

import java.io.\*;

import java.math.\*;

import java.text.\*;

import java.util.\*;

import java.util.regex.\*;

public class Solution {

static class SinglyLinkedListNode {

public int data;

public SinglyLinkedListNode next;

public SinglyLinkedListNode(int nodeData) {

this.data = nodeData;

this.next = null;

}

}

static class SinglyLinkedList {

public SinglyLinkedListNode head;

public SinglyLinkedListNode tail;

public SinglyLinkedList() {

this.head = null;

this.tail = null;

}

public void insertNode(int nodeData) {

SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);

if (this.head == null) {

this.head = node;

} else {

this.tail.next = node;

}

this.tail = node;

}

}

public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

while (node != null) {

bufferedWriter.write(String.valueOf(node.data));

node = node.next;

if (node != null) {

bufferedWriter.write(sep);

}

}

}

static boolean compareLists(SinglyLinkedListNode head1, SinglyLinkedListNode head2) {

SinglyLinkedListNode list1=head1;

SinglyLinkedListNode list2=head2;

while(list1 !=null && list2 !=null){

if(list1.data !=list2.data)

return false;

list1=list1.next;

list2=list2.next;

}

if(list1 !=null || list2 !=null)

return false;

return true;

}

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) throws IOException {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int tests = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int testsItr = 0; testsItr < tests; testsItr++) {

SinglyLinkedList llist1 = new SinglyLinkedList();

int llist1Count = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int i = 0; i < llist1Count; i++) {

int llist1Item = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

llist1.insertNode(llist1Item);

}

SinglyLinkedList llist2 = new SinglyLinkedList();

int llist2Count = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int i = 0; i < llist2Count; i++) {

int llist2Item = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

llist2.insertNode(llist2Item);

}

boolean result = compareLists(llist1.head, llist2.head);

bufferedWriter.write(String.valueOf(result ? 1 : 0));

bufferedWriter.newLine();

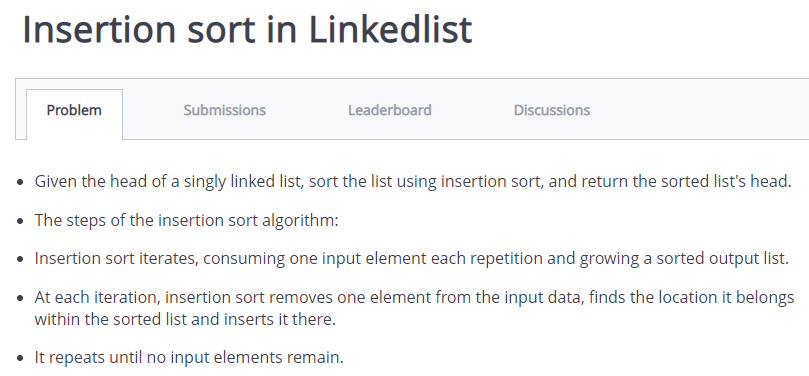
}

bufferedWriter.close();

scanner.close();

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

Node head;

Node sorted;

class Node{

int k;

Node next;

Node(int k){

this.k=k;

}

}

void insert(int d){

Node temp=new Node(d);

temp.next=head;

head=temp;

}

void insertionSort(Node ref){

sorted=null;

Node curr=ref;

while(curr != null){

Node next=curr.next;

sortedInsert(curr);

curr=next;

}

head=sorted;

}

void sortedInsert(Node nn){

if(sorted==null || sorted.k>nn.k){

nn.next=sorted;

sorted=nn;

}

else{

Node curr=sorted;

while(curr.next != null && curr.next.k<nn.k){

curr=curr.next;

}

nn.next=curr.next;

curr.next=nn;

}

}

void print(Node head){

while(head!=null){

System.out.print(head.k+" ");

head=head.next;

}

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

Solution list=new Solution();

for(;;){

int data=in.nextInt();

if(data==-1)

break;

else

list.insert(data);

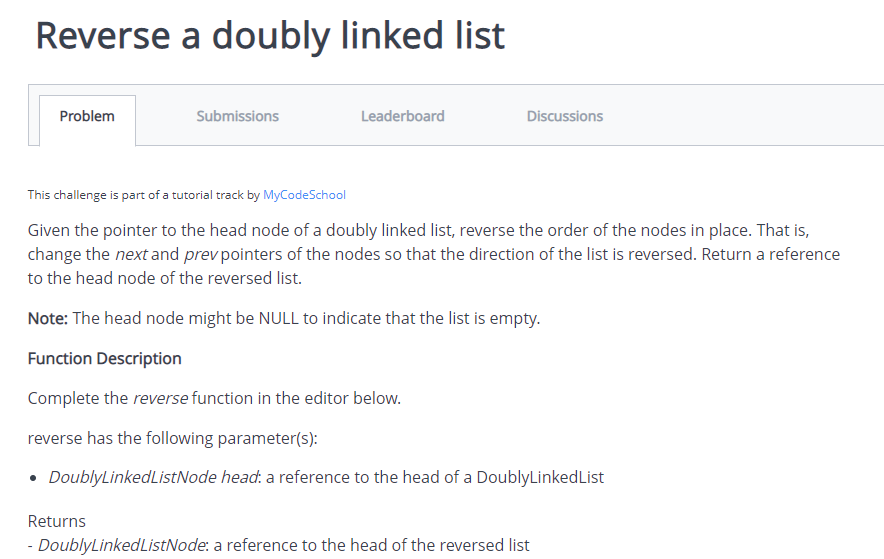
}

list.insertionSort(list.head);

list.print(list.head);

