1.VERTICAL ORDER PRINT BINARY TREE

You will be given a Binary Tree. Your task is to print the binary tree in Vertical Fashion. The image below
how we define a vertical traversal of a tree.

Input Format

You will be given an Integer N denoting the number of levels in the corresponding tree. On the next line
given (2^N)-1 order integers denoting the level order input for the tree. If at any level any node is absen
will denoted by -1 and every integer other than -1 shows the presence of a node at that level.

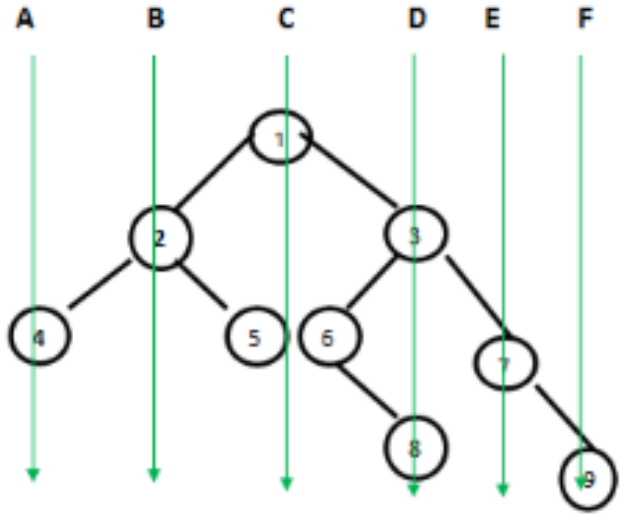
Constraints

```
1 <= Total nodes <= 10^5
```

Output Format

Print all nodes of a given column in the same line.

Vertical Lines



Vertical order traversal is:

A- 4

B- 2

C- 1 5 6

D- 3 8

E- 7

F- 9

Sample Input

```
4 1 2 3 4 5 6 7 -1 -1 -1 -1 -1 8 -1 9 -1 -1 -1 -1
```

Sample Output

```
4 2 1 5 6 3 8 7 9
```

```java
import java.util.*;
import java.util.Map.Entry;
import java.util.TreeMap;
import java.util.Vector;
public class Main {
    static Scanner sc = new Scanner(System.in);
    public static void main(String[] args) {
        Main m = new Main();
        int n = sc.nextInt();
        BinaryTree bt = m.new BinaryTree();
        HashMap<Integer,ArrayList<Integer>> map=new HashMap<>();
        bt.printVerticalOrder(bt.root,map);
        ArrayList<Integer> keys=new ArrayList<>(map.keySet());
        Collections.sort(keys);
        for(int i:keys){
            ArrayList<Integer> list=map.get(i);
            for(int j:list)
                System.out.print(j+" ");
        }
    }
    private class BinaryTree {
        private class Node {
            int data;
            Node left;
            Node right;
        }
        private Node root;
        private int size;
        public BinaryTree() {
        this.root = this.takeInput();
    }
    public Node takeInput() {
        Queue<Node> q = new LinkedList<>();
        Node nn = new Node();
        int val = sc.nextInt();
        nn.data = val;
        root = nn;
        q.add(root);
        while(!q.isEmpty()){
        Node help = q.remove();
        int left = sc.nextInt();
        int right = sc.nextInt();
        if(left != -1){
            Node l = new Node();
            l.data = left;
            help.left = l;
```

```
            q.add(l);
        }
        if(right != -1){
            Node r = new Node();
            r.data=right;
            help.right = r;
            q.add(r);
        }
        }
 return root;
 }
 public void printVerticalOrder(Node root, HashMap<Integer,ArrayList<Integer>>
a){
//  TreeMap<Integer, Vector<Integer> > m
//  = new TreeMap<>();
 int b= 0;
 VOT(root,b, a);
//  for (Entry<Integer, Vector<Integer> > entry :
//  m.entrySet()) {
// //  System.out.println(entry.getValue());
//  a.add(entry.getValue());
//  }
 }
    public void VOT(Node root, int axis, HashMap<Integer,ArrayList<Integer>>
map)
    {
        if(root==null)
            return;

       if(map.containsKey(axis)){
           ArrayList<Integer> list=map.get(axis);
           list.add(root.data);
       }
       else{
           ArrayList<Integer> list=new ArrayList<>();
           list.add(root.data);
           map.put(axis,list);
       }
       VOT(root.left,axis-1,map);
       VOT(root.right,axis+1,map);
     }
 }
}
```

2.Arrays-Intersectionn of two Arrays

Take as input N, the size of array. Take N more inputs and store that in an array. Take N more inputs in another array. Write a function which gives the intersection of two arrays in an ArrayList of intege ArrayList.

Input Format

First line contains N denoting the size of the two arrays. Second line contains N space separated in denoting the elements of the first array. Third line contains N space separated integers denoting th the second array.

