Data Structure

Problems on Array

1. Given an array A of size N, construct a Sum Array S(of same size) such that S is equal to the sum of all the elements of A except A[i]. Your task is to complete the function SumArray(A, N) which accepts the array A and N(size of array).

Input:

The first line of input is the number of testcases T. Each of the test case contains two lines. The first line of which is an integer N(size of array). The second line contains N space separated integers.

Output:

For each test case print the sum array in new line.

User Task:

Since this is a functional problem you did not have to worry about the input. You just have to complete the function SumArray().

Constraint:

1 <= T <= 10

1 <= N <= 104

1 <= Ai <= 104

Example:

Input:

2

5

3 6 4 8 9

6

4 5 7 3 10 1

Output:

27 24 26 22 21

26 25 23 27 20 29

Explanation:

Testcase 1: For the sum array S, at i=0 we have 6+4+8+9. At i=1, 3+4+8+9. At i=2, we have 3+6+8+9. At i=3, we have 3+6+4+9. At i = 4, we have 3+6+4+8. So S is 27 24 26 22 21.

2. This is a functional problem. Your task is to return the product of array elements under a given modulo.

The modulo operation finds the remainder after division of one number by another. For example, K(mod(m))=K%m= remainder obtained when K is divided by m.

Input:

The first line of input contains T denoting the number of testcases. Then each of the T lines contains a single positive integer N denotes number of element in array. Next line contain 'N' integer elements of the array.

Output:

Return the product of array elements under a given modulo.

That is, return (Array[0]Array[1]Array[2]...*Array[n])%modulo.

Constraints:

1<=T<=200

1<=N<=10^5

1<=ar[i]<=10^5

Example:

Input:

1

4

1 2 3 4

Output:

24

3. Given an increasing sequence a [], we need to find the K-th missing contiguous element in the increasing sequence which is not present in the sequence. If no k-th missing element is there output -1.

Input:

The first line consists of an integer T i.e. the number of test cases. The first line of each test case consists of two integers N and K.Next line consists of N spaced integers.

Output:

For each test case, print the Kth missing number if present otherwise print -1.

Constraints:

1<=T<=100

1<=N,K,A[i]<=105

Examples:

Input

2

5 2

1 3 4 5 7

6 2

1 2 3 4 5 6

Output

6

-1

Explanation:

#TestCase 1:

K=2

We need to find the 2nd missing number in the array. Missing numbers are 2 and 6. So 2nd missing number is 6.

#Testcase 2:

K=2

We need to find the 2nd missing number in the array. As there is no missing number, hence the output is -1.

4. Write a program to input a list of n integers in an array and arrange them in a way similar to the to-and-fro movement of a Pendulum.

The minimum element out of the list of integers, must come in center position of array. If there are even elements, then minimum element should be moved to (n-1)/2 index (considering that indexes start from 0)

The next number (next to minimum) in the ascending order, goes to the right, the next to next number goes to the left of minimum number and it continues like a Pendulum.

Input:

The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case contains an integer n denoting the size of the array. Then next line contains N space separated integers forming the array.

Output:

Output the array in Pendulum Arrangement.

Constraints:

1<=T<=500

1<=N<=100

1<=a[i]<=1000

Example:

Input:

2

5

1 3 2 5 4

5

11 12 31 14 5

Output:

5 3 1 2 4

31 12 5 11 14

5. Given two arrays and a number x, find the pair whose sum is closest to x and the pair has an element from each array.

Input:

The first line consists of a single integer T, the number of test cases. For each test case, the first line contains 2 integers n & m denoting the size of two arrays. Next line contains n- space separated integers denoting the elements of array A and next lines contains m space separated integers denoting the elements of array B followed by an integer x denoting the number whose closest sum is to find.

Output:

For each test case, the output is 2 space separated integers whose sum is closest to x.

Constraints:

1<=T<=100

1<=n,m<=50

1<=A[i],B[i]<=500

Example:

Input:

2

4 4

1 4 5 7

10 20 30 40

32

4 4

1 4 5 7

10 20 30 40

50

Output:

1 30

7 40

Problems on Matrix

1. There is the Rectangular path for a Train to travel consisting of n and m rows and columns respectively. The train will start from one of grid cells and it will be given a command in the form of String s. consisting of characters 'L', 'R', 'U', 'D'. The train will follow the instructions of the command string, where 'L' corresponds moving to the left, 'R' towards the right, 'U' for moving up, and 'D' means down.