}

}



import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

public class Solution {

static class DoublyLinkedListNode {

public int data;

public DoublyLinkedListNode next;

public DoublyLinkedListNode prev;

public DoublyLinkedListNode(int nodeData) {

this.data = nodeData;

this.next = null;

this.prev = null;

}

}

static class DoublyLinkedList {

public DoublyLinkedListNode head;

public DoublyLinkedListNode tail;

public DoublyLinkedList() {

this.head = null;

this.tail = null;

}

public void insertNode(int nodeData) {

DoublyLinkedListNode node = new DoublyLinkedListNode(nodeData);

if (this.head == null) {

this.head = node;

} else {

this.tail.next = node;

node.prev = this.tail;

}

this.tail = node;

}

}

public static void printDoublyLinkedList(DoublyLinkedListNode node, String sep, BufferedWriter bufferedWriter) throws IOException {

while (node != null) {

bufferedWriter.write(String.valueOf(node.data));

node = node.next;

if (node != null) {

bufferedWriter.write(sep);

}

}

}

public static DoublyLinkedListNode reverse(DoublyLinkedListNode llist) {

// Write your code here

DoublyLinkedListNode temp=llist.next;

llist.next=llist.prev;

llist.prev=temp;

if(temp==null)

return llist;

return reverse(temp);

}

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) throws IOException {

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int t = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int tItr = 0; tItr < t; tItr++) {

DoublyLinkedList llist = new DoublyLinkedList();

int llistCount = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

for (int i = 0; i < llistCount; i++) {

int llistItem = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

llist.insertNode(llistItem);

}

DoublyLinkedListNode llist1 = reverse(llist.head);

printDoublyLinkedList(llist1, " ", bufferedWriter);

bufferedWriter.newLine();

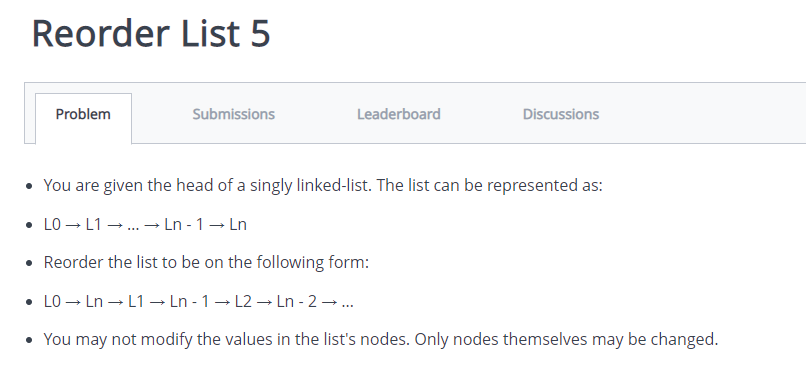
}

bufferedWriter.close();

scanner.close();

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

LinkedList<Integer> list=new LinkedList<>();

int data=0;

while(true){

data=in.nextInt();

if(data==-1)

break;

list.add(data);

}

int m=list.size()-1;

int cnt=0;

for(int i=0;i<list.size();i++){

System.out.print(list.get(i)+" ");

cnt++;

if(cnt==list.size())

break;

System.out.print(list.get(m)+" ");

m--;

cnt++;

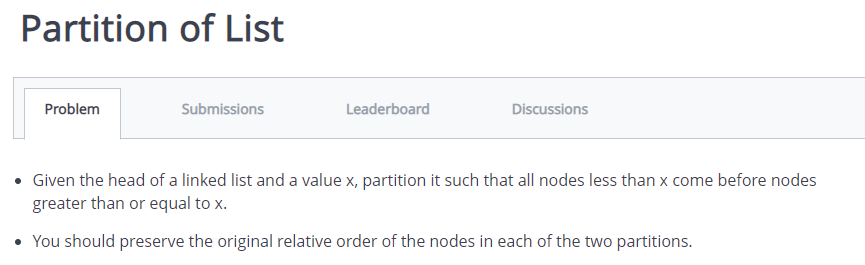
if(cnt==list.size())

break;

}

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

ArrayList<Integer> list=new ArrayList<>();

int data=0,i=0;

while(true){

data=in.nextInt();

if(data==-1)

break;

list.add(data);

}

int k=in.nextInt();

ArrayList<Integer> arr=new ArrayList<>();

while(true){

if(i<list.size() && list.get(i)<k){

arr.add(list.get(i));

list.remove(i);

}

else if(i>=list.size())

break;

else

i++;

}

for(i=0;i<list.size();i++)

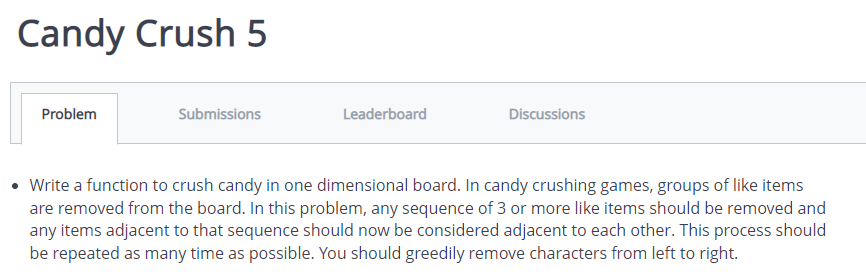
arr.add(list.get(i));

for(int et:arr)

System.out.print(et+" ");

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

static class Pair{

char c;

int ctr;

Pair(char c,int ctr){

this.c=c;

this.ctr=ctr;

}

}

public static String reduced\_String(int k, String s){

if(k==1){

return "";

}

Stack<Pair> st=new Stack<Pair>();

int l=s.length();

int ctr=0;

for(int i=0;i<l;i++){

if(st.size()==0){

st.push(new Pair(s.charAt(i),1));

continue;

