

Outputs of challenges

Q1

The screenshot shows the VS Code interface with the following details:

- Code Editor:** The file `challenge1teamcontributor.py` is open. The code defines a function `teamContribution` that calculates a weighted product of contributions based on their index. It then prints the result.
- Terminal:** The terminal shows the command `python "c:/anas_3rd_sem/logic_forge/challenge1teamcontributor.py"` being run, resulting in the output `Impact Array: [24, 12, 8, 6]`.
- Bottom Bar:** The status bar indicates the file is Python, has 3.13.5, and is at Line 8, Column 22.

Q2

The screenshot shows the VS Code interface with the following details:

- Code Editor:** The file `challenge2passwordrecoverywindow.py` is open. The code implements a sliding window approach to find the minