(val);
    if(head==NULL){
        head=n;
        return;
    }
    head->prev=n;
    n->next=head;
    head=n;
}

void insertAtTail(node* head,int val){
    if(head==NULL){
        insertAtHead(head,val);
    }
    node* n=new node(val);
    node* temp=head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=n;
    n->prev=temp;
}

void insertAftLoc(node* &head,int key,int val){//INSERT AFTER GIVEN KEY

    if(head==NULL){
        return;
    }
    if(head->data==key){
        insertAtHead(head,val);
        return;
    }
    node* n=new node(val);
    node* temp=head;
    while(temp!=NULL and temp->data!=key){
        temp=temp->next;
    }
}
```

```

        if(temp==NULL){
            insertAtHead(head,val);
            return;
        }
        else if(temp->next==NULL){
            insertAtTail(head,val);
            return;
        }
        node* newnode=temp->next;
        temp->next=n;
        n->prev=temp;
        newnode->prev=n;
        n->next=newnode;
    }
}

void deleteAtstart(node* &head){
    if(head==NULL){
        cout<<"List is empty"<<endl;
        return;
    }
    node* temp=head;
    temp->next->prev=NULL;
    head=temp->next;
    delete temp;
}

void deletAftLoc(node* &head,int loc){ //delete After loc
    if(head==NULL){
        cout<<"List is empty"<<endl;
        return;
    }
    else{
        node* temp=head;
        while(temp!=NULL and temp->data!=loc){
            temp=temp->next;
        }
        if(temp==NULL){
            return;
        }
        node* nextptr=temp->next;
        if(nextptr==NULL){
            deleteAtstart(head);
            return;
        }
        nextptr->prev->next=nextptr->next;
        if(nextptr->next!=NULL){
            nextptr->next->prev=nextptr->prev;
        }
        delete nextptr;
    }
}

```

```

    }
}
void display(node* head){
    cout<<"Display in original form: ";
    if(head==NULL){
        return;
    }
    while(head!=NULL){
        cout<<head->data<<"->";
        head=head->next;
    }
    cout<<"NULL"<<endl;
}
void display2(node* head){
    cout<<"Display in reverse: ";
    if(head==NULL){
        return;
    }
    while(head->next!=NULL){
        head=head->next;
    }
    while(head!=NULL){
        cout<<head->data<<"->";
        head=head->prev;
    }
    cout<<"NULL"<<endl;
}
int main(){

node* head=NULL;

insertAtHead(head,3);
insertAtHead(head,2);
insertAtHead(head,1);
display(head);
display2(head);

```

```

insertAftLoc(head,2,4);
insertAftLoc(head,4,5);
insertAftLoc(head,3,6);
insertAftLoc(head,8,7);
display(head);

```

```

insertAtTail(head,9);
insertAtTail(head,8);
display(head);

```

## Output

```
PS C:\Users\anil kumar\Documents\anil\.vscode\DataStructure_in_nsut> cd "c:\Users\anil kumar\Documents\anil\.vscode\6_Doublylinklist\" ; if ($?) { g++ -std=c++17 6_Doublylinklist.cpp -o 6_Doublylinklist } ; if ($?) { .\6_Doublylinklist }
Display in original form: 1->2->3->NULL
Display in reverse: 3->2->1->NULL
Display in original form: 7->1->2->4->5->3->6->NULL
Display in original form: 7->1->2->4->5->3->6->9->8->NULL
deletion after loc
Display in original form: 1->4->5->3->6->8->NULL
PS C:\Users\anil kumar\Documents\anil\.vscode\DataStructure_in_nsut>
```