Constraints

Length of Arrays should be between 1 to 100000.

Output Format

Display the repeating elements in a comma separated manner enclosed in square brackets. The n be sorted in increasing order.

Sample Input

```
7
1 2 3 1 2 4 1
2 1 3 1 5 2 2
```

Sample Output

```
[1, 1, 2, 2, 3]
```

Explanation

Check which integer is present in both the arrays and add them in an array .Print this array as the ans.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner psc = new Scanner(System.in);
```

```
        int psize = psc.nextInt();
        int parr[] = new int [psize];
        int parr1[] = new int[psize];

        for(int i=0; i<psize;i++){
            parr[i] = psc.nextInt();
        }

        for(int i=0; i<psize;i++){
            parr1[i] = psc.nextInt();
        }

        HashMap<Integer, Integer>map = new HashMap<>();

        for(int i=0; i<psize; i++){
            if(map.containsKey(parr[i])){
                map.put(parr[i],map.get(parr[i])+1);
            }
            else{
                map.put(parr[i],1);
            }
        }

        List<Integer>list = new ArrayList<>();
        for(int i=0; i<psize; i++){
            if(map.containsKey(parr1[i])){
                if(map.get(parr1[i]) > 1)
                    map.put(parr1[i],map.get(parr1[i])-1);
                else
                    map.remove(parr1[i]);
                list.add(parr1[i]);
            }
        }
        Collections.sort(list);
        System.out.println(list);
    }
}
```

3.Exist OR NOT

Given an array arr of n length. You will be given Q queries for the array. Each query contains
to determine whether a number exist in the array or not

Input Format

First line of input contains number of test cases T. First line of each case contains an integer L
of the array. The next line contains L space separated integers. The next line contains an inte
number of queries. The next Q lines contains a number N to be searched in the array.

Constraints

1<=t<=12
1<=L<=10^5 (n=number of elements in array).
-10^5<=A[i]<=10^5 (A[i]=ith element of array).

1<=Q<=10^4
-10^5<=x<=10^5

Output Format

For each Query print "Yes" if the number is present and "No" if it's not.

Sample Input

```
1
6
12 3 -67 67 34 2
4
4
5
67
7
```

Sample Output

```
No
No
Yes
No
```

```java
import java.util.*;
public class Main {
```

```java
    public static void main(String args[]) {
        Scanner psc = new Scanner(System.in);
        int t = psc.nextInt();
        while(t-- > 0){
            int psize = psc.nextInt();
            HashSet<Integer> pset = new HashSet<>();
            for(int i=0; i<psize; i++){
                pset.add(psc.nextInt());
            }

            int pqueries = psc.nextInt();
            for(int i=0; i<pqueries; i++){
                int pquery = psc.nextInt();
                if(pset.contains(pquery)){
                    System.out.println("Yes");
                }
                else{
                    System.out.println("No");
                }
            }
        }
        }
}
```

4.UNLOCK

Shekhar is a bomb defusal specialist. He once encountered a bomb that can be defused on
He is given a number N and a number K. And he is also given permutation of first N natural nu
code is the largest permutation possible by doing atmost K swaps among a pair of the given
him to find the final permutation.

Input Format

First line contains an integer N and an integer k. The next line contains N space separated in
given permutation.

Constraints

1 <= n <= 10^5
1<= K <= 10^9

Output Format

The final permutation of the numbers with every number separated by a space with other nu

Sample Input

```
5 2
3 4 1 2 5
```

Sample Output

```
5 4 3 2 1
```

Explanation

First we can swap 5 with 3 which gives us 5 4 1 2 3 and then we can swap 3 and 1 which gives us 5 4 3 2 1.

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
```

```
        Scanner psc = new Scanner(System.in);
        int size = psc.nextInt();
        int pk = psc.nextInt();
        int[]parr = new int[size];
        TreeMap<Integer,Integer>map = new TreeMap<>();
        for(int i=0; i<size;i++){
            parr[i] = psc.nextInt();
            map.put(parr[i],i);
        }

        for(int i=0; i<size && pk > 0; i++){
            if(parr[i] != map.lastKey()){
                int temp = parr[i];
                parr[i] = map.lastKey();
                parr[map.get(map.lastKey())] = temp;
                map.put(temp,map.get(map.lastKey()));
                pk--;
            }
            map.remove(map.lastKey());
        }

        for(int i=0; i<parr.length;i++){
            System.out.print(parr[i]+" ");
        }


    }
}
```

5. Highest Frequency(Hashing)

Given an array find the number which comes with maximum frequency. It must work in O(n)

Input Format

Enter the size of the array N and add N more numbers and store in an array

Constraints

$1 <= N <= 10^7$
$-10^9 <= A[i] <= 10^9$

Output Format

Display the number with the maximum frequency.

