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;
    }
    if(val<head->data){
        head->prev=n;
        n->next=head;
        head=n;
    }
    else{
        cout<<"Value:"<<val<<" is greater than head"<<endl;
    }
}
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;
    }
    if(val>temp->data){
        temp->next=n;
        n->prev=temp;
    }
    else{
        cout<<"Value:"<<val<<" is Smaller than Tail"<<endl;
    }

}
void insertEle(node* &head,int val){  //INSERT VALUE
    if(head==NULL){
        insertAtHead(head,val);
```

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

}

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 deletion(node* &head,int val){  //delete At pos
    if(head==NULL){
        cout<<"List is empty"<<endl;
        return;
    }
    if(head->data==val){
        deleteAtstart(head);
    }
    else{
        node* temp=head;
        while(temp!=NULL and temp->data<val){
            temp=temp->next;
```

```
        }
        if(temp==NULL||temp->data>val){
            return;
        }
        temp->prev->next=temp->next;
        if(temp->next!=NULL){
            temp->next->prev=temp->prev;
        }
        delete temp;
    }
}
void display(node* head){
    cout<<"Display in Sorted 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;
insertEle(head,5);
insertEle(head,3);
insertEle(head,2);
insertEle(head,4);
insertEle(head,9);
insertEle(head,8);
insertEle(head,7);
insertEle(head,1);
display(head);
```

```
display2(head);

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

```
cout<<" deletion begin"<<endl;
deletion(head,6);
deletion(head,8);
deletion(head,1);
deletion(head,7);
deletion(head,3);

display(head);
return 0;
```

OUTPUT

```
PS C:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut> cd "c:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut
\" ; if ($?) { g++ -std=c++17 6_DoublySortedList.cpp -o 6_DoublySortedList } ; if ($?) { .\6_DoublySortedList }
Display in Sorted form: 1->2->3->4->5->7->8->9->NULL
Display in reverse: 9->8->7->5->4->3->2->1->NULL
Value:9 is Smaller than Tail
Value:8 is Smaller than Tail
Display in Sorted form: 1->2->3->4->5->7->8->9->NULL
 deletion begin
Display in Sorted form: 2->4->5->9->NULL
PS C:\Users\anil kumar\Documents\anil\.vscode\DataSructure_in_nsut>
```