```java
import java.util.*;
class HeapSortID {
    static void heapifyMax(int[] a,int n,int i){
        int largest=i,l=2*i+1,r=2*i+2;
        if(l<n && a[l]>a[largest]) largest=l;
        if(r<n && a[r]>a[largest]) largest=r;
        if(largest!=i){
            int t=a[i];a[i]=a[largest];a[largest]=t;
            heapifyMax(a,n,largest);
        }
    }
    static void heapifyMin(int[] a,int n,int i){
        int smallest=i,l=2*i+1,r=2*i+2;
        if(l<n && a[l]<a[smallest]) smallest=l;
        if(r<n && a[r]<a[smallest]) smallest=r;
        if(smallest!=i){
            int t=a[i];a[i]=a[smallest];a[smallest]=t;
            heapifyMin(a,n,smallest);
        }
    }
    static void heapSortInc(int[] a,int n){
        for(int i=n/2-1;i>=0;i--) heapifyMax(a,n,i);
        for(int i=n-1;i>0;i--){
            int t=a[0];a[0]=a[i];a[i]=t;
            heapifyMax(a,i,0);
        }
    }
    static void heapSortDec(int[] a,int n){
        for(int i=n/2-1;i>=0;i--) heapifyMin(a,n,i);
        for(int i=n-1;i>0;i--){
            int t=a[0];a[0]=a[i];a[i]=t;
            heapifyMin(a,i,0);
        }
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int[] a=new int[n];
        int[] b=new int[n];
        for(int i=0;i<n;i++){a[i]=sc.nextInt();b[i]=a[i];}
        heapSortInc(a,n);
        heapSortDec(b,n);
        for(int x:a) System.out.print(x+" ");
        System.out.println();
        for(int x:b) System.out.print(x+" ");
        sc.close();
    }
}

// Q2. Priority Queue using Heap
import java.util.*;
class PQHeap {
    static class HeapPQ{
        ArrayList<Integer> h=new ArrayList<>();
        void insertKey(int x){
```

```
            h.add(x);
            int i=h.size()-1;
            while(i>0){
                int p=(i-1)/2;
                if(h.get(p)<h.get(i)){
                    int t=h.get(p);
                    h.set(p,h.get(i));
                    h.set(i,t);
                    i=p;
                }else break;
            }
        }
        boolean empty(){
            return h.isEmpty();
        }
        int getMax(){
            if(h.isEmpty()) return -1;
            return h.get(0);
        }
        int extractMax(){
            if(h.isEmpty()) return -1;
            int root=h.get(0);
            int last=h.get(h.size()-1);
            h.set(0,last);
            h.remove(h.size()-1);
            int n=h.size(),i=0;
            while(true){
                int l=2*i+1,r=2*i+2,largest=i;
                if(l<n && h.get(l)>h.get(largest)) largest=l;
                if(r<n && h.get(r)>h.get(largest)) largest=r;
                if(largest!=i){
                    int t=h.get(i);
                    h.set(i,h.get(largest));
                    h.set(largest,t);
                    i=largest;
                }else break;
            }
            return root;
        }
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        HeapPQ pq=new HeapPQ();
        for(int i=0;i<n;i++){
            int x=sc.nextInt();
            pq.insertKey(x);
        }
        while(!pq.empty()){
            System.out.print(pq.extractMax()+" ");
        }
        sc.close();
    }
}
```