}

if(st.peek().c==s.charAt(i)){

Pair p=st.peek();

st.pop();

p.ctr+=1;

if(p.ctr==k){

continue;

}

else{

st.push(p);

}

}

else{

st.push(new Pair(s.charAt(i),1));

}

}

StringBuilder output=new StringBuilder();

while(st.size()>0){

char c=st.peek().c;

int cnt=st.peek().ctr;

while (cnt-- > 0)

output.append(String.valueOf(c));

st.pop();

}

output.reverse();

return output.toString();

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

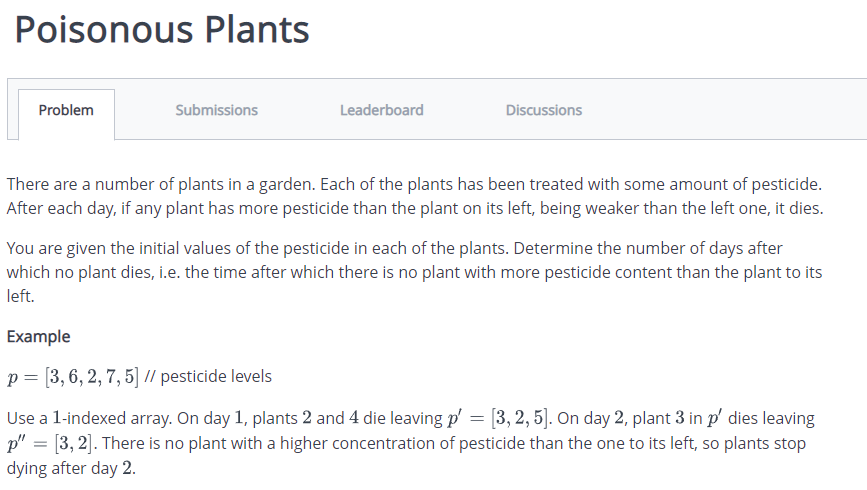
String s=in.nextLine();

String a=reduced\_String(3,s);

System.out.println(a);

}

}



import java.util.ArrayList;

import java.util.Collections;

import java.util.Scanner;

public class Solution {

static Scanner in=new Scanner(System.in);

public static void main(String[]args){

int N=in.nextInt();

int P[]=new int[N+2];

int prev[]=new int[N+2];

int next[]=new int[N+2];

next[0]=1;

prev[N+1]=N;

for(int i=1;i<=N;++i){

P[i]=Integer.parseInt(in.next());

prev[i]=i-1;

next[i]=i+1;

}

ArrayList<Integer> killer=new ArrayList<>();

for(int i=1;i<N;++i){

if(P[i]<P[i+1]){

killer.add(i);

}

}

int day=0;

while(!killer.isEmpty()){

++day;

ArrayList<Integer> nk=new ArrayList<>();

for(int i=killer.size()-1;i>=0;--i){

int k=killer.get(i);

int killed=next[k];

prev[next[killed]]=k;

next[k]=next[killed];

if(!nk.isEmpty() && nk.get(nk.size()-1) == killed)

nk.remove(nk.size()-1);

if(next[k]<=N && P[k]<P[next[k]]){

nk.add(k);

}

}

Collections.reverse(nk);

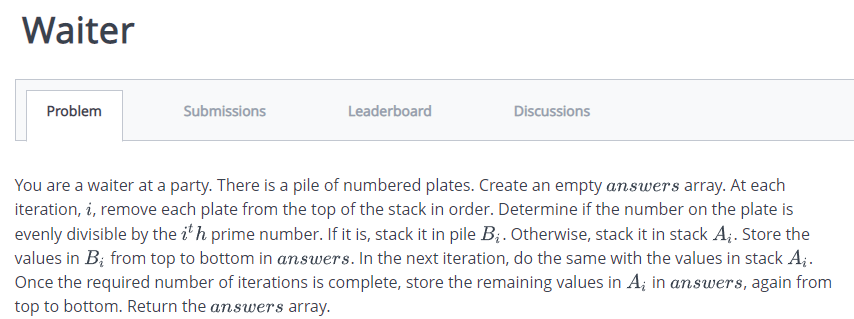
killer=nk;

}

System.out.println(day);

}

}



import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'waiter' function below.

\*

\* The function is expected to return an INTEGER\_ARRAY.

\* The function accepts following parameters:

\* 1. INTEGER\_ARRAY number

\* 2. INTEGER q

\*/

public static List<Integer> waiter(List<Integer> number, int q) {

// Write your code here

Stack<Integer> stack=new Stack<Integer>();

Stack<Integer> st=new Stack<Integer>();

int p=2;

for(int i=0;i<q;i++){

for(int j=0;j<number.size();j++){

if(number.get(j)%p==0){

stack.add(number.get(j));

number.remove(j);

j--;

}

else{

st.add(0,number.get(j));

}

}

number.clear();

number.addAll(st);

st.clear();

p++;

for(int k=2;k<p;k++){

if (p%k==0){

p++;

k=2;

}

}

}

Collections.reverse(number);

stack.addAll(number);

return stack;

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

String[] firstMultipleInput = bufferedReader.readLine().replaceAll("\\s+$", "").split(" ");

int n = Integer.parseInt(firstMultipleInput[0]);

int q = Integer.parseInt(firstMultipleInput[1]);

List<Integer> number = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Integer::parseInt)

.collect(toList());

List<Integer> result = Result.waiter(number, q);

bufferedWriter.write(

result.stream()

.map(Object::toString)

.collect(joining("\n"))

+ "\n"