Sample Input

```
5
1 2 2 2 3
```

Sample Output

```
2
```

Explanation

2 has the highest frequency in the array i.e. 3.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner psc = new Scanner(System.in);
        int psize = psc.nextInt();
        int []parr = new int[psize];
        for(int i=0; i<parr.length;i++){
            parr[i] = psc.nextInt();
        }
```

```
        HashMap<Integer,Integer>map = new HashMap<>();
        int max = 0;
        int max_num = Integer.MIN_VALUE;
        for(int i=0; i<psize;i++){
            if(map.containsKey(parr[i])){
                map.put(parr[i], map.get(parr[i])+1);
                if(max < map.get(parr[i])){
                    max = map.get(parr[i]);
                    max_num = parr[i];
                }
            }
            else{
                map.put(parr[i], 1);
                if(max < 1){
                    max = 1;
                    max_num = parr[i];
                }
            }
        }

        System.out.println(max_num);
    }
}
```

6.SubArray With Distinct Elements

Given an array, the task is to calculate the sum of lengths of contiguous subarrays having all elements

Input Format

An integer n denoting size of array followed by n integers

Constraints

```
1<=N<=10^5
```

Output Format

The answer mod 10^9+7

Sample Input

```
5
1 2 3 2 3
```

Sample Output

```
16
```

Explanation

Sub arrays of length 1 are=>{1},{2},{3},{2},{3}. =>ans = ans+ 5*1.

Sub arrays of length 2 are=>{1,2},{2,3},{3,2},{2,3}. =>ans = ans+ 4*2.

Sub arrays of length 3 are=>{1,2,3},{2,3,2},{3,2,3}. =>ans = ans+ 1*3. We add only 1 subarray of length 3 because, out of 3 subarrays of length 3, only 1 is unique

Sub arrays of length 4 are=>{1,2,3,2},{2,3,2,3}. =>ans = ans+ 0*4. We add 0 because, all subarrays of length 4, have duplicate elements.

Sub arrays of length 5 are=>{1,2,3,2,3}. =>ans = ans+ 0*4. We add 0 because, all subarrays of length 5, have duplicate elements.

Hence the answer is=>5+8+3=>16

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner psc = new Scanner(System.in);
        int pn = psc.nextInt();
        int []parr = new int[pn];
        for(int i=0; i<parr.length;i++){
            parr[i] = psc.nextInt();
        }

        Set<Integer>st = new HashSet<>();
        int p = 0, ans = 0;
        for(int i=0; i<parr.length; i++){
            while(p<parr.length && !st.contains(parr[p])){
                st.add(parr[p]);
                p++;
            }
            ans+=((p-i)*(p-i+1))/2;
            st.remove(parr[i]);
        }
        System.out.println(ans);
    }
}
```

7.Merge K Sorted Array

Given K sorted arrays each with N elements merge them and output the sorted array

Input Format

First line contains 2 space separated integers K and N.
Next lines contain K*N space separated integers

Constraints

Elements of array <= |10^15|
N <= 10^5
K <= 10

Output Format

Single line consisting of space separated numbers

Sample Input

```
3 4
1 3 5 7
2 4 6 8
0 9 10 11
```

Sample Output

```
0 1 2 3 4 5 6 7 8 9 10 11
```

Explanation

If we were to combine all the arrays into one and then sort it , the output would be as above.

```
import java.util.*;
public class Main {
```

```java
    public static void main (String args[]) {
        Scanner psc = new Scanner(System.in);
        int ArrCnt = psc.nextInt();
        int size = psc.nextInt();
        int [][] arrays = new int[ArrCnt][size];
        for(int i=0; i<ArrCnt;i++){
            for(int j=0; j<size;j++){
                arrays[i][j] = psc.nextInt();
            }
        }

        PriorityQueue<int []>pq = new PriorityQueue<>(new Comparator<int[]>(){
            @Override
            public int compare(int [] a1, int [] a2){
                return arrays[a1[0]][a1[1]] - arrays[a2[0]][a2[1]];
            }
        });

        for(int i=0; i<ArrCnt; i++){
            int [] arr = {i,0};
            pq.add(arr);
        }

        int []ans = new int[ArrCnt*size];
        int i=0;
        while(!pq.isEmpty()){
            int[]arr = pq.poll();
            // System.out.println(arr[0]+" "+arr[1]);
            ans[i] = arrays[arr[0]][arr[1]];
            if(arr[1] < size-1){
                arr[1]++;
                pq.add(arr);
            }
            i++;
        }

        for(int j=0; j<ans.length;j++){
            System.out.print(ans[j]+" ");
        }
    }
}
```

8.Tree Top View

Given a binary tree , print the nodes in left to right manner as visible from above the tree

Input Format

Level order input for the binary tree will be given.

