JTG Problem List

Q. Given a linked list, how do you ensure that it is in decreasing order?

Ans.

Check if a linked list is sorted in decreasing order by comparing adjacent nodes.

- Traverse the linked list from head to tail.
- For each node, compare its value with the next node's value.
- If any node's value is less than the next node's value, the list is not in decreasing order.
- Example: For list 5 -> 3 -> 2, it is in decreasing order.
- Example: For list 5 -> 6 -> 2, it is not in decreasing order.

Q. Divide Chocolates -

Ninja bought chocolate consisting of several chunks, and the sweetness of each chunk is represented in an array `ARR`. He wants to divide the chocolate into `K + 1` parts (cuts), where `K` is the number of his friends. Ninja desires to take only one part with the minimum total sweetness.

Your task is to maximize the total sweetness of the part that Ninja will get based on the condition stated above.

Input:

The first line contains an integer 'T', representing the number of test cases.
For each test case, the first line consists of two space-separated integers, 'N' and 'K', indicating the number of chunks and the number of friends.
The second line for each test case contains 'N' space-separated integers representing the sweetness of each chunk in 'ARR'.

Output:

For each test case, provide the maximum sweetness that Ninja can obtain under the given condition.

Example:

Input:

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

Output:

```
15
9
```

Explanation:

In the first example, by making 2 cuts, Ninja can take a part with the maximum minimum sweetness of 15.
In the second example, making 3 cuts allows Ninja to take a part with the maximum minimum sweetness of 9.

Q. Merge Two Binary Trees

You are given the roots of two binary trees, `root1` and `root2`. Merge these two trees into a new binary tree. If two nodes overlap, sum the node values as the new node value. If they don't overlap, retain the non-null node in the merged tree. Return the root of this merged tree.

Input:

```
The first line contains an integer T denoting the number of test cases.
In the first of the 2*T subsequent lines of each test case, given as space-separated integers are the nodes of the first binary tree in preorder format, with -1 representing a null pointer.
In the second line of each test case, given as space-separated integers are the nodes of the second binary tree in preorder format, with -1 representing a null pointer.
```

Output:

```
For each test case, output the merged binary tree nodes in preorder format, printed in a separate line for each test case.
```

Example:

Input:
```
‘ROOT1’ = [1, 2, -1, -1, 3, -1, -1]  ‘ROOT2’ = [3, -1, -1]
```
Output:
```
[4, 2, 3]
```
Explanation:

The trees are merged to form a new tree represented as a pre-order traversal: `4 2 3`.

Constraints:

- `1 <= T <= 5`
- `1 <= N <= 10^3`
- `-10^3 <= DATA <= 10^3`

Q. Divide Chocolates

Ninja bought chocolate consisting of several chunks, and the sweetness of each chunk is represented in an array `ARR`. He wants to divide the chocolate into `K + 1` parts (cuts), where `K` is the number of his friends. Ninja desires to take only one part with the minimum total sweetness.

Your task is to maximize the total sweetness of the part that Ninja will get based on the condition stated above.

Input:

```
The first line contains an integer ‘T’, representing the number of test cases.
 For each test case, the first line consists of two space-separated integers, ‘N’ and ‘K’, indicating the number of
```

chunks and the number of friends.
The second line for each test case contains ‘N’ space-separated integers representing the sweetness of each chunk in ‘ARR’.

Output:

For each test case, provide the maximum sweetness that Ninja can obtain under the given condition.

Example:

Input:

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

Output:

```
15
9
```

Explanation:

In the first example, by making 2 cuts, Ninja can take a part with the maximum minimum sweetness of 15.
In the second example, making 3 cuts allows Ninja to take a part with the maximum minimum sweetness of 9.

Q. K Largest Elements

You are given an unsorted array containing 'N' integers. Your task is to find the 'K' largest elements from this array and return them in non-decreasing order.