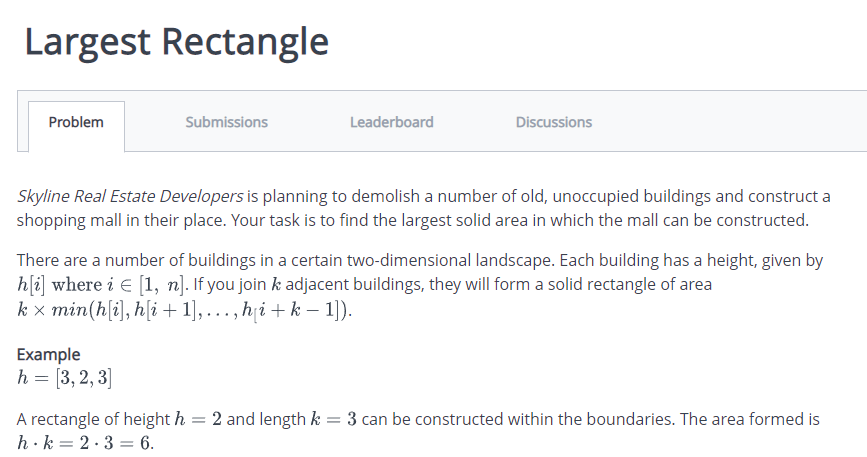
);

bufferedReader.close();

bufferedWriter.close();

}

}



import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.function.\*;

import java.util.regex.\*;

import java.util.stream.\*;

import static java.util.stream.Collectors.joining;

import static java.util.stream.Collectors.toList;

class Result {

/\*

\* Complete the 'largestRectangle' function below.

\*

\* The function is expected to return a LONG\_INTEGER.

\* The function accepts INTEGER\_ARRAY h as parameter.

\*/

public static long largestRectangle(List<Integer> h) {

// Write your code here

long ans=0;

for(int i=0;i<h.size();i++){

long cnt=1;

int cur=h.get(i);

for(int j=i+1;j<h.size();j++){

if(cur>h.get(j))

break;

cnt++;

}

for(int k=i-1;k>=0;k--){

if(cur>h.get(k))

break;

cnt++;

}

long sum=cnt\*cur;

if(sum>ans){

ans=sum;

}

}

return ans;

}

}

public class Solution {

public static void main(String[] args) throws IOException {

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));

BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT\_PATH")));

int n = Integer.parseInt(bufferedReader.readLine().trim());

List<Integer> h = Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))

.map(Integer::parseInt)

.collect(toList());

long result = Result.largestRectangle(h);

bufferedWriter.write(String.valueOf(result));

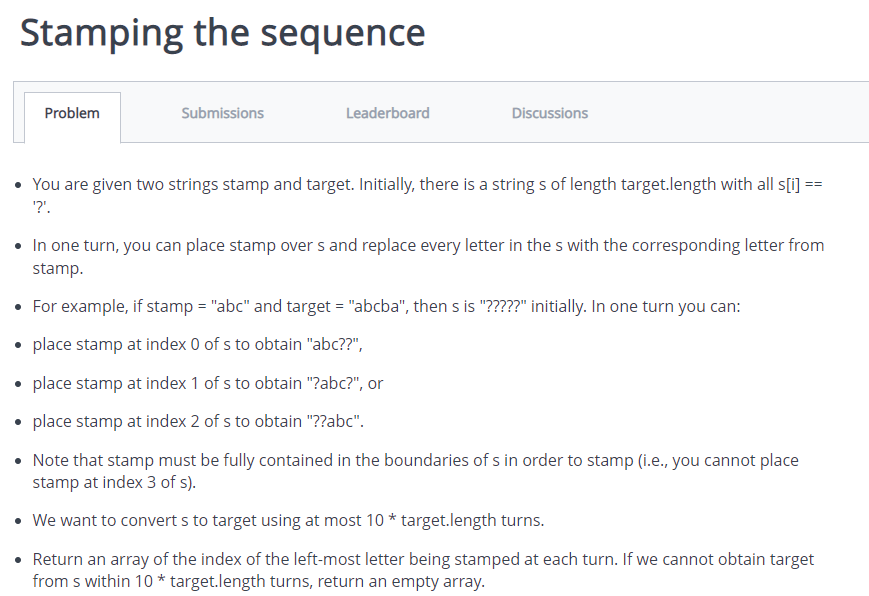
bufferedWriter.newLine();

bufferedReader.close();

bufferedWriter.close();

}

}



import java.util.\*;

import java.util.List;

import java.util.ArrayList;

class Solution{

public static int[] movetoStamp(String st,String tr){

if(st==tr)

return new int[]{0};

char sc[]=st.toCharArray();

char tc[]=tr.toCharArray();

int sl=sc.length,tl=tc.length-sl+1,i,j;

List<Integer> ans=new ArrayList<>();

boolean td=true,sd;

while(td){

for(i=0,td=false;i<tl;i++){

for(j=0,sd=false;j<sl;j++){

if(tc[i+j]=='\*')

continue;

else if(tc[i+j]!=sc[j])

break;

else

sd=true;

}

if(j==sl && sd){

for(j=i,td=true;j<sl+i;j++)

tc[j]='\*';

ans.add(0,i);

}

}

}

for(i=0;i<tc.length;i++)

if(tc[i]!='\*') return new int[]{};

int res[]=new int[ans.size()];

for(i=0;i<ans.size();i++)

res[i]=ans.get(i);

return res;

}

public static void main(String[]args){

Scanner in=new Scanner(System.in);

System.out.print("");

String st=in.nextLine();

System.out.print("");

String tr=in.nextLine();

int res[]=movetoStamp(st,tr);

System.out.print("");

if(res.length==0){

System.out.println("");