Constraints

No of nodes in the tree can be less than or equal to 10^7

Output Format

A single line containing space separated integers representing the top view of the tree

Sample Input

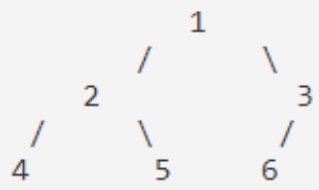
```
1 2 3 4 5 6 -1 -1 -1 -1 -1 -1 -1
```

Sample Output

```
4 2 1 3
```

Explanation

The tree looks like



When viewed from the top , we would see the nodes 4, 2, 1 and 3.

```
import java.util.*;
import java.io.*;
```

```java
public class Main {
    public static void main(String args[]) throws Exception{
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        String[] arr=br.readLine().split(" ");
        BinaryTree bt=new BinaryTree(arr);

        bt.topview();

    }
}

class BinaryTree
{
    private class Node
    {
        int data;
        Node left,right;
        Node(int data)
        {
            this.data=data;
        }
    }

    Node root;

    BinaryTree(String[] arr)
    {
        Queue<Node> queue=new LinkedList<Node>();
        construct(arr,0,queue);
    }

    TreeMap<Integer,Integer>map = new TreeMap<>();
    public void topview()
    {
        verticalTraversalTop(this.root,0);
        Set<Integer>keyset = map.keySet();
        // System.out.println("------------------");
        for(int key : keyset){

            System.out.print(map.get(key)+" ");
        }
    }

   private void verticalTraversalTop(Node root, int axis) {
       if(root.data == -1)
            return;
```

```
        // System.out.println(root.data+" "+axis);
       if(!map.containsKey(axis)){
           map.put(axis,root.data);
       }

       verticalTraversalTop(root.left,axis-1);
       verticalTraversalTop(root.right, axis+1);
   }


    private void construct(String[] arr,int ind,Queue<Node> queue)
    {
        if(ind>=arr.length)
        return;
        if(queue.size()==0)
        {
            Node nn=new Node(Integer.parseInt(arr[ind]));
            this.root=nn;
            queue.add(nn);
        }
        else
        {
            Node parent=queue.peek();
                if(parent.data!=-1){
                if(parent.left==null)
                {
                    parent.left=new Node(Integer.parseInt(arr[ind]));
                    queue.add(parent.left);
                }
                else
                {
                    if(parent.right==null)
                    {
                    parent.right=new Node(Integer.parseInt(arr[ind]));
                    queue.add(parent.right);
                    queue.poll();
                    }
               }
               }
               else
               {
                   queue.poll();
                   ind--;
               }
        }
        construct(arr,ind+1,queue);
    }

    public void display()
```

```
    {
        display_tree(this.root);
    }

    private void display_tree(Node root)
    {
        if(root==null)
        return;
        String str=root.data+"";
        if(root.left!=null)
        {
            str=root.left.data+" <= "+str;
        }
        else
        {
            str="END <= "+str;
        }

        if(root.right!=null)
        {
            str=str+" => "+root.right.data;
        }
        else
        {
            str=str+" => END";
        }
        System.out.println(str);
        display_tree(root.left);
        display_tree(root.right);

    }


}
```

9.Tree Bottom View

Given a binary tree , print the nodes in left to right manner as visible from below the

Input Format

Level order input for the binary tree will be given.

Constraints

No of nodes in the tree can be less than or equal to 10^7

Output Format

A single line containing space separated integers representing the bottom view of t

Sample Input

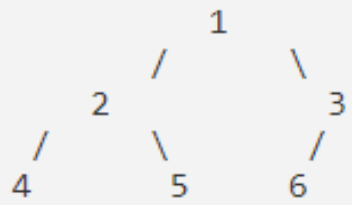
```
1 2 3 4 5 6 -1 -1 -1 -1 -1 -1 -1
```

