```
#include <iostream>
using namespace std;
class Node{
private:
   int data;
   Node* next;
public :
   void putData(int value){
      data=value;
   }
   void putNext(Node* node){
      next=node;
   }
   int getData() const {
      return data;
   }

   Node *getNext() const {
        return next;
    }
};
class Sorted{
private:
    Node* start;
public:
    Sorted(){
        start=NULL;
    }
    void putAtBeg(int num){
        Node* n =new Node();
        n->putData(num);
        n->putNext(start);
        start=n;
    }
    void putAtEnd(int num){
        Node* n,*t;
        n=new Node();
        t=start;
        n->putData(num);
        while(t->getNext()!=NULL)
            t=t->getNext();
        t->putNext(n);
    }
    void insert(int put){
        Node* temp=start;
        Node* last=start;
        if(start==NULL){
            Node* n=new Node();
            n->putData(put);
            start=n;}
        else {
            if(!searchItem(put)) {//not to add
same element
                Node *n = new Node();
                n->putData(put);
                while (last->getNext() != NULL)
                    last =
last->getNext();//pointing last to the last
node
                if (start->getData() > put)//if
inserting value is less then the beginning
put it at big
                    putAtBeg(put);
                else if (last->getData() < put)
                    putAtEnd(put);//if inserting
value is large then the last value put at
last
                else {//pointing temp until the
value is between small and large
                    while (temp->getData() < put
&& put > temp->getNext()->getData()) {
                    temp = temp->getNext();
                    }

n->putNext(temp->getNext());;
                    temp->putNext(n);
                }
            }
        }

    }
    void deleteNode(int del){
        if(start!=NULL){
            if(searchItem(del)) {
                Node *current = start;
                if (start->getData() == del) {
                    Node *n = start;
                    start = start->getNext();
                    delete n;
                }else {
```

```
                while
(current->getNext()->getData() != del)
                    current =
current->getNext();//jis node ko delete
karna ha current ko uss se phle node tak
point karna
                Node *deleValue =
current->getNext();//delvalue point to
the node jis hma delete karna ha

current->putNext(deleValue->getNext());
                delete deleValue;
              }
          } else
                  cout<<"NUumber is not
present "<<endl;
      }
      else
```

```
            cout<<"List is empty"<<endl;

    }

    bool searchItem(int q){
        Node* n;
        bool check=false;
        n=start;
        while(n!=NULL){
            if(n->getData()==q){
                check=true;
                return check;
            }
            n=n->getNext();
        }
        return check;
    }
    void display(){
```

```
    if(start!=NULL){
        cout<<"List is : ";
        Node* temp;
        temp=start;
        while(temp!=NULL){
            cout<<temp->getData()<<" ";
            temp=temp->getNext();
        }
    }else
             cout<<"List is Empty";
}
void merge(){
    Sorted l1;
    int size1,size2,num=0;
    cout<<"Enter the size of first list: ";
    cin>>size1;
    cout<<"Enter the first list: "<<endl;
    for(int i=0;i<size1;i++){
            cin>>num;
            l1.insert(num);}
        l1.display();
        cout<<"Enter the size of second list:
";
        cin>>size2;
        cout<<"Enter the second list"<<endl;
        for(int i=0;i<size2;i++){
            cin>>num;
            l1.insert(num);
        }
        cout<<"After Merging: "<<endl;
        l1.display();
    }
};
int main() {
    bool  check=true;
    int choice,num;
```

```
Sorted sb;

cout<<"1. Insert"<<endl;
cout<<"2. Delete"<<endl;
cout<<"3. Merge"<<endl;
cout<<"4. Display"<<endl;
cout<<"5. Exit"<<endl;
while(check){
    cout<<"\nEnter choice: ";
    cin>>choice;
    switch(choice){
        case 1:
            cout<<"Enter the number: ";
            cin>>num;
            sb.insert(num);
            break;
        case 2:
                cout<<"Enter the number: ";
                cin>>num;
                sb.deleteNode(num);
                break;
            case 3:
                sb.merge();
                break;
            case 4:
                sb.display();
                break;
            case 5:
                check= false;
                break;
            default:
                cout<<"Wrong choice ";
        }
    }
    return 0;
```

```
}
```