}

else{

System.out.print("");

for(int i=0;i<res.length;i++){

System.out.print(res[i]);

if(i<res.length-1)

System.out.print(" ");

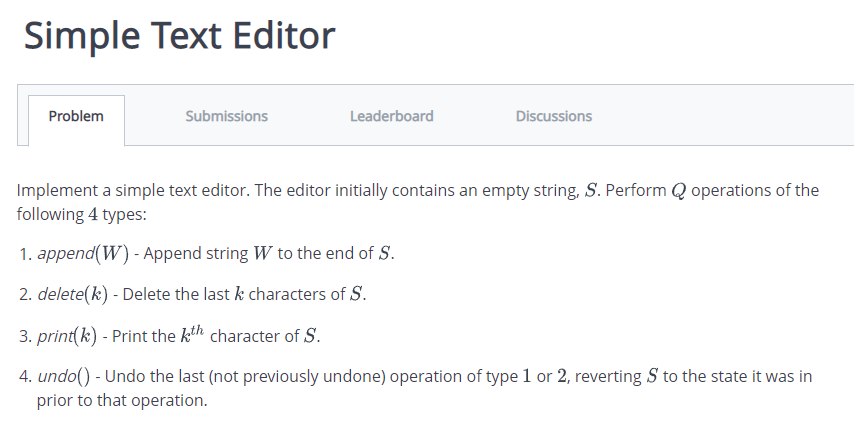
}

System.out.println("");

}

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

int n=in.nextInt();

int t,k;

String last,str;

Stack<String> st=new Stack<>();

while(n-- > 0){

t=in.nextInt();

switch(t){

case 1:

last=st.size()>0 ? st.peek() : "";

str=last+in.next();

st.push(str);

break;

case 2:

k=in.nextInt();

last=st.peek();

str=last.substring(0,last.length()-k);

st.push(str);

break;

case 3:

k=in.nextInt()-1;

if(st.size()>0){

last=st.peek();

String s=String.valueOf(last.charAt(k));

System.out.println(s);

}

break;

case 4:

st.pop();

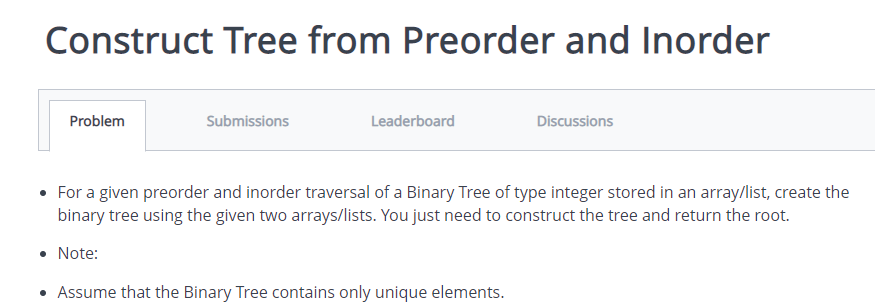
break;

}

}

}

}



import java.io.\*;

import java.util.\*;

class Node{

int val;

Node left,right;

Node(int x){

val = x;

}

}

public class Solution{

public static void main(String[]args){

Scanner in=new Scanner(System.in);

int n=in.nextInt();

int[] preorder=new int[n];

int[] inorder=new int[n];

for(int i=0;i<n;i++){

preorder[i]=in.nextInt();

}

for(int i=0;i<n;i++){

inorder[i]=in.nextInt();

}

Node root=buildBST(preorder,inorder);

printTree(root);

}

public static Node buildBST(int[] preorder,int[] inorder){

if (preorder.length==0 || inorder.length==0){

return null;

}

int rootVal=preorder[0];

Node root=new Node(rootVal);

int rootIndex=0;

for(int i=0;i<inorder.length;i++){

if(inorder[i]==rootVal){

rootIndex=i;

break;

}

}

int[] leftInorder=Arrays.copyOfRange(inorder,0,rootIndex);

int[] rightInorder=Arrays.copyOfRange(inorder,rootIndex+1,inorder.length);

int[] leftPreorder=Arrays.copyOfRange(preorder,1,rootIndex+1);

int[] rightPreorder=Arrays.copyOfRange(preorder,rootIndex+1,preorder.length);

root.left=buildBST(leftPreorder,leftInorder);

root.right=buildBST(rightPreorder,rightInorder);

return root;

}

public static void printTree(Node root){

if(root==null){

return;

}

Queue<Node> queue=new LinkedList<>();

queue.offer(root);

while(!queue.isEmpty()){

int size=queue.size();

for(int i=0;i<size;i++){

Node node=queue.poll();

System.out.print(node.val+" ");

if (node.left != null) {

queue.offer(node.left);

}

if (node.right != null) {

queue.offer(node.right);

}

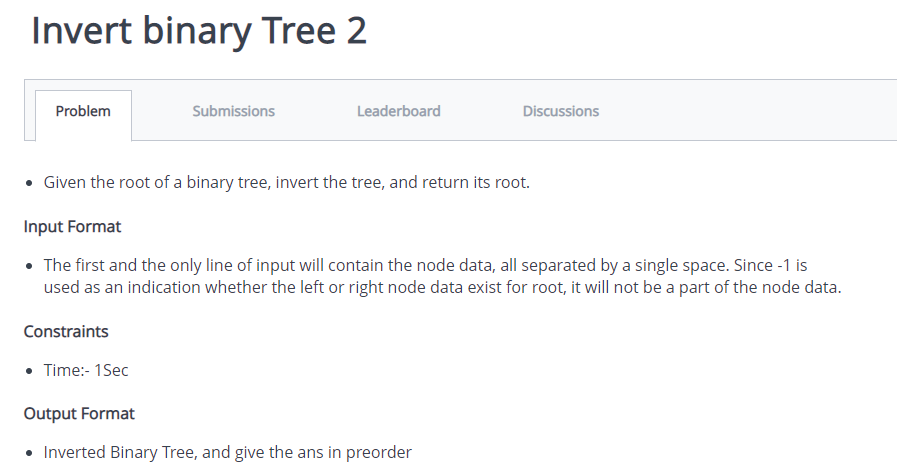
}

System.out.println();