Input:

The first line contains an integer 'T' which represents the number of test cases.
Each test case begins with a line containing two space-separated positive integers 'N' and 'K', where 'N' is the number of elements in the array and 'K' is the number of largest elements you need to return.
The following line contains 'N' space-separated integers representing the elements of the array.

Output:

For each test case, output 'K' largest elements from the array in non-decreasing order on a new line.

Example:

Input:

```
2
5 3
1 2 3 4 5
7 2
3 -2 7 1 5 0 6
```

Output:

```
3 4 5
6 7
```

Constraints:

- $1 \leq T \leq 100$
- $1 \leq N \leq 10^4$
- $1 \leq K \leq N$
- $-10^9 \leq ARR[i] \leq 10^9$

Q. 0/1 Knapsack Problem

A thief is planning to rob a store and can carry a maximum weight of 'W' in his knapsack. The store contains 'N' items where the ith item has a weight of 'wi' and a value of 'vi'. Given the maximum weight capacity of the knapsack, determine the maximum total value of items that the thief can steal.

Input:

The first line contains an integer 'T' representing the number of test cases.
Each test case includes:
1. An integer 'N' denoting the number of items.
2. A line with 'N' space-separated integers representing the weights of the items.
3. A line with 'N' space-separated integers representing the values of the items.
4. An integer 'W' denoting the maximum weight capacity of the knapsack.

Output:

For each test case, output a single integer denoting the maximum value that can be stolen. Each result should be printed on a new line.

Example:

Input:
2
3
10 20 30
60 100 120
50
3
1 2 3
10 20 30
5
Output:
220
50

Constraints:

- 1 <= T <= 10
- 1 <= N <= 10^2
- 1 <= wi <= 50
- 1 <= vi <= 10^2
- 1 <= W <= 10^3

Can the problem be solved using a space complexity of not more than O(W)?

Q. Size of Largest BST in a Binary Tree

Given a binary tree with 'N' nodes, determine the size of the largest subtree that is also a Binary Search Tree (BST).

Explanation:

A Binary Search Tree (BST) is defined as follows:

- **The left subtree of a node contains only nodes with values less than the node's value.**
- **The right subtree of a node contains only nodes with values greater than the node's value.**
- **Both the left and right subtrees must also be binary search trees.**

Input:

The first line contains an integer 'T', representing the number of test cases. Each test case follows: A single line with space-separated values representing the binary tree in level order. Use -1 for null nodes.

Output:

For each test case, return an integer that indicates the size of the largest BST subtree within the binary tree.

Example:

```
Input:
1
1 2 3 4 -1 5 6 -1 7 -1 -1 -1 -1 -1 -1

Output:
3
```

Explanation: In the given tree, the largest subtree that is also a BST has 3 nodes.

Constraints:

- **1 <= T <= 100**
- **1 <= N <= 5000**
- **1 <= data <= 10^5 and data != -1**

Note: You do not need to print anything; simply implement the function to return the result.

Q. Longest Common Subsequence

Given two strings `STR1` and `STR2`, determine the length of their longest common subsequence.

A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements. A common subsequence is one that appears in both strings.

Input:

The first line contains an integer T denoting the number of test cases. Each test case consists of two space-separated strings STR1 and STR2.

Output:

Print the length of the longest common subsequence for each test case on a separate line.

Example:

Input:
STR1 = "abcde", STR2 = "ace"
Output:
3
Explanation:

The longest common subsequence is "ace", which has a length of 3.

Constraints:

- 1 <= T <= 50
- 1 <= |STR1| <= 10^2
- 1 <= |STR2| <= 10^2
- **Time Limit: 1 sec**

Q. Longest Palindromic Substring

You are provided with a string STR of length N. The task is to find the longest palindromic substring within STR. If there are several palindromic substrings of the maximum length, return the one that starts with the smallest index.

Input:

The first line of the input contains a single integer **T**, the number of test cases.
Each of the following **T** lines contains a single string STR.

