Assignment 3

Instructions

1. Write your functions according to the signature provided below in a python(.py) file and submit them to the autograder for evaluation.
2. The autograder has a limited number of attempts. Hence, test it thoroughly before submitting it for evaluation.
3. Save your python file as text(.txt) file and submit it to the canvas.
4. If the assignment has theoretical questions, upload those answers as a pdf file on the canvas.
5. Submission on the canvas is mandatory and the canvas submitted text file should be the same as that of the final submission of the autograder. If not, marks will be deducted.
6. No submission on canvas or no submission on autograder then marks will be deducted.
7. Access the auto grader at https://c200.luddy.indiana.edu.

Question 1

In the enigmatic world of Colorland there exists two kind of forces namely red force denoted with R (pulling to left) and blue force denoted with B (pulling to right). So if they are kept same distance from each other they cancel each other for ex: RB. Now if a space has RBRB(The R at index 0 is supported by B at index 1 and R at index 2 is supported by a B at index 3) or RRBB (The R at index 0 is supported by a B at index 3 and R at index 1 is supported by G at index 2) these would also have a net zero force in the system. But if we have RRBBB it would create an imbalance of forces. Also, if we have a BBRR would also create an imbalance as the red force has no right blue force to counteract and similarly for blue force. Your task is to complete the function allCombinationForces(n) given n where n is the no of red and blue forces allowed to be used in a system and you need to return how many possible combinations may exist in a system such that the force of the entire system is 0.

Examples

Example 1:

```
Input: n = 1
Output: ["RB"]
```

Explanation: The possible combinations for n=1 are [RB,BR] but we only take RB because BR would create an imbalanced system.

Example 2:

```
Input: n = 3
Output: ["RRRBBB","RRBRBB","RRBBRB","RBRRBB","RBRBRB"]
```

Explanation: These are the possible combinations for n = 3

Constraints

- Solve the question using recursion.
- n is a positive integer.

Function

```
def allCombinationForces(n:int)->List[str]:
  # Output all possible combinations for n
  return combinations
```

Question 2

King Josephus and his `n` soldiers are trapped in a cave by Roman soldiers. Instead of being captured by the enemy, they decided to execute themselves in the following way.

The rules are as follows:

- Start with the 1st soldier.
- Count the next k soldiers in the clockwise direction, including the person you started with. The counting wraps around the circle and may count some soldiers more than once.
- The last soldier you counted will be executed.
- If there are still more than one soldier in the circle, go back to step 2 starting from the soldier immediately clockwise of the soldier who was just lost and repeat.
- Else, the last soldier in the circle survives and will be captured by the enemy.

The king also wants to participate in the process but wants to be in a position that survives in the end and gets captured by the enemy instead of dying. Given the number of soldiers, `n`, and an integer `k`, return the position of the soldier that survives. Write a program that returns the position of the soldier that survives at the end of the process.

Constraints

- Solve the question using recursion.
- 1 <= k <= n <= 500

Function

```
def josephus_plot(n: int, k: int) -> int:
  # return one based index
  return 1
```

Question 3

In the bustling city of Stars Hollow, skyscrapers pierced the clouds. The city was renowned for its ever-evolving skyline, a testament to human innovation and architectural prowess. But with such towering structures came an intriguing puzzle: how could one determine, for each building, the number of buildings shorter than itself to its right?

Given the heights of Stars Hollow's skyscrapers, determine, for each building, how many other buildings to its right have a lower height.

Hint: Can you use something similar to merge sort?

Examples

Example 1:

```
Input: heights = [5, 4, 3, 2, 6, 3]
Output: [4, 3, 1, 0, 1, 0]
Explanation:
To the right of 5 there are 4 buildings of lesser height(4, 3, 2, 3).
To the right of 4 there are 3 buildings of lesser height (3, 2, 3).
To the right of 3 there is 1 building of lesser height (2).
To the right of 2 there are 0 buildings of lesser height ().
To the right of 6 there is 1 building of lesser height (3).
To the right of 3 there are 0 buildings of lesser height ().
```

Constraints

- Solve the problem in $O(nlogn)$
- Do not use any inbuilt sorting funtions

Function

```
def shorterBuildings(heights):
  # your code goes here
  return your answer as a list
```

Question 4

Determine whether a given binary tree is height-balanced by ensuring that the disparity in depth between any two subtrees of each node does not exceed one.