}

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

static class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

static Node invert(Node root){

if(root==null)

return root;

Node left=invert(root.left);

Node right=invert(root.right);

root.left=right;

root.right=left;

return root;

}

static void preorder(Node root){

if(root==null)

return;

System.out.print(root.data+" ");

preorder(root.left);

preorder(root.right);

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

String str[]=in.nextLine().split(" ");

Queue<Node> que=new LinkedList<>();

Node root=new Node(Integer.parseInt(str[0]));

que.offer(root);

for(int i=1;i<str.length;i+=2){

Node parent=que.poll();

if(!str[i].equals("-1")){

Node left=new Node(Integer.parseInt(str[i]));

parent.left=left;

que.offer(left);

}

if(i+1<str.length && !str[i+1].equals("-1")){

Node right=new Node(Integer.parseInt(str[i+1]));

parent.right=right;

que.offer(right);

}

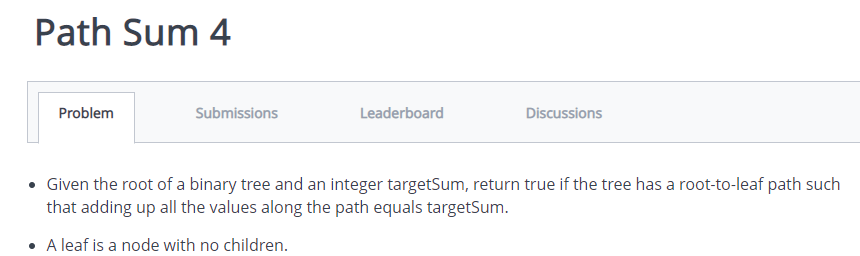
}

invert(root);

preorder(root);

}

}



import java.util.\*;

import java.util.LinkedList;

class Node{

int data;

Node left;

Node right;

Node(int data){

this.data = data;

}

}

public class demo{

static boolean isLeaf(Node root){

return(root.left==null && root.right==null);

}

static int sum=0;

static boolean sum (Node root,int in, int target){

if(root==null && target==0){

return true;

}

else if(root==null)

return false;

in=in+root.data;

if(in == target){

return true;

}

if((sum(root.left,in,target) || sum(root.right,in,target))==true)

return true;

return false;

}

static Node build(String s[])

{

if(s[0]=="-1"||s.length==0) return null;

Node root=new Node(Integer.parseInt(s[0]));

Queue<Node> q=new LinkedList<Node>();

q.add(root);

int i=1;

while(!q.isEmpty() && i<s.length)

{

Node curr=q.poll();

String cval=s[i];

if(!cval.equals("-1"))

{

int h=Integer.parseInt(cval);

curr.left=new Node(h);

q.add(curr.left);

}

i++;

if(i >= s.length)

break;

cval = s[i];

if(!cval.equals("-1"))

{

int h=Integer.parseInt(cval);

curr.right=new Node(h);

q.add(curr.right);

}

i++;

}

return root;

}

public static void main(String[] args) throws Exception

{

Scanner sc=new Scanner(System.in);

String s[]=sc.nextLine().split(" ");

int target = sc.nextInt();

int i;

if(s[0].equals("-1"))

{

System.out.print("0"); return;

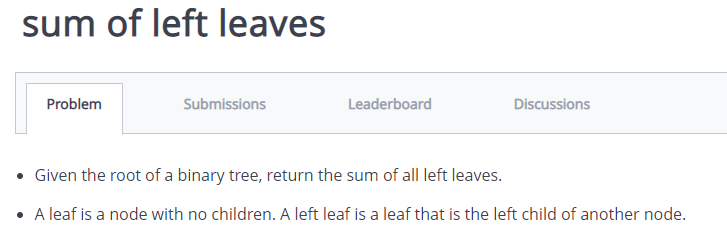
}

Node root=build(s);

System.out.println(sum(root,0,target));

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

static class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

static int sumofLeft(Node root){

if(root==null)

return 0;

if(root.left != null && root.left.left==null && root.left.right==null)

return root.left.data+sumofLeft(root.left)+sumofLeft(root.right);

return sumofLeft(root.left)+sumofLeft(root.right);

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

String str[]=in.nextLine().split(" ");

Queue<Node> que=new LinkedList<>();

Node root=new Node(Integer.parseInt(str[0]));

que.offer(root);

for(int i=1;i<str.length;i+=2){

Node parent=que.poll();

if(!str[i].equals("-1")){

Node left=new Node(Integer.parseInt(str[i]));

parent.left=left;

que.offer(left);

}

if(i+1<str.length && !str[i+1].equals("-1")){

Node right=new Node(Integer.parseInt(str[i+1]));

parent.right=right;

que.offer(right);