Sample Output

```
4 2 6 3
```

Explanation

The tree looks like



(Note that 5 and 6 are at the same position so we consider the right one to lower)

```
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String args[]) throws Exception{
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        String[] arr=br.readLine().split(" ");
        BinaryTree bt=new BinaryTree(arr);

        bt.bottomView();

    }
}

class BinaryTree
{
    private class Node
    {
        int data;
        Node left,right;
        Node(int data)
        {
            this.data=data;
        }
    }

    Node root;

    BinaryTree(String[] arr)
    {
        Queue<Node> queue=new LinkedList<Node>();
        construct(arr,0,queue);
    }

    TreeMap<Integer, List<Integer>>map = new TreeMap<>();
    public void bottomView()
    {
        bottomView(this.root, 0, 0);
        Set<Integer>keyset = map.keySet();
        for(int key : keyset){
            System.out.print(map.get(key).get(0)+" ");
        }
    }

    int deep = 0;
   private void bottomView(Node root, int axis, int deep) {
```

```
       if(root.data == -1)
            return;
        List<Integer>ll = new ArrayList<>();
        ll.add(root.data);
        ll.add(deep);
        if(map.containsKey(axis) && deep >= map.get(axis).get(1) ||
!map.containsKey(axis)){
            map.put(axis, ll);
        }



        bottomView(root.left, axis-1, deep+1);
        bottomView(root.right,axis+1, deep+1);

    }


    private void construct(String[] arr,int ind,Queue<Node> queue)
    {
        if(ind>=arr.length)
        return;
        if(queue.size()==0)
        {
            Node nn=new Node(Integer.parseInt(arr[ind]));
            this.root=nn;
            queue.add(nn);
        }
        else
        {
            Node parent=queue.peek();
                if(parent.data!=-1){
                if(parent.left==null)
                {
                    parent.left=new Node(Integer.parseInt(arr[ind]));
                    queue.add(parent.left);
                }
                else
                {
                    if(parent.right==null)
                    {
                    parent.right=new Node(Integer.parseInt(arr[ind]));
                    queue.add(parent.right);
                    queue.poll();
                    }
               }
               }
               else
               {
```

```
                    queue.poll();
                    ind--;
                }
            }
            construct(arr,ind+1,queue);
        }

        public void display()
        {
            display_tree(this.root);
        }

        private void display_tree(Node root)
        {
            if(root==null)
            return;
            String str=root.data+"";
            if(root.left!=null)
            {
                str=root.left.data+" <= "+str;
            }
            else
            {
                str="END <= "+str;
            }

            if(root.right!=null)
            {
                str=str+" => "+root.right.data;
            }
            else
            {
                str=str+" => END";
            }
            System.out.println(str);
            display_tree(root.left);
            display_tree(root.right);

        }


}
```

10. Find Kth Largest number

Find the kth largest element in an unsorted array.
Note: It is the kth largest element in the sorted order, not the kth distinct element.

Input Format

First line contains integer n as size of array. Second line contains the value of k. Third line cont
element of array.

Constraints

1 ≤ k ≤ array's length.

Output Format

Print the kth largest element as output.

Sample Input

```
6
2
3 2 1 5 6 4
```

Sample Output

```
5
```

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner psc = new Scanner(System.in);
        int psize = psc.nextInt();
        int pk = psc.nextInt();
        int []parr = new int[psize];
        for(int i=0; i<psize;i++){
```

```
            parr[i] = psc.nextInt();
        }
        PriorityQueue<Integer>pq = new PriorityQueue<>();
        for(int i=0; i<pk;i++){
            pq.add(parr[i]);
        }

        for(int i=pk; i<parr.length;i++){
            if(parr[i] > pq.peek()){
                pq.poll();
                pq.add(parr[i]);
            }
        }
        System.out.println(pq.peek());
    }
}
```

11. Sort Game

Sanju needs your help! He gets a list of employees with their salary. The maximum salary is 100

Sanju is supposed to arrange the list in such a manner that the list is sorted in decreasing orde
two employees have the same salary, they should be arranged in lexicographical manner such
contains only names of those employees having salary greater than or equal to a given numb

Help Sanju prepare the same!

Input Format

On the first line of the standard input, there is an integer x. The next line contans integer N, de
of employees. N lines follow, which contain a string and an integer, denoting the name of the e
salary.

Constraints

1 <= N <= 10^5 1 <= | Length of the name | <= 100 1 <= x, salary <= 100

Output Format

You must print the required list.

Sample Input

```
79
4
Eve 78
Bob 99
Suzy 86
Alice 86
```

Sample Output

```
Bob 99
Alice 86
Suzy 86
```

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
```

```
        // Your Code Here
        Scanner psc = new Scanner(System.in);
        int pminSal = psc.nextInt();
        int psize = psc.nextInt();
        PEmploy[] employees = new PEmploy[psize];
        for(int i=0; i<psize;i++){
            String name = psc.next();
            int sal = psc.nextInt();
            employees[i] = new PEmploy(name, sal);
        }

        Arrays.sort(employees, new Comparator<PEmploy>(){
            @Override
            public int compare(PEmploy p1, PEmploy p2){
                if(p1.sal == p2.sal){
                    return p1.name.compareTo(p2.name);
                }
                else{
                    return  p2.sal - p1.sal;
                }
            }
        });

        for(int i=0; i<employees.length;i++){
            if(employees[i].sal >= pminSal)
                System.out.println(employees[i]);
        }

    }

    static class PEmploy{
        String name;
        int sal;

        public PEmploy(String name, int sal){
            this.name = name;
            this.sal = sal;
        }

        public String toString(){
            return this.name+" "+this.sal;
        }
    }
}
```

12. Hostel Visit

Dean of MAIT is going to visit Hostels of MAIT. As you know that he is a very busy person so he decided to vis
first "K" nearest Hostels. Hostels are situated in 2D plane. You are given the coordinates of hostels and you ha
answer the Rocket distance of Kth nearest hostel from origin (Dean's place) .