Examples

Example 1:

Input:

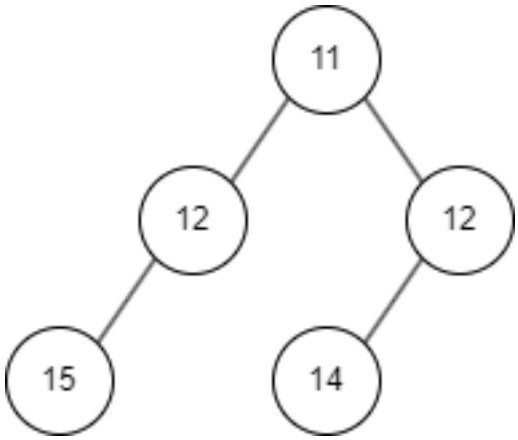

Figure 1: Alt text

```
Output: True
```

Example 2:

Input:

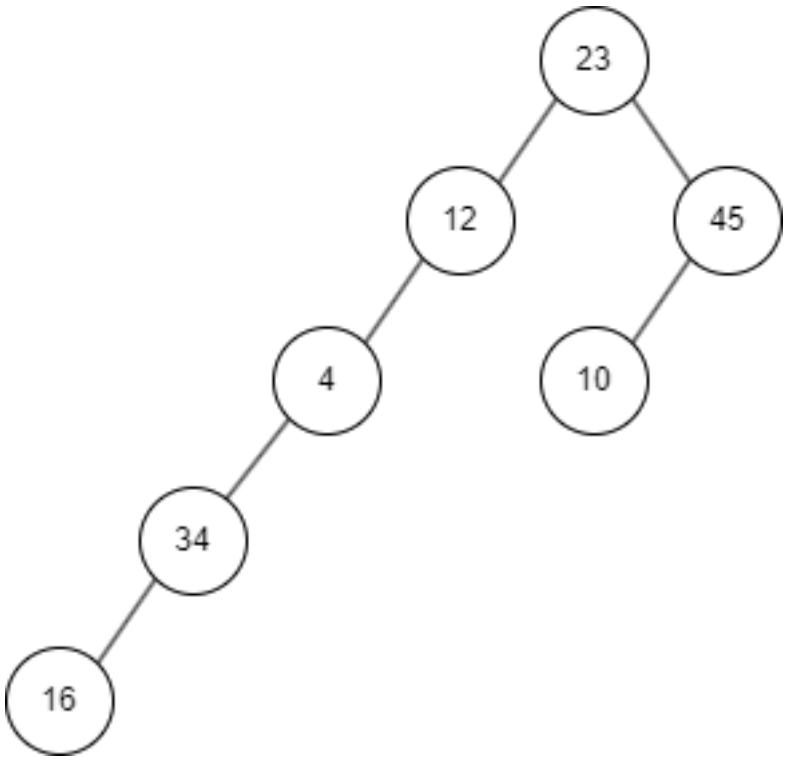

Figure 2: Alt text

```
Output: False
```

Function

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def isHeightBalanced(root: TreeNode) -> bool:
    pass
```

Question 5

In a mystical forest, two woodcutters, Tom and Emily, stumbled upon two arrays resembling a binary tree's pre-order and in-order traversals. Intrigued, they deciphered the arrays, reconstructing a magnificent binary tree.

Now, inspired by Tom and Emily's journey, you're invited to unravel the same problem: construct the binary tree from the pre-order and in-order traversals of the tree.

Examples

Example 1:

```
Input: preorder = [1, 2, 3, 4, 5], inorder = [3, 2, 1, 5, 4]
```

Output:

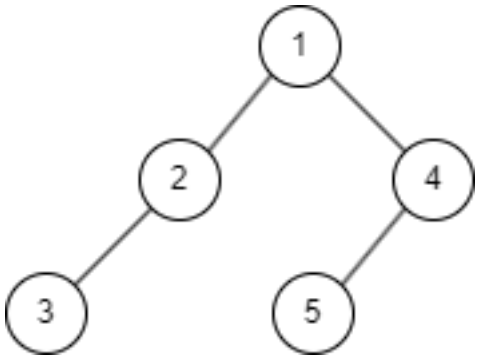

Figure 3: Alt text

Function

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def createBinaryTree(preorder: list[int], inorder: list[int]) -> TreeNode:
    pass
```

Question 6

Consider a binary tree where each node contains a digit from 1 to 9. Design an algorithm to find the number of paths from the root to a leaf that can be rearranged into a palindrome, essentially forming a jumbled palindrome.

Complete the given function to traverse the binary tree, exploring paths from the root to the leaf nodes. Identify and count the paths where the jumbled arrangement of digit values along the path can form a palindrome. The objective is to efficiently determine the number of such jumbled palindrome paths in the given binary tree.

Examples

Example 1:

```
Input: root = [2,3,1,3,1,null,1]
Output: 2
```

Explanation: The figure above represents the given binary tree. There are three paths going from the root node to leaf nodes: the red path [2,3,3], the green path [2,1,1], and the path [2,3,1]. Among these paths only red path and green path are jumbled palindromic paths since the red path [2,3,3] can be rearranged in [3,2,3] (palindrome) and the green path [2,1,1] can be rearranged in [1,2,1] (palindrome).