Output:

For each test case, output the longest palindromic substring. If multiple substrings qualify, return the one that appears first.

Example:

Input:
2
ababc
cbbd

Output:

```
aba
bb
```

Q. Rotate Linked List

Given a linked list consisting of 'N' nodes and an integer 'K', your task is to rotate the linked list by 'K' positions in a clockwise direction.

Example:

Input:

```
Linked List: 1 2 3 4 -1
K: 2
```

Output:

```
3 4 1 2
```

Q. Balanced Binary Trees -

You are provided with an integer 'H'. Your task is to determine and print the maximum number of balanced binary trees possible with height 'H'.

A balanced binary tree is defined such that for every node, the difference between the heights of the left and right subtrees is no more than 1.

Your answer should be printed modulo 1e9+7.

Example:

Input:

```
H = 2
```

Output:

```
3
```

Q. Wave Array Sorting Problem

Your goal is to sort a given unsorted array `ARR` such that it forms a wave array.

Explanation:

After sorting, the array should satisfy the wave pattern conditions:

- `ARR[0] >= ARR[1]`
- `ARR[1] <= ARR[2]`
- `ARR[2] >= ARR[3]`
- `ARR[3] <= ARR[4]`
- **And so on...**

Multiple solutions may exist, but you need to return only one valid wave array.

Example:

Input:

```
ARR = [4, 3, 5, 2, 3, 1, 2]
```

Output:

```
[4, 3, 5, 2, 3, 1, 2]
```

Q. Validate Binary Tree Nodes Problem

You are provided with 'N' binary tree nodes labeled from 0 to N-1, where node *i* has two potential children, recorded in the `LEFT_CHILD[i]` and `RIGHT_CHILD[i]` arrays. Determine whether all the nodes given constitute precisely one valid binary tree. If node *i* lacks a left child, then `LEFT_CHILD[i]` will be -1; the same applies to the right child.

Example:

Input:

```
n = 4
LEFT_CHILD = [1, -1, 3, -1]
RIGHT_CHILD = [2, -1, -1, -1]
```

Output:

```
True
```

Q. Find Nodes at Distance K in a Binary Tree

Your task is to find all nodes that are exactly a distance K from a given node in an arbitrary binary tree. The distance is defined as the number of edges between nodes.

Input:

The first line contains an integer T, the number of test cases. For each test case: 1. The tree's nodes are given in level order (use -1 for a NULL node). 2. The value of the target node is specified. 3. An integer K represents the required distance from the target node.

Output:

For each test case, output a list of node values that are exactly K edges away from the given target node. If no such nodes exist, return an empty list.

Example:

```
Input:
1
3 5 1 6 2 0 8 -1 -1 7 4 -1 -1 -1 -1 -1 -1 -1 -1
target = 5
K = 2

Output:
[7, 4, 0, 8]
```

Constraints:

- $1 \leq T \leq 100$
- $1 \leq N \leq 3000$
- $0 \leq K \leq 3000$
- $0 \leq nodeValue \leq 3000$

Q. Maximum Sum BST -

You are presented with a Binary Tree 'root', which may not necessarily be a Binary Search Tree (BST). Your objective is to identify the maximum sum of node values in any subtree that qualifies as a Binary Search Tree (BST).

Explanation:

A Binary Search Tree is defined as follows:

1) If a left subtree exists for any node, it should comprise only nodes with values less than that node's value.
2) If a right subtree exists for any node, it should comprise only nodes with values greater than that node's value.
3) Both the left and right subtrees must themselves be Binary Search Trees.

Input:

The first line contains a single integer 'T' denoting the number of test cases, followed by each test case.
The first line of each test case contains the tree's elements in a level-order format, separated by spaces. If a node lacks a left or right child, represent it with -1.

Output:

For each test case, return the maximum sum achievable. Output each result on a new line.

Example:

Sample Input:

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

Sample Output:

```
28
3
```

Q. Pair Sum in BST -

Given a Binary Search Tree (BST) and a target value `K`, determine if there exist two unique elements in the BST such that their sum equals `K`.