Input Format

First line of input contains Q Total no. of queries and K There are two types of queries:

first type : 1 x y For query of 1st type, you came to know about the co-ordinates (x , y) of newly constructed
hostel. second type : 2 For query of 2nd type, you have to output the Rocket distance of Kth nearest hostel t
now.

The Dean will always stay at his place (origin). It is gauranted that there will be atleast k queries of type 1 b
first query of type 2.

Rocket distance between two points (x2 , y2) and (x1 , y1) is defined as $(x2 - x1)^2 + (y2 - y1)^2$

Constraints

```
1 < = k < = Q < = 10^5
-10^6 < = x , y < = 10^6
```

Output Format

For each query of type 2 output the Rocket distance of Kth nearest hostel from Origin.

Sample Input

```
9 3
1 10 10
1 9 9
1 -8 -8
2
1 7 7
2
1 6 6
1 5 5
2
```

Sample Output

```
200
162
98
```

Explanation

Here , No of queries = n = 9
k = 3

We have to print the kth distance from the hotel.

We are calculating and storing the rocket distance here i.e. (x2-x1)^2 + (y2-y1)^2 ... basically the cartesian distance but without the squareroot.

First integer of each input defines the query type. 1 means take the coordinates input and 2 means display the kth distance so far.

Iteration 1 :
First we get 1 10 10
Distance = 10^2 + 10^2 = 200
We store it in our data structure. Lets call it A.
A = { 200 }

Iteration 2 :
1 9 9
Distance = 9^2 + 9^2 = 162
A = { 162 , 200 }

Iteration 3 :
1 -8 -8
Distance = (-8)^2 + (-8)^2 = 168
A = { 128, 162 , 200 }

Iteration 4 :
2
A = { 128, 162 , 200 }
Time to print the 3rd nearest distance (k=3)
Output : 200

Iteration 5 :
1 7 7
Distance = 7^2 + 7^2 = 98
A = { 98, 128, 162 , 200 }

Iteration 6 :

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner psc = new Scanner(System.in);
        int pn = psc.nextInt();
        int pk = psc.nextInt();
        PriorityQueue<Long>pq = new
PriorityQueue<>(Collections.reverseOrder());
        while(pn-- > 0){
            int qtype = psc.nextInt();
            if(qtype == 1){
                long n1 = psc.nextInt();
                long n2 = psc.nextInt();
                pq.add(n1*n1+n2*n2);
                if(pq.size() > pk){
                    pq.poll();
                }
            }
            if(qtype == 2){
                // List<Long>list = new ArrayList<>(tset);
                System.out.println(pq.peek());
            }
            // System.out.println(qtype);
        }
    }
}
```

13. Top K most Frequent Numbers in a Stream

Given an array of n numbers. Your task is to read numbers from the array and keep at-most K numbers at the top (according to their decreasing frequency) every time a new number is read. We basically need to print top k numbers sorted by frequency when input stream has included k distinct elements, else need to print all distinct elements sorted by frequency. If frequency of two numbers are same then print them in increasing order.

Input Format

First line contains integer t as number of test cases. Each test case contains following input. First line contains integer n and k, n represents the length of the array and next line contains n space separated integers.

Constraints

```
1 < t < 100 1< n < 1000
```

Output Format

Print top k running stream.

Sample Input

```
1
5 2
5 1 3 5 2
```

Sample Output

```
5 1 5 1 3 5 1 5 1
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner psc = new Scanner(System.in);
        int ptcase = psc.nextInt();
        while(ptcase-- >0){
            int pn = psc.nextInt();
            int pk= psc.nextInt();
            int[] parr = new int[pn];
            for (int i = 0; i < parr.length; i++) {
                parr[i] = psc.nextInt();
            }
            TopKEle(parr,pk);
        }
    }
    public static int isValid(int[] parr, int pres) {
```