Example 2:

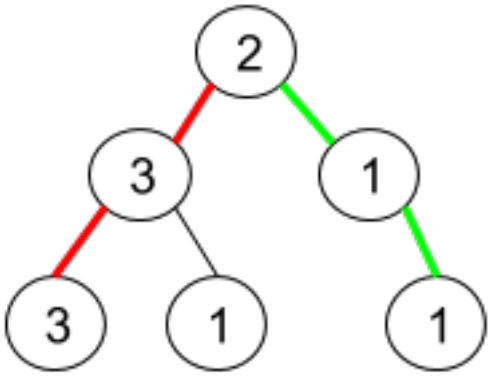

Figure 4: Alt text

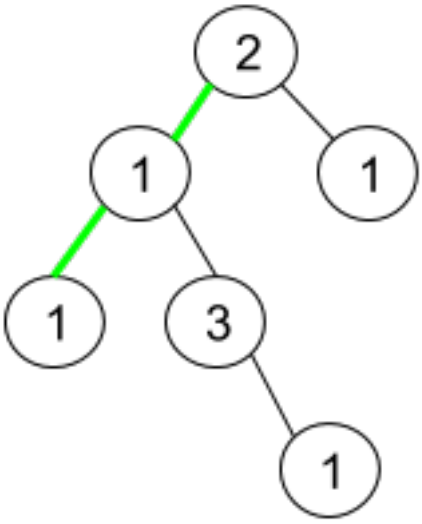

Figure 5: Alt text

```
Input: root = [2,1,1,1,3,null,null,null,null,null,1]
Output: 1
```

Explanation: The figure above represents the given binary tree. There are three paths going from the root node to leaf nodes: the green path [2,1,1], the path [2,1,3,1], and the path [2,1]. Among these paths only the green path is jumbled palindromic since [2,1,1] can be rearranged in [1,2,1] (palindrome).

Constraints

- Solve the problem in $O(n)$ Time complexity, where n = number of nodes
- $1 <= Node.val <= 9$

Function

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def palindromic_paths(root: TreeNode) -> int:
  # your code goes here
  return 0
```

Question 7

In a quiet town, where the streets were dimly lit by flickering lanterns, lived a brilliant inventor named Edison. Fueled by a desire to bring light to every home. One day, after numerous experiments, he unveiled his masterpiece - the electric light bulb. The first light bulb illuminated Edison's workshop, casting away the darkness that had lingered for so long. But Edison's dream was bigger – he wanted every home to bask in the warm glow of electric light.

With determination and innovation, Edison set up a network of electrical wires, connecting each house in the town to his new found creation. The moment arrived when Edison flipped the switch, and the town erupted in a brilliant glow as light passed from house to house, dispelling the darkness that had enveloped the community.

The entire network is given in form of a binary tree. Here the electrical wires represent the tree branches, and each node represents a house. Given the root of this binary tree and location of edison's workshop where the switch was first turned on, find the minimum time taken to illuminate the entire community. Rememeber, In just one minute, the electricity would pass from a node to its left child, right child, and parent.

Examples

Example 1:

```
Input: root = [1,5,3,null,4,10,6,9,2], location = 3
Output: 4
Explanation: The following nodes are illuminated during:
```

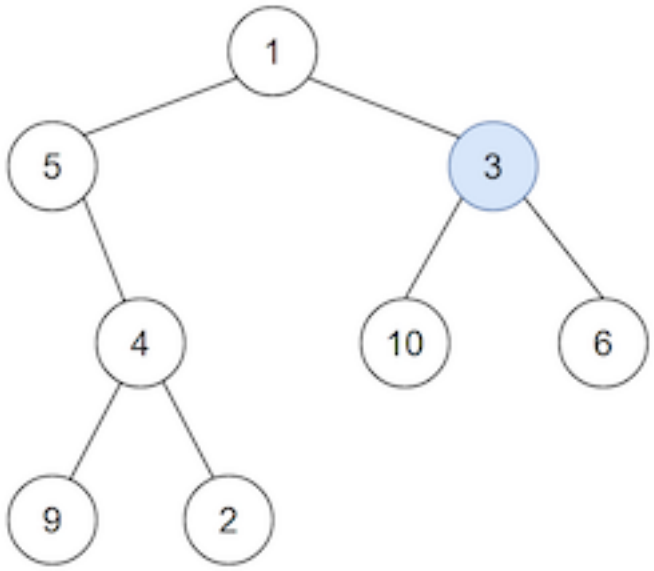

Figure 6: Alt text

```
- Minute 0: Node 3
- Minute 1: Nodes 1, 10 and 6
- Minute 2: Node 5
- Minute 3: Node 4
- Minute 4: Nodes 9 and 2
It takes 4 minutes for the whole tree to be illuminated so we return 4.
```

Constraints

- Solve the problem in $O(n)$ Time complexity, where n = number of nodes
- Space complexity should be $O(1)$ (You can ignore stack space if you are using recursion)
- All the node values will be unique numbers. A node with a value of location exists in the tree.

Function

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def minimumTime(root: TreeNode, location: int) -> int:
  # your code goes here
  return 0
```