Input:

An integer T, the number of test cases. For each test case, the first line contains the BST elements in level order format. Values of nodes are separated by a space. Use -1 for null nodes. The second line contains an integer K, the target sum.

Output:

For each test case, output 'true' if any pair of unique elements sums to K, otherwise output 'false'.

Example:

Input:
20 10 35 5 15 30 42 -1 13 -1 -1 -1 -1 -1 -1 -1
33
Output:
true

Constraints:

- $1 \leq T \leq 5$
- $1 \leq N \leq 10^3$
- $0 \leq DATA \leq 10^9$
- $1 \leq K \leq 10^9$

Q. Last Stone Weight Problem

Given a collection of stones, each having a positive integer weight, perform the following operation: On each turn, select the two heaviest stones and smash them together. Assume the stones have weights 'x' and 'y' where 'x' <= 'y'. The outcomes of smashes are:

1. **If 'x' == 'y', both stones are destroyed.**
2. **If 'x' != 'y', the stone with weight 'x' is destroyed, and the stone with weight 'y' is updated to 'y - x'.**

Continue this process until at most one stone remains. Return the weight of the last stone, or 0 if no stones are left.

Input:

First line: Integer 'N', the number of stones.
Second line: 'N' space-separated integers representing the weights of the stones.

Output:

Print the weight of the final stone, or 0 if no stones remain.

Example:

Input:
N = 6
Weights = [2, 7, 4, 1, 8, 1]

Output:
1

Constraints:

- 1 <= N <= 10^5
- 1 <= W[i] <= 10^6
- **Time Limit: 1 sec**

Q. Given a Binary Search Tree (BST), find if there exists a pair of nodes whose values sum up to a given target. The time complexity required for this solution is O(n), and the space complexity should be less than O(n).
Ans.
Check for a pair of nodes in a BST that sum to a target using O(n) time and O(1) space.

- Use an in-order traversal to get sorted values from the BST.
- Utilize two pointers: one at the start and one at the end of the sorted array.
- If the sum of the values at the two pointers equals the target, return true.
- If the sum is less than the target, move the left pointer right; if greater, move the right pointer left.
- Example: For BST with values [2, 4, 5, 7, 10] and target 9, return true (4 + 5).

Q. Convert the Binary Search Tree (BST) to a Greater Tree.
Ans.
Convert a BST to a Greater Tree by replacing each node's value with the sum of all greater values.

- Traverse the tree in reverse in-order (right, root, left) to accumulate values.
- Start with a variable to keep track of the cumulative sum.
- For each node, update its value to the cumulative sum and add its original value to the sum.
- Example: For BST with values [4, 1, 6, 0, 2, 5, 7], the Greater Tree will have values [22, 27, 21, 28, 26, 26, 15].

Q. Given a linked list, how do you ensure that it is in decreasing order?
Ans.
Check if a linked list is sorted in decreasing order by comparing adjacent nodes.

- Traverse the linked list from head to tail.
- For each node, compare its value with the next node's value.
- If any node's value is less than the next node's value, the list is not in decreasing order.
- Example: For list 5 -> 3 -> 2, it is in decreasing order.
- Example: For list 5 -> 6 -> 2, it is not in decreasing order.

Q. Sort Linked List Based on Actual Values

You are given a Singly Linked List of integers that is initially sorted according to the absolute values of its elements. Your task is to sort this Linked List based on the actual integer values.

The absolute value of a number `x`, denoted as `|x|`, is the non-negative value of `x` without regard to its sign.

Input:
The first line consists of a single integer ‘T’, denoting the number of test cases. Each test case includes a single line containing the elements of the singly linked list, separated by spaces, and terminated by -1. The value -1 is not part of the list data and serves as a terminator.

Output:

For each test case, provide the sorted linked list where the elements are separated by a space, ending with -1.