}

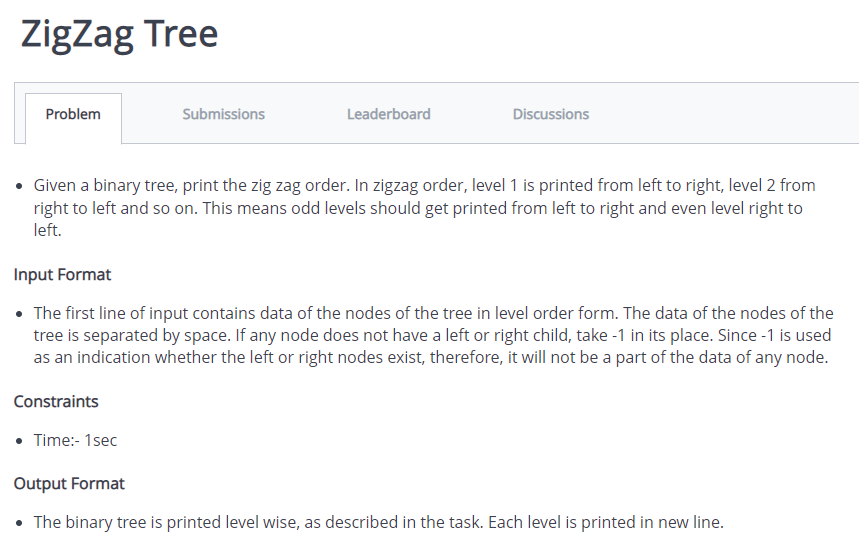
}

int x=sumofLeft(root);

System.out.println(x);

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

public static void zigzag(Node root){

if(root==null){

return;

}

Stack<Node> curr=new Stack<>();

Stack<Node> next=new Stack<>();

next.push(root);

while(!curr.isEmpty() || !next.isEmpty()){

while(!next.isEmpty()){

Node curNode=next.pop();

System.out.print(curNode.data+" ");

if(curNode.left!=null){

curr.push(curNode.left);

}

if(curNode.right!=null){

curr.push(curNode.right);

}

}

System.out.println();

while(!curr.isEmpty()){

Node nexNode=curr.pop();

System.out.print(nexNode.data+" ");

if(nexNode.right!=null){

next.push(nexNode.right);

}

if(nexNode.left!=null){

next.push(nexNode.left);

}

}

System.out.println();

}

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

String str[]=in.nextLine().split(" ");

Queue<Node> que=new LinkedList<>();

Node root=new Node(Integer.parseInt(str[0]));

que.offer(root);

for(int i=1;i<str.length;i+=2){

Node parent=que.poll();

if(!str[i].equals("-1")){

Node left=new Node(Integer.parseInt(str[i]));

parent.left=left;

que.offer(left);

}

if(i+1<str.length && !str[i+1].equals("-1")){

Node right=new Node(Integer.parseInt(str[i+1]));

parent.right=right;

que.offer(right);

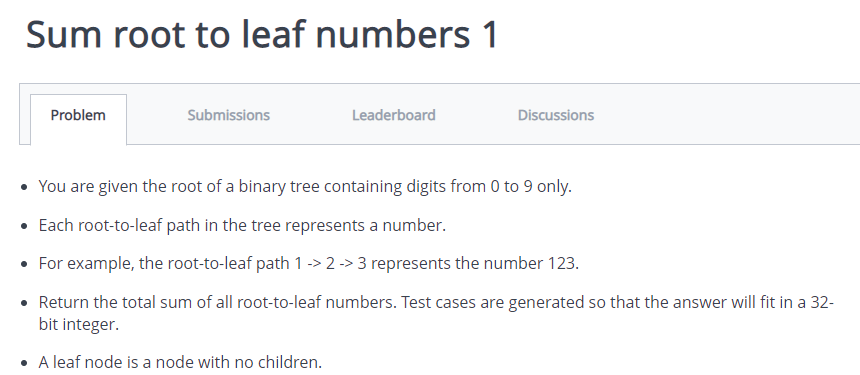
}

}

zigzag(root);

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

static class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

static int sumtoleaf(Node root,int curr){

if(root==null)

return 0;

curr=curr\*10+root.data;

if(root.left==null && root.right==null)

return curr ;

return sumtoleaf(root.left,curr)+sumtoleaf(root.right,curr);

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

String str[]=in.nextLine().split(" ");

Queue<Node> que=new LinkedList<>();

Node root=new Node(Integer.parseInt(str[0]));

que.offer(root);

for(int i=1;i<str.length;i+=2){

Node parent=que.poll();

if(!str[i].equals("-1")){

Node left=new Node(Integer.parseInt(str[i]));

parent.left=left;

que.offer(left);

}

if(i+1<str.length && !str[i+1].equals("-1")){

Node right=new Node(Integer.parseInt(str[i+1]));

parent.right=right;

que.offer(right);

}

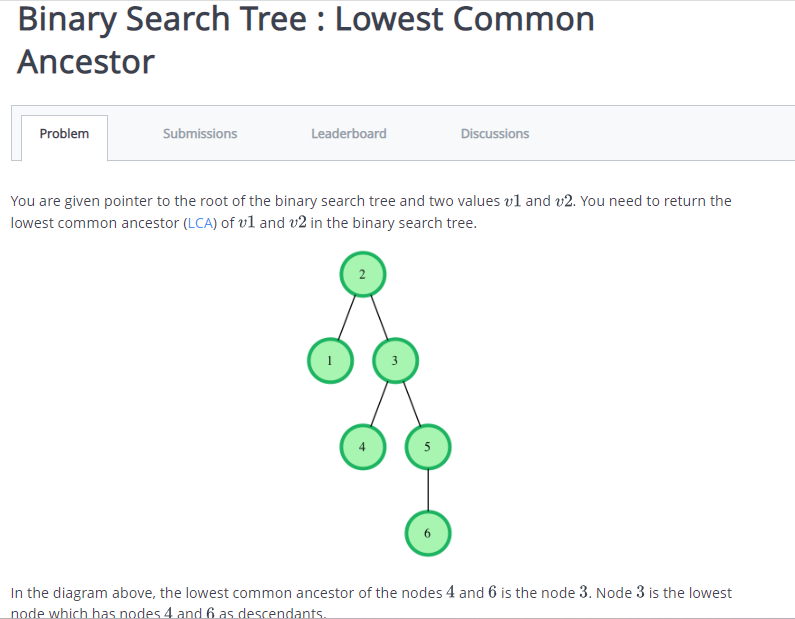
}

int x=sumtoleaf(root,0);