You have already selected the command string s, and are wondering if it is possible to place the train in one of the grid cells initially and have it always stay entirely within the grid upon execution of the command string s. Output "1" if there is a starting cell for which the train doesn't fall off the grid(track) on following command s, otherwise, output "0".

Input:

The first line of input will contain an integer T, the number of test cases.

Each test case will be on two lines.

The first line will have two space separated integers n,m.

The second line will have the command string s.

Output:

For each test case, output "1" (without quotes) or "0" (without quotes) in a new line.

Constraints:

1 <= T <= 1,00

1 <= n,m <= 10

1 <= |s| <= 10

Example:

Input:

2

1 1

R

2 3

LLRU

Output:

0

1

2. Given an incomplete Sudoku configuration in terms of a 9x9 2-D square matrix (mat[][]) the task to check if the configuration has a solution or not.

Input:

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains 9*9 space separated values of the matrix mat[][] representing an incomplete Sudoku state where a 0 represents empty block.

Output:

For each test case in a new line print 1 if the sudoku configuration is valid else print 0.

Constraints:

1<=T<=10

0<=mat[]<=9

Example:

Input:

2

3 0 6 5 0 8 4 0 0 5 2 0 0 0 0 0 0 0 0 8 7 0 0 0 0 3 1 0 0 3 0 1 0 0 8 0 9 0 0 8 6 3 0 0 5 0 5 0 0 9 0 6 0 0 1 3 0 0 0 0 2 5 0 0 0 0 0 0 0 7 4 0 0 5 2 0 6 3 0 0

3 6 7 5 3 5 6 2 9 1 2 7 0 9 3 6 0 6 2 6 1 8 7 9 2 0 2 3 7 5 9 2 2 8 9 7 3 6 1 2 9 3 1 9 4 7 8 4 5 0 3 6 1 0 6 3 2 0 6 1 5 5 4 7 6 5 6 9 3 7 4 5 2 5 4 7 4 4 3 0 7

Output:

1

0

3. Given a N X N matrix (M) filled with 1 , 0 , 2 , 3 . Your task is to find whether there is a path possible from source to destination, while traversing through blank cells only. You can traverse up, down, right and left.

A value of cell 1 means Source.

A value of cell 2 means Destination.

A value of cell 3 means Blank cell.

A value of cell 0 means Blank Wall.

Note : there is only single source and single destination.

Examples:

Input : M[3][3] = {{ 0 , 3 , 2 },

{ 3 , 3 , 0 },

{ 1 , 3 , 0 }};

Output : Yes

Input : M[4][4] = {{ 0 , 3 , 1 , 0 },

{ 3 , 0 , 3 , 3 },

{ 2 , 3 , 0 , 3 },

{ 0 , 3 , 3 , 3 }};

Output : Yes

Input:

The first line of input is an integer T denoting the no of test cases. Then T test cases follow. Each test case consists of 2 lines . The first line of each test case contains an integer N denoting the size of the square matrix . Then in the next line are N*N space separated values of the matrix (M) .

Output:

For each test case in a new line print 1 if the path exist from source to destination else print 0.

Constraints:

1<=T<=20

1<=N<=20

Example:

Input:

2

4

3 0 0 0 0 3 3 0 0 1 0 3 0 2 3 3

3

0 3 2 3 0 0 1 0 0

Output:

1

0

4. Given an square matrix, turn it by 90 degrees in anti-clockwise direction without using any extra space.

Input:

The first line of input contains a single integer T denoting the number of test cases. Then T test cases follow. Each test case consist of two lines. The first line of each test case consists of an integer N, where N is the size of the square matrix. The second line of each test case contains NxN space separated values of the matrix M.

Output:

Corresponding to each test case, in a new line, print the rotated array.

Constraints:

1 ≤ T ≤ 50

1 ≤ N ≤ 50

Example:

Input

1

3

1 2 3 4 5 6 7 8 9

Output

3 6 9 2 5 8 1 4 7

5. Given a 2D matrix of size M*N. Traverse and print the matrix in spiral form.

Input:

The first line of the input contains a single integer T, denoting the number of test cases. Then T test cases follow. Each testcase has 2 lines. First line contains M and N respectively separated by a space. Second line contains M*N values separated by spaces.

Output:

Elements when travelled in Spiral form, will be displayed in a single line.

Constraints:

1 <=T<= 100

2 <= M, N <= 10

0 <= Ai <= 100

Example:

Input:

1

4 4

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Output:

1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Problems on Linked list

1. Given a Linked List of integers, write a function to modify the linked list such that all even numbers appear before all the odd numbers in the modified linked list. Also, keep the order of even and odd numbers same.