```
        for (int i = 0; i < parr.length; i++)
            if (parr[i] == pres)
                return i;
        return -1;
    }

   public static void TopKEle(int[] parr,int pk) {
        int length=parr.length;
       int[] pnum = new int[pk + 1];
       HashMap<Integer, Integer> map = new HashMap<>();
       for (int i = 0; i < pk + 1; i++){
           map.put(i, 0);
        }

        for (int j = 0; j < length; j++) {
            if (map.containsKey(parr[j]))
                map.put(parr[j], map.get(parr[j]) + 1);
            else
                map.put(parr[j], 1);
            pnum[pk] = parr[j];
            int i = isValid(pnum, parr[j]);
            i -= 1;
            while (i >= 0) {
                if (map.get(pnum[i]) < map.get(pnum[i + 1])) {
                    int tem = pnum[i];
                    pnum[i] = pnum[i + 1];
                    pnum[i + 1] = tem;
                }
                else if ((map.get(pnum[i]) == map.get(pnum[i + 1])) &&
(pnum[i] > pnum[i + 1])) {
                    int temp = pnum[i];
                    pnum[i] = pnum[i + 1];
                    pnum[i + 1] = temp;
                }

                else
                    break;
                i -= 1;
            }
            for (int k = 0; k < pk && pnum[k] != 0; ++k)
                System.out.print(pnum[k] + " ");
        }
        System.out.println();
    }


}
```

14.String Sort

Nishant is a very naughty boy in Launchpad Batch. One day he was playing with strings, and ran them all. Your task is to help Nishant Sort all the strings (lexicographically) but if a string is preser a prefix in another string, then string with longer length should come first. Eg bat, batman are 2 s string bat is present as a prefix in Batman - then sorted order should have - Batman, bat.

Input Format

N(integer) followed by N strings.

Constraints

N<=1000

Output Format

N lines each containing one string.

Sample Input

```
3
bat
apple
batman
```

Sample Output

```
apple
batman
bat
```

Explanation

In mathematics, the lexicographic or lexicographical order (also known as lexical order, dictionary order, alphabetical order or lexicographic(al) product) is a generalization of the way words are alphabetically ordered based on the alphabetical order of their component letters.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner psc = new Scanner(System.in);
        int pn = psc.nextInt();
        String []parr= new String[pn];
        for(int i=0; i<parr.length; i++){
            parr[i] = psc.next();
        }

        PriorityQueue<String>pq = new PriorityQueue<>(new
Comparator<String>(){
            @Override
            public int compare(String s1, String s2){
                if(s1.length() < s2.length () &&
s2.substring(0,s1.length()).equals(s1)){
                    return 1;
                }
                else if(s2.length() < s1.length () &&
s1.substring(0,s2.length()).equals(s2)){
                    return -1;
                }
                else
                    return s1.compareTo(s2);
            }
        });

        for(int i=0; i<parr.length; i++){
            pq.add(parr[i]);
        }

        for(int i=0; i<parr.length; i++){
            System.out.println(pq.remove());
        }

    }
}
```

15.Median in a Stream of running Integers

You are given a running data stream of n integers. You read all integers from that running data s
effective median of elements read so far in efficient way. All numbers are unique in the datastre
integer part of the median.

Input Format

First line contains integer t as number of test cases. Each test case contains following input. Firs
integer n which represents the length of the data stream and next line contains n space separa

Constraints

$1 < t < 100$
$1 < n < 10000$

Output Format

Print the effective median of running data stream..

Sample Input

```
1
6
5 15 1 3 2 8
```

Sample Output

```
5 10 5 4 3 4
```

Explanation

Iteration 1 : Array = {5} Median = 5

Iteration 2 : Array = {5,15} Median = (5+15)/2 = 10

Iteration 3 : Array = {1,5,15} Median = 5

Iteration 4 : Array = {1,3,5,15} Median = (3+5)/2 = 4
Iteration 5 : Array = {1,2,3,5,15} Median = 3
Iteration 6 : Array = {1,2,3,5,8,15} Median = (3+5)/2 = 4

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner pavansc=new Scanner(System.in);
        int pavantest=pavansc.nextInt();
        while(pavantest-->0){
            int pavann=pavansc.nextInt();
            int []pavanarr=new int[pavann];
            for(int i=0;i<pavann;i++){
                pavanarr[i]=pavansc.nextInt();
            }
            printMedian(pavanarr);
            System.out.println();

        }
    }
    public static void printMedian(int pavanarr[]) {
        PriorityQueue<Integer> pavanmin = new PriorityQueue<>();
        PriorityQueue<Integer> pavanmax = new PriorityQueue<>();
        for (int i = 0; i < pavanarr.length; i++) {
            pavanmax.add(-1 * pavanarr[i]);
            pavanmin.add(-1 * pavanmax.poll());
            if (pavanmin.size() > pavanmax.size()) {
                pavanmax.add(-1 * pavanmin.poll());
            }
            if (pavanmin.size() != pavanmax.size()) {
                System.out.print(-1 * pavanmax.peek()+" ");
            }
            else {
                System.out.print(((pavanmin.peek() - pavanmax.peek()) / 2)+"
");
            }
        }
    }