Example:

```
Input:
1 -> -2 -> 3 -1
Output:
-2 -> 1 -> 3 -1
```

Q. Find Unique Element in Array

You have been provided an integer array/list `ARR` of size `N`. Here, `N` is equivalent to `2M + 1` where each test case has `M` numbers appearing twice and one number appearing exactly once.

Your task is to identify and return this unique number that appears only once.

Example:

Input:

```
t = 2
N = 5
ARR = [1, 2, 3, 2, 1]
N = 7
ARR = [4, 5, 6, 7, 7, 6, 5]
```

Output:

```
3
4
```

Explanation:

In the first test case, `3` is the only number that is not repeated. In the second test case, `4` is the number which appears only once.

Q. Zig-Zag Array Rearrangement

You are provided with an array of distinct elements, and your task is to rearrange the array elements in a zig-zag manner. Specifically, for every odd index `i`, the element `ARR[i]` should be greater than both `ARR[i-1]` and `ARR[i+1]`.

Input:

```
The first line contains an integer 'T' which represents the number of test cases.
Each test case starts with an integer 'N' representing the size of the array.
The next line contains 'N' space-separated integers denoting the array elements.
```

Output:

```
A zig-zag array for each test case. The output function will validate if the returned array is in zig-zag fashion.
```

Example:

Input:

```
N = 4, ARR = [4, 3, 2, 1]
```

Output:

```
A possible zig-zag array: [3, 4, 1, 2]
```

Q. Inorder Successor in a Binary Tree

Given an arbitrary binary tree and a specific node within that tree, determine the inorder successor of this node.

Explanation:

The inorder successor of a node in a binary tree is the node that is visited immediately after the given node during an inorder traversal of the tree. If the node is the last to be visited in the traversal, then its inorder successor is NULL.

In an inorder traversal, the left subtree is visited first, followed by the node itself, and then the right subtree.

Input:

```
The first line contains a single integer 'T' representing the number of test cases. Each test case includes:
1. Values of the nodes of the tree in level order format (with -1 indicating a NULL node).
2. Value of the node for which the inorder successor is to be determined.
```

Output:

```
For each test case, print the value of the inorder successor node, or 'NULL' if no successor exists. Print each result on a new line.
```

Example:

Input:

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

Output:

```
5
NULL
```

Q. Huffman Coding Challenge

Given an array `ARR` of integers containing 'N' elements where each element denotes the frequency of a character in a message composed of 'N' alphabets of an alien language, your task is to find the Huffman codes for these alphabets.

Ensure the following for Huffman codes:

```
1) All codes should be binary strings.
2) Each code can distinctly identify its corresponding character.
3) Minimize the total number of bits used for the message.
```

Example:

Input:

```
ARR = [1, 4, 2]
```

Output:

```
Possible Huffman Codes: ['10', '0', '11']
```

Q. Ways To Make Coin Change

Given an infinite supply of coins of varying denominations, determine the total number of ways to make change for a specified value using these coins. If it's not possible to make the change, the result should be 0.

Input:

```
The first line contains an integer N, denoting the total number of denominations.
The second line contains N space-separated integers, each representing a denomination value.
The third line contains the integer V, representing the value to make change for.
```

Output:

```
An integer denoting the total number of ways to make change for V.
```

Example:

Input:

```
N = 3
D = [1, 2, 3]
V = 4
```

Output:

```
4
```

Explanation:

There are four ways to make change for 4 using the denominations [1, 2, 3]:
1. 1+1+1+1
2. 1+1+2
3. 1+3
4. 2+2

Q. Remove BST Keys Outside Given Range

Given a Binary Search Tree (BST) and a specified range [min, max], your task is to remove all keys from the BST that fall outside this range. The BST should remain valid after modifications.

Input:

The first line contains an integer 'T', representing the number of test cases. Each test case involves: 1. An initial line describing the BST elements in level order format (use -1 for missing children). 2. A subsequent line containing two integers 'min' and 'max'.

