Contributor Name: Vishal Motwani
Company Name:Amazon
Profile:Software Development Engineer (SDE) 2
Job Location:Bangalore
Applied[OnCampus/OffCampus]:on-campus

Round:1
Detailed Round Description
Online Coding Interview focused on Data Structures and Algorithms (DSA) and Object-Oriented Programming (OOPS).
Detailed Question Description(with Test Cases, if possible)
Coding Problems:3

Detailed Question Description
Coding Question:

Given a binary tree with N nodes, check if the input tree is a **Binary Search Tree (BST)** or not.
A BST must satisfy the following properties:

- The left subtree of a node contains only nodes with values **less than or equal** to the node's value.
- The right subtree of a node contains only nodes with values **greater than or equal** to the node's value.
- Both left and right subtrees must also be valid BSTs.

Code Logic Block

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

def is_valid_BST(root):
    def inorder_traversal(node, prev=[float('-inf')]):
        if not node:
            return True
        if not inorder_traversal(node.left, prev):
            return False
        if node.val <= prev[0]:  # Check if the current node is greater than the previous one
            return False
        prev[0] = node.val
        return inorder_traversal(node.right, prev)

    return inorder_traversal(root)
```

Detailed Question Description(with Test Cases, if possible)
Coding Question:
Given a **Binary Search Tree (BST)** with N nodes, modify it such that the value of each node is updated to the **sum of all nodes greater than or equal to it** in the BST.

A **Binary Search Tree (BST)** follows these properties:

- 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 left and right subtrees must also be valid BSTs.

Code Logic Block

Detailed Question Description(with Test Cases, if possible)
Coding Question:
You are given a string S consisting of "{" , "}" , "(" , ")" , "[" , "]".

Return True if the given string S is balanced, else return False

TestCases:Input: S = "{}()"
Output: True
Input: S = "{[()]}"
Output: True
Input: S = "{[}"
Output: False

Code Logic Block

```
def is_valid_parenthesis(s):
    stack = []
    mapping = {")": "(", "}": "{", "]": "["}

    for char in s:
        if char in mapping:
            top = stack.pop() if stack else "#"
            if mapping[char] != top:
                return False
        else:
            stack.append(char)

    return not stack
```

Each experience is unique in itself. And can help other developers who look up to you.
To contribute and help the LinuxSocials' Open Developer Community, use the format to write your experience and mail it here. [You can check your resource contribution on: https://www.linuxsocials.com]

Round:2
Detailed Round Description
Online Coding Interview focused on Data Structures and Algorithms (DSA) and Object-Oriented Programming (OOPS).
Coding Problems:2

Detailed Question Description(with Test Cases, if possible)
Coding Question:

Detailed Question Description (with Test Cases, if possible)

Coding Question:
You are given a **multi-level linked list** of `N` nodes. Each node has:

- A `next` pointer, which points to the next node in the same level.
- A `child` pointer, which may or may not point to a separate node (sublist).

Your task is to **flatten the multi-level linked list into a single-level linked list** by merging all levels into one.

Test Cases:
Input:

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

Output:
1 -> 2 -> 5 -> 6 -> 3 -> 4

Code Logic Block

```
class ListNode:
    def __init__(self, val=0, next=None, child=None):
        self.val = val
        self.next = next
        self.child = child

from collections import deque

def flatten_multilevel_linked_list(head):
    if not head:
        return None

    queue = deque([head])
    prev = None

    while queue:
        node = queue.popleft()

        if prev:
            prev.next = node  # Attach current node to the previous one

        prev = node

        if node.next:
            queue.append(node.next)

        if node.child:
            queue.append(node.child)
            node.child = None  # Remove child pointer after adding it to the queue

    return head
```

Detailed Question Description (with Test Cases, if possible)

Coding Question:
You are given an array `arr` of size `N` representing **elevation heights**. Each element `arr[i]` denotes the height of a bar at index `i`.

Find the total amount of **rainwater trapped** between these bars after raining.

Test Cases:Input: arr = [3, 0, 0, 2, 0, 4]
Output: 10

Explanation:
Total trapped rainwater = (3 - 0) + (3 - 0) + (2 - 0) + (4 - 2) + (4 - 0) = 10

Code Logic Block

```
def trap_rain_water(height):
    if not height:
        return 0

    n = len(height)
    left_max = [0] * n
    right_max = [0] * n
    water_trapped = 0

    # Compute left max for each index
    left_max[0] = height[0]
    for i in range(1, n):
        left_max[i] = max(left_max[i - 1], height[i])

    # Compute right max for each index
    right_max[n - 1] = height[n - 1]
    for i in range(n - 2, -1, -1):
        right_max[i] = max(right_max[i + 1], height[i])

    # Calculate trapped water
    for i in range(n):
        water_trapped += min(left_max[i], right_max[i]) - height[i]

    return water_trapped
```

Round:3
Detailed Round Description
System Design Question
Coding Problems:1
Detailed Question Description(with Test Cases, if possible)

Problem Statement:
Design an **Online Food Ordering System** similar to Swiggy/Zomato, where users can:

- Browse restaurants
- Select food items
- Place orders
- Track orders

The system should **handle high traffic, ensure scalability, and provide a seamless user experience**.

Round 4

Type: HR Round
Difficulty: Medium
Duration: 30 minutes

Questions:Discussed about past Projects, challenges faced, disagreements with manager, why looking for job switch, leadership principle questions were asked.

Round 5

Type: HR+Technical Round
Difficulty: Hard
Duration: 30 minutes

Questions:**Design ATM System**

Round 6

Type: HR Round
Difficulty: Easy
Duration: 30 minutes

Questions:Discussed about the Offer and Location.