Input:

The first line of input contains an integer T denoting the number of test cases.

The first line of each test case is N,N is the number of elements in Linked List.

The second line of each test case contains N input, elements in Linked List.

Output:

Print the all even numbers then odd numbers in the modified Linked List.

Constraints:

1 ≤ T ≤ 100

1 ≤ N ≤ 100

1 ≤ size of elements ≤ 1000

Example:

Input

3

7

17 15 8 9 2 4 6

4

1 3 5 7

7

8 12 10 5 4 1 6

Output

8 2 4 6 17 15 9

1 3 5 7

8 12 10 4 6 5 1

2. Write a Count() function that counts the number of times a given int occurs in a list. The code for this has the classic list traversal structure as demonstrated in Length().

3. Write a Pop() function that is the inverse of Push(). Pop() takes a non-empty list, deletes the head node, and returns the head node's data. If all you ever used were Push() and Pop(), then our linked list would really look like a stack. However, we provide more general functions like GetNth() which what make our linked list more than just a stack. Pop() should assert() fail if there is not a node to pop.

4. Write a SortedInsert() function which given a list that is sorted in increasing order, and a single node, inserts the node into the correct sorted position in the list. While Push() allocates a new node to add to the list, SortedInsert() takes an existing node, and just rearranges pointers to insert it into the list.

5. Write a RemoveDuplicates() function which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.

Problems on Stack

1. Delete the middle element of the stack. Given a stack with push(), pop(), empty() operations, delete middle of it without using any additional data structure.

2 .Print Bracket Number. Given an expression exp of length n consisting of some brackets. The task is to print the bracket numbers when the expression is being parsed.

Examples :

Input : (a+(b*c))+(d/e)

Output : 1 2 2 1 3 3

The highlighted brackets in the given expression

(a+(b*c))+(d/e) has been assigned the numbers as:

1 2 2 1 3 3.

Input : ((()))(()))

Output : 1 2 3 3 2 4 5 5 4 1

3. Given array A[] of integers, the task is to complete the function findMaxDiff which finds the maximum absolute difference between nearest left and right smaller element of every element in array.If the element is the leftmost element, nearest smaller element on left side is considered as 0. Similarly if the element is the rightmost elements, smaller element on right side is considered as 0.

4. Given an array A of size N having distinct elements, the task is to find the next greater element for each element of the array in order of their appearance in the array. If no such element exists, output -1.

5. Given an array of integers, find the nearest smaller number for every element such that the smaller element is on left side.If no small element present on the left print -1.

Problems on Binary Tree

1. Given a Binary Tree you need to find maximum value which you can get by subtracting value of node B from value of node A, where A and B are two nodes of the binary tree and A is an ancestor of B . You are required to complete the function maxDiff . You should not read any input from stdin/console. There are multiple test cases. For each test case, this method will be called individually.

Input:

The task is to complete the function maxDiff which takes 1 argument, root of the Tree . The struct node has a data part which stores the data, pointer to left child and pointer to right child.

There are multiple test cases. For each test case, this method will be called individually.

Output:

The function should return an integer denoting the maximum difference.

Constraints:

1 <=T<= 30

1 <=Number of nodes<= 100

1 <=Data of a node<= 1000

Example

Input

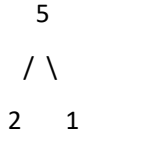
1

2

5 2 L 5 1 R

Output

4



In above example there is one test case which represents a tree with 3 nodes and 2 edges where root is 5, left child of 5 is 2 and right child of 5 is 1 hence the max difference we can get is from 5 and 1 ie 4 .

2. Given a Complete Binary tree, print the level order traversal in sorted order.

Input:

The first line of the input contains integer T denoting the number of test cases. For each test case, the first line takes an integer n denoting the size of array i.e number of nodes followed by n-space separated integers denoting the nodes of the tree in level order fashion.

Output:

For each test case, the output is the level order sorted tree.

Note: For every level, we only print distinct elements.

Constraints:

1<=T<=100

1<=n<=10^5

Example:

Input:

2

7

7 6 5 4 3 2 1

6

5 6 4 9 2 1

Output:

7

5 6

1 2 3 4

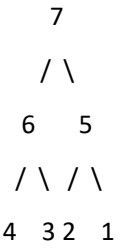
5

4 6

1 2 9

Explanation:

Tree looks like this



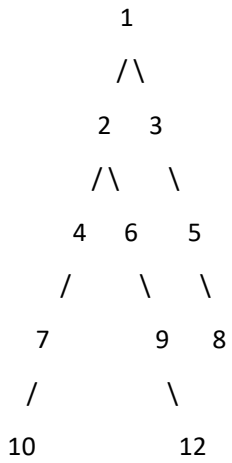
Sorted order:

7

5 6

1 2 3 4

3. Given a binary tree, your task is to complete the function findRightSibling(), that should return the right sibling to a given node if it doesn't exist return null.