Output:

For each test case, output a single line of 'N' space-separated integers, representing the inorder traversal of the adjusted BST.

Example:

If the BST elements are [4, 2, 7, 1, 3, -1, -1] and the range is [1, 3], the output should be [1, 2, 3] after removing keys outside the specified range.

Constraints:

- **1 <= T <= 5**
- **1 <= N <= 10^5**
- **-10^3 <= nodeVal <= 10^3**

Q. Segregate Odd-Even -

In a wedding ceremony at NinjaLand, attendees are blindfolded. People from the bride's side hold odd numbers, while people from the groom's side hold even numbers. For the game to start quickly, all the bride's side people should come first, followed by the groom's side people, maintaining their original order.

Explanation:

The attendees are represented as a singly linked list, and you need to rearrange the list such that all odd-numbered attendees appear before the even-numbered ones, preserving the order of appearance.

Input:

The first line contains an integer T, the number of test cases.
Each test case consists of a single line of integers representing the linked list, separated by spaces, and terminated by -1.
-1 is not a part of the list.

Output:

For each test case, output the rearranged list with odd numbers first, followed by even numbers, separated by spaces and terminated by -1.
Each test case should be printed on a new line.

Example:

Input:

```
1
1 4 3 -1
```

Output:

```
1 3 4 -1
```

Q. Check If Linked List Is Palindrome

Given a singly linked list of integers, determine if the linked list is a palindrome.

Explanation:

A linked list is considered a palindrome if it reads the same forwards and backwards.

Input:

The first line contains an integer T, representing the number of test cases. For each test case, a single line contains the elements of the linked list separated by spaces, ending with -1. The -1 is not part of the list, and only acts as a terminator.

Output:

For each test case, output "True" if the linked list is a palindrome, otherwise output "False".

Example:

Input:

```
3
1 2 1 -1
3 4 4 3 -1
1 -1
```

Output:

```
True
True
True
```

Q. Balanced Parentheses Combinations

Given an integer `N` representing the number of pairs of parentheses, find all the possible combinations of balanced parentheses using the given number of pairs.

Explanation:

Conditions for valid parentheses:

- **All open brackets must be closed by the closing brackets.**
- **Open brackets must be closed in the correct order.**

Input:

The first line of input contains an integer ‘T’, which denotes the number of test cases. Then each test case follows. Each line of the test case contains an integer ‘N’ denoting the pair of parentheses.

Output:

For each test case, print all the combinations of balanced parentheses separated by a single space. The output of each test case will be printed on a separate line.

Example:

For N = 2, the valid combinations are:

```
Input:
T = 1
N = 2
Output:
()() (())
```

Q. Intersection of Linked List Problem

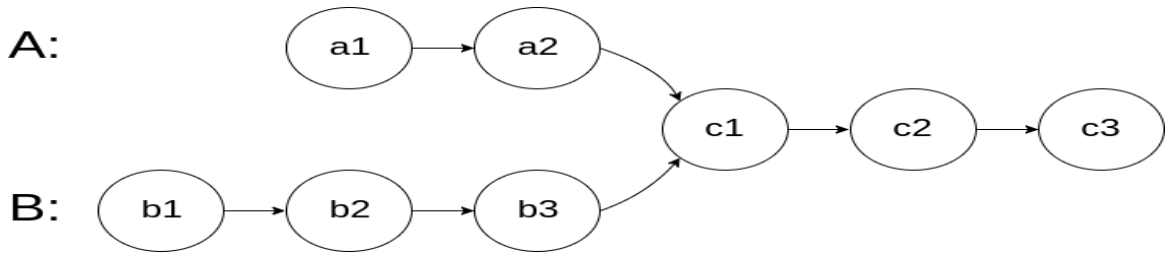
You are provided with two singly linked lists containing integers, where both lists converge at some node belonging to a third linked list.