}
```

16.Mapped String

We are given a hashmap which maps all the letters with number. Given 1 is mapped with A, 2 is mapped w
B.....26 is mapped with Z. Given a number, you have to print all the possible strings.

Input Format

A single line contains a number.

Constraints

Number is less than 10^6

Output Format

Print all the possible strings in sorted order in different lines.

Sample Input

```
123
```

Sample Output

```
ABC
AW
LC
```

Explanation

'1' '2' '3' = ABC
'1' '23' = AW
'12' '3' = LC

```
import java.util.*;
public class Main {
static String arr[] = { "", "A", "B", "C", "D", "E",
"F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P",
"Q",
"R", "S", "T", "U", "V", "W", "X", "Y", "Z" };
public static void main(String[] args) {
Scanner pavansc = new Scanner(System.in);
int n = pavansc.nextInt();
String s = ""+n;
Mappedstring(s, "");
```

```
}
public static void Mappedstring(String s, String ans)
{
if (s.length() == 0) {
System.out.println(ans);
return;
}
char s1 = s.charAt(0);
Mappedstring(s.substring(1), ans + arr[s1-'0']);
if (s.length() >= 2) {
String s2 = s.substring(0, 2);
int num = Integer.parseInt(s2);
if (num <= 27) {
Mappedstring(s.substring(2), ans +
arr[num]);
}
}
}
}
```

17. Merge K Sorted Lists

Given K sorted linked lists of equal sizes. The task is to merge them in such a way that after merg
single sorted linked list.

Input Format

First line contains 2 space separated integers K and N.
Next lines contain N*K space separated integers

Constraints

None

Output Format

Single line consisting of space separated numbers

Sample Input

```
3 4
5 5 5 9 3 13 14 17 1 8 11 18
```

Sample Output

```
1 3 5 5 5 8 9 11 13 14 17 18
```

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner psc = new Scanner(System.in);
        int ArrCnt = psc.nextInt();
        int size = psc.nextInt();
        int [][] arrays = new int[ArrCnt][size];
        for(int i=0; i<ArrCnt;i++){
            for(int j=0; j<size;j++){
                arrays[i][j] = psc.nextInt();
            }
        }
```

```
        PriorityQueue<int []>pq = new PriorityQueue<>(new Comparator<int[]>(){
            @Override
            public int compare(int [] a1, int [] a2){
                return arrays[a1[0]][a1[1]] - arrays[a2[0]][a2[1]];
            }
        });

        for(int i=0; i<ArrCnt; i++){
            int [] arr = {i,0};
            pq.add(arr);
        }

        int []ans = new int[ArrCnt*size];
        int i=0;
        while(!pq.isEmpty()){
            int[]arr = pq.poll();
            // System.out.println(arr[0]+" "+arr[1]);
            ans[i] = arrays[arr[0]][arr[1]];
            if(arr[1] < size-1){
                arr[1]++;
                pq.add(arr);
            }
            i++;
        }

        for(int j=0; j<ans.length;j++){
            System.out.print(ans[j]+" ");
        }
    }
}
```

18. Frequent Elements in Array

Given an integer array nums and an integer k, Print the k most frequent elements. You have return the answer i
sorted order.

Input Format

The first line contains two Integer space-separated integers N (array size) and K.

The second line contains Element of Array

Constraints

$1 <= N <= 10^5$
$-10^4 <= arr[i] <= 10^4$
k is in the range [1, the number of unique elements in the array].
It is guaranteed that the answer is unique.

Output Format

Single Line consisting of K frequent elements in sorted order ,where each number is separated by space.

Sample Input

```
6  2
1 1 1 3 2 2
```

Sample Output

```
1 2
```

Explanation

None

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner psc=new Scanner(System.in);
        int n=psc.nextInt();
        int k=psc.nextInt();
        HashMap<Integer,Integer> map=new HashMap<>();
        for(int i=0;i<n;i++){
            int num=psc.nextInt();
            map.put(num,map.getOrDefault(num,0)+1);
```

```
        }
        PriorityQueue<Integer> pq=new PriorityQueue<>(new
Comparator<Integer>(){
            @Override
            public int compare(Integer o1,Integer o2){
                if(map.get(o1) == map.get(o2))
                    return o1-o2;
                else
                    return map.get(o1)-map.get(o2);
            }
        });


        List<Integer> keyset=new ArrayList<>(map.keySet());
        for(int i=0;i<k;i++)
            pq.add(keyset.get(i));

        for(int i=k;i<keyset.size();i++){
            if(map.get(pq.peek()) <= map.get(keyset.get(i))){
                pq.poll();
                pq.add(keyset.get(i));
            }
        }


        int[] ans=new int[k];
        for(int i=0;i<k;i++)
            ans[i]=pq.poll();

        Arrays.sort(ans);
        for(int ele:ans)
            System.out.print(ele+" ");


    }
}
```