Input : Given above tree with parent pointer and node 10

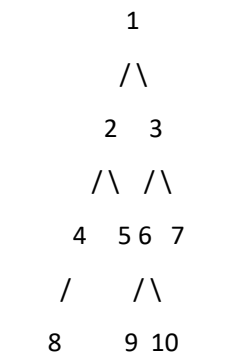
Output : 12

4. Given a binary tree of size N and two nodes. Your task is to complete the function NumberOFTurn() that should return the count of the number of turns needs to reach from one node to another node of the Binary tree.

Example:

Input: Below Binary Tree and two nodes

5 & 6



Output: Number of Turns needed to reach from 5 to 6: 3

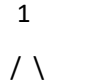
Input: For above tree if two nodes are 1 & 4

Output: Straight line : 0 turn

5. Given a binary tree of size N+1, your task is to complete the function tiltTree(), that return the tilt of the whole tree. The tilt of a tree node is defined as the absolute difference between the sum of all left subtree node values and the sum of all right subtree node values. Null nodes are assigned tilt to be zero. Therefore, tilt of the whole tree is defined as the sum of all nodes' tilt.

Examples:

Input :



2 3

Output : 1

Explanation:

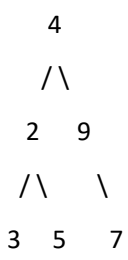
Tilt of node 2 : 0

Tilt of node 3 : 0

Tilt of node 1 : |2-3| = 1

Tilt of binary tree : 0 + 0 + 1 = 1

Input :



Output : 15

Explanation:

Tilt of node 3 : 0

Tilt of node 5 : 0

Tilt of node 7 : 0

Tilt of node 2 : |3-5| = 2

Tilt of node 9 : |0-7| = 7

Tilt of node 4 : |(3+5+2)-(9+7)| = 6

Tilt of binary tree : 0 + 0 + 0 + 2 + 7 + 6 = 15

Problems on Binary Search Tree

1. Given a binary tree, Your task is to complete the function largestBst that returns the size of the largest subtree which is also a Binary Search Tree (BST). If the complete Binary Tree is BST, then return the size of whole tree.

Input:

The task is to complete the method which takes one argument, root of Binary Tree. The Node has a data part which stores the data, pointer to left child and pointer to right child.

There are multiple test cases. For each test case, this method will be called individually.

Output:

The function should return the size of the max subtree which is also the BST.

Constraints:

1 <=T<= 30

1 <=Number of nodes<= 100

1 <=Data of a node<= 1000

2. Given a Binary Search Tree (BST) and a node no 'x' , your task is to delete the node 'x' from the BST . You are required to complete the function deleteNode. You should not read any input from stdin/console. There are multiple test cases. For each test case, this method will be called individually.

Input (only to be used for Expected Output):

The first line of the input contains an integer 'T' denoting the number of test cases. Then 'T' test cases follow. Each test case consists of three lines. Description of test cases is as follows:

The First line of each test case contains an integer 'N' which denotes the no of nodes in the BST. .

The Second line of each test case contains 'N' space separated values of the nodes in the BST.

The Third line of each test case contains an integer 'x' the value of the node to be deleted from the BST.

Output:

You are required to complete the function deleteNode which takes two arguments. The first being the root of the tree, and an integer 'x' denoting the node to be deleted from the BST . The function returns a pointer to the root of the modified BST .

Constraints:

1 <= T <= 50

1 <= N <= 50

3. Given two BST, Your task is to complete the function merge which prints the elements of both BSTs in sorted form.

Input:

The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains three lines. The first line of each test case contain's an integer N and M denoting the size of the two BST's. Then In the next two line are space separated values of the two BST's.

Output:

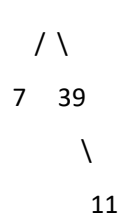
The function should print the elements of both BST's in sorted form.

Constraints:

1<=T<=100

1<=N,M<=100

4. Given a binary tree, return true if it is BST, else false. For example, the following tree is not BST, because 11 is in left subtree of 10.



Input:

The task is to complete the method which takes one argument, root of Binary Tree. The struct Node has a data part which stores the data, pointer to left child and pointer to right child.

There are multiple test cases. For each test case, this method will be called individually.