Your task is to determine the data of the node at which they start merging. If no merging occurs, return -1.

Example: The linked lists are structured as follows:

```
First Linked List: a1, a2, c1
Second Linked List: b1, b2, b3, c1
Third Linked List where merging occurs: c1, c2, c3
```

An image shows the merging:



Input:

Three lines representing three singly linked lists.
Each line contains elements separated by spaces, ending with -1.
Line 1 (First linked list): a1, a2, ...an, c1, -1
Line 2 (Second linked list): b1, b2, ...bm, c1, -1
Line 3 (Third linked list): c2, c3, ...ck, -1

Output:

A single integer representing the data of the node where the merging starts. Output -1 if no merging occurs. You don't need to print this yourself as it is managed automatically.

Q. Rearrange Linked List -

Given a singly linked list in the form 'L1' -> 'L2' -> 'L3' -> ... 'Ln', your task is to rearrange the nodes to the form 'L1' -> 'Ln' -> 'L2' -> 'Ln-1', etc. You must not alter the data of the nodes.

Example:

Input:
```
1 -> 2 -> 3 -> 4 -> 5 -> NULL
```
Output:
```
1 -> 5 -> 2 -> 4 -> 3 -> NULL
```

Input:

The first line contains an integer 'T' representing the number of test cases. Each test case consists of elements of the linked list separated by spaces and terminated by -1.

Output:

For each test case, output a single line of the linked list in rearranged form, separated by spaces and terminated by -1.

Q. Flatten Binary Tree to Linked List

Transform a binary tree with integer values into a linked list by ensuring the linked list's nodes follow the pre-order traversal order of the binary tree.

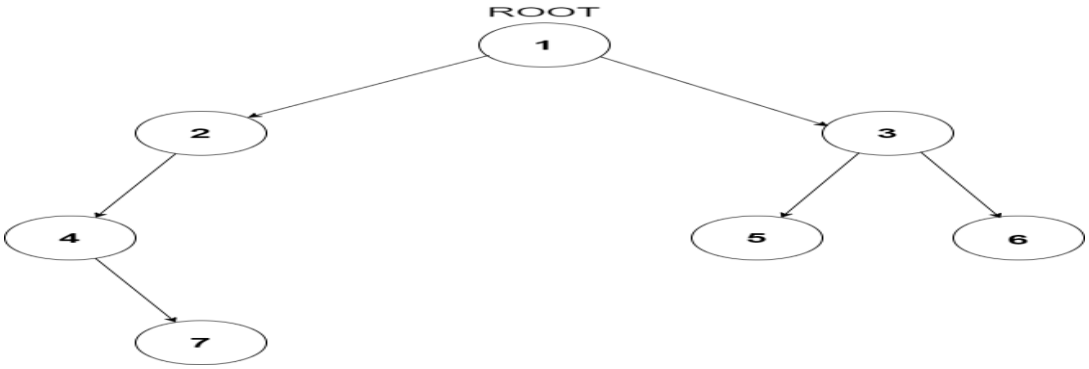
Explanation:

Utilize the binary tree's right pointer as the linked list's 'next' pointer, and set each node's left pointer to NULL.

Input:
The first line of input contains an integer 'T' denoting the number of test cases.
The subsequent lines for each test case contain the binary tree's elements in level order format, using `-1` to indicate null nodes.

Output:
Print the nodes of the linked list for each test case, separated by spaces. Display each test case's output on a new line.

Example:



Input:
1
1 2 3 4 -1 5 6 -1 7 -1 -1 -1 -1 -1 -1

Output:
1 2 4 7 3 5 6

Q. Add Two Numbers Represented by Linked Lists

Your task is to find the sum list of two numbers represented by linked lists and return the head of the sum list.

Explanation:

The sum list should be a linked list representation of the addition of the two numbers.

Input:

The first line of input contains a single integer T, indicating the number of test cases to be processed.