System.out.print(x);

}

}



import java.util.\*;

import java.io.\*;

class Node {

Node left;

Node right;

int data;

Node(int data) {

this.data = data;

left = null;

right = null;

}

}

class Solution {

public static Node lca(Node root, int v1, int v2) {

// Write your code here.

if(root==null)

return null;

if(root.data>v1 && root.data>v2)

return lca(root.left,v1,v2);

if(root.data<v1 && root.data<v2)

return lca(root.right,v1,v2);

return root;

}

public static Node insert(Node root, int data) {

if(root == null) {

return new Node(data);

} else {

Node cur;

if(data <= root.data) {

cur = insert(root.left, data);

root.left = cur;

} else {

cur = insert(root.right, data);

root.right = cur;

}

return root;

}

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int t = scan.nextInt();

Node root = null;

while(t-- > 0) {

int data = scan.nextInt();

root = insert(root, data);

}

int v1 = scan.nextInt();

int v2 = scan.nextInt();

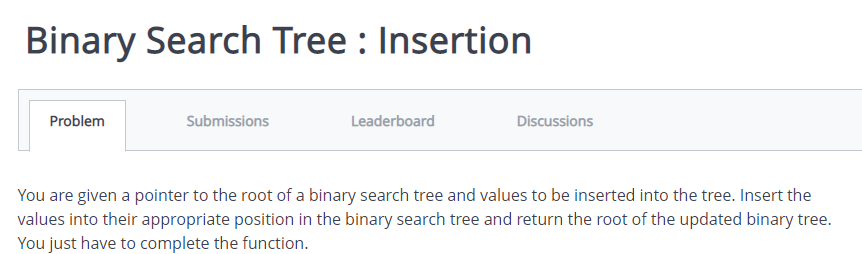
scan.close();

Node ans = lca(root,v1,v2);

System.out.println(ans.data);

}

}



import java.util.\*;

import java.io.\*;

class Node {

Node left;

Node right;

int data;

Node(int data) {

this.data = data;

left = null;

right = null;

}

}

class Solution {

public static void preOrder( Node root ) {

if( root == null)

return;

System.out.print(root.data + " ");

preOrder(root.left);

preOrder(root.right);

}

public static Node insert(Node root,int data) {

if(root==null){

root=new Node(data);

return root;

}

else if(root.data>data){

root.left=insert(root.left,data);

}

else if(root.data<data){

root.right=insert(root.right,data);

}

return root;

}

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

int t = scan.nextInt();

Node root = null;

while(t-- > 0) {

int data = scan.nextInt();

root = insert(root, data);

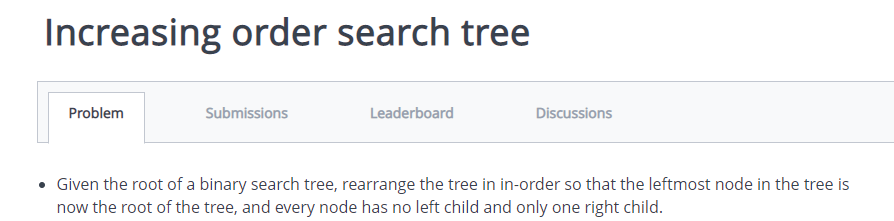
}

scan.close();

preOrder(root);

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static class Node{

int data;

Node left,right;

Node(int data){

this.data=data;

left=right=null;

}

}

public static Node root;

public static void bst(int data){

root=buildBST(root,data);

}

public static Node buildBST(Node root,int key){

if(root==null){

root=new Node(key);

return root;

}

else if(root.data>key){

root.left=buildBST(root.left,key);

}

else if(root.data<key){

root.right=buildBST(root.right,key);

}

return root;

}

public static void inOrder(Node root){

if(root==null){

return;

}

inOrder(root.left);

System.out.print(root.data+" ");

inOrder(root.right);

}

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner in=new Scanner(System.in);

int s=in.nextInt();

int arr[]=new int[s];

for(int i=0;i<s;i++){

arr[i]=in.nextInt();

}

for(int i=0;i<arr.length;i++){

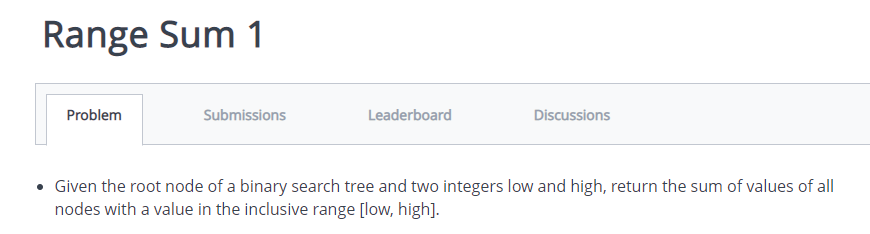
bst(arr[i]);

}

inOrder(root);

}

}



import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named Solution. \*/

Scanner s=new Scanner(System.in);

int n=s.nextInt();

int[] a=new int[n];

for(int i=0;i<n;i++){

a[i]=s.nextInt();

}

int l=s.nextInt();

int h=s.nextInt();

int sum=0;

for(int i=0;i<n;i++){

if(a[i]>=l && a[i]<=h){

sum+=a[i];

}

}

System.out.println(sum);

}

}