Output:

The function should return 1 if BST else return 0.

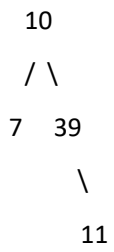
Constraints:

1 <=T<= 30

1 <= Number of nodes<= 100

1 <= Data of a node<= 1000

5. Given a binary tree, return true if it is BST, else false. For example, the following tree is not BST, because 11 is in left subtree of 10.



Input:

The task is to complete the method which takes one argument, root of Binary Tree. The struct Node has a data part which stores the data, pointer to left child and pointer to right child.

There are multiple test cases. For each test case, this method will be called individually.

Output:

The function should return 1 if BST else return 0.

Constraints:

1 <=T<= 30

1 <= Number of nodes<= 100

1 <= Data of a node<= 1000

Problems on Graph

Given a graph, check whether it is Biconnected or not.

Input:

First line consists of T test cases. First line of every test case consists of 2 integers N, E,denoting the number of vertices and number of edges respectively. Second line of every test case consists of pair of E spaced integers, denoting the edges connecting each other.

Output:

Single line output, print 1 if graph is biconnected else 0.

Constraints:

1<=T<=100

1<=N,E<=100

Example:

Input:

3

2 1

0 1

5 6

1 0 0 2 2 1 0 3 3 4 2 4

3 2

0 1 1 2

Output:

1

1

0

2. Mayank once came to New Delhi for an International Mathematics Conference. His mathematics skills were a great topic of discussion in those days. So a mathematician Harsh "Challenged" Mayank for a competition and Mayank accepted it.

Rules: They will ask 3 questions each and the one who will give all the answers right will be the winner.

Mayank answered all the questions very easily. Now its time for Harsh to answer the questions but Harsh was confused by the very first question now you have to help so let's read the question.

Question: Mayank asked that you are in a beautiful Castle which is huge and contains 'N' number of rooms and each room contains 'M' path to 'M' other rooms. There may be only one path which connects two room i.e either one path or no path. Mayank asked that you have to find whether you can reach back to the room from which you started after travelling all the paths but note that once you travel a path, then it will be closed and make sure all the rooms are travelled atleast once. In general all paths are to be travelled exactly once and each room must be travelled atleast once. Note: If any room is not connected print False and if there is only one room then print True.

Input:

The first line will contain an integer T(number of test cases). Each test case will contain an Integer 'N' i.e.number of rooms and next 'N' lines will contain 'N' integers each stating whether there is a path between room ni & nj.

Note: If no path exists then the value is -1, if the room is same then the value will be 0 and if a path exists the value will be 1. Path between room ni & nj is both ways.

Output:

For each test case, print "True" if you can reach back else print "False".

Constraints:

1<=T<=100

1<=N<=10000

3. Given N * M string array of O's and X's

Return the number of 'X' total shapes. 'X' shape consists of one or more adjacent X's (diagonals not included).

Example (1):

OOOXOOO

OXXXXXO

OXOOOXO

answer is 1 , shapes are :

(i) X

Example (2):

XXX

OOO

XXX

answer is 2, shapes are

(i) XXX

(ii) XXX

4. Given a connected acyclic graph with N nodes and N-1 edges, find out the pair of nodes that are at even distance from each other.

Input:

The first line of input contains an integer T denoting the number of test cases.

First line of each test case contains a single integer N denoting the number of nodes in graph.

Second line of each test case contains N-1 pair of nodes xi , yi denoting that there is an edges between them.

Output:

For each test case output a single integer denoting the pair of nodes which are at even distance.

Constraints:

1<=T<=10

1<=N<=10000

1<=xi,yi<=N

Example

Input

1

3

1 2 2 3

Output

1

5. Given a directed graph your task is to complete the method isCycle to detect if there is a cycle in the graph or not. You should not read any input from stdin/console.

There are multiple test cases. For each test case, this method will be called individually.

Input (only to be used for Expected Output):

The first line of the input contains an integer 'T' denoting the number of test cases. Then 'T' test cases follow. Each test case consists of two lines.

Description of test cases is as follows:

The First line of each test case contains two integers 'N' and 'M' which denotes the no of vertices and no of edges respectively.

The Second line of each test case contains 'M' space separated pairs u and v denoting that there is an unidirected edge from u to v.

Output:

The method should return true if there is a cycle else it should return false.

Constraints:

1 <=T<= 100

1<=N,M<=100

0 <=u,v<= N-1

Example:

Input:

2

2 2

0 1 0 0

4 3

0 1 1 2 2 3

Output:

1

0