For each test case, there are two lines of input:
- The first line contains the elements of the first linked list, separated by spaces and ending with -1. The -1 is not part of the list elements.
- The second line contains the elements of the second linked list, also separated by spaces and ending with -1.

Output:

For each test case, print the sum linked list with elements separated by spaces and ending with -1.
Each test case should have its output on a new line.

Example:

Input:
2
7 5 9 4 6 -1
8 4 -1
5 6 3 -1
8 4 2 -1

Output:
5 0 0 5 0 -1
1 4 0 6 -1

Q. Multiply Linked Lists

Your task is to multiply two numbers represented as linked lists and return the resultant multiplied linked list.

Explanation:

The multiplied list should be a linked list representation of the product of the two numbers. Each node contains a single digit.

Input:

The first line contains an integer T, the number of test cases.
For each test case, the first line contains elements of the first linked list, separated by spaces and terminated by -1.
The second line contains elements of the second linked list, separated by spaces and terminated by -1.

Output:

For each test case, print the linked list representing the product, with elements single-space separated and terminated by -1.

Example:

Input:

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

Output:

```
1 5 8 1 0 -1
4 4 2 0 4 -1
```

Q. Railway Seat Berth Determination

Given a railway seat number represented as an integer, determine if it is a valid seat number and identify its berth type. Possible berth types include lower berth, middle berth, upper berth, side lower berth, and side upper berth.

Input:

The first line contains an integer 't', representing the number of test cases. Each of the next 't' lines contains one integer representing the railway seat number.

Output:

For each test case, output the berth type if the seat number is valid, otherwise output "Invalid". Possible berth types are "Lower", "Middle", "Upper", "Side Lower", and "Side Upper".

Example:

Input:

```
t = 2
 Seat numbers: 25
 Seat numbers: 38
```

Output:

```
Upper
 Invalid
```

Q. Matrix Symmetry Check

You are provided with a square matrix. Your task is to return `true` if the matrix is symmetric; otherwise, return `false`.

A symmetric matrix is characterized by its transpose being equal to the matrix itself.

Example of Symmetric Matrix:

Please refer to the example image below.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 8 \end{bmatrix}^{T}$$

Symmetric matrix

Input:

```
The first line contains an integer 'T', the number of test cases or queries to be executed.
Each test case starts with an integer 'N', representing the size of the square matrix.
The following line contains 'N' * 'N' integers, representing the matrix in a row-wise manner.
```

Output:

```
For each test case/query, output true if the matrix is symmetric; otherwise, output false.
Each result should be printed on a new line.
```

Q. Spiral Matrix Path -

Given a N x M matrix of integers, print the spiral order of the matrix.

Input:

The input starts with an integer 'T' representing the number of test cases. Each test case consists of:

1. **The first line contains two integers, N and M, representing the dimensions of the matrix.**
2. **The next N lines each contain M space-separated integers, representing the rows of the matrix.**

Output:

For each test case, output the spiral order of the given matrix in a single line.

Example:

Input:

```
T = 1
N = 3, M = 3
matrix =
1 2 3
4 5 6
7 8 9
```

Output:

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

Q. Most Frequent Non-Banned Word Problem

Given a paragraph consisting of letters in both lowercase and uppercase, spaces, and punctuation, along with a list of banned words, your task is to find the most frequent word that is not in the list of banned words. The solution will always exist and be unique.

While considering words, treat letters as case-insensitive (e.g., 'AsK' and 'aSK' are the same). Words consist solely of alphabets, separated by spaces or punctuation, and the result should be returned in uppercase.

Example:

Input:

```
Paragraph = "It's a square SqUare. It's a FLAT flat."
N = 3
BANNEDWORDS = [ "FLAT", "IT", "S" ]
```

Output:

```
SQUARE
```

Explanation:

The most frequent words (case-insensitive) are [IT, S, SQUARE, FLAT], each appearing twice. After checking with the banned list, 'SQUARE' remains as the most frequent word not banned, with a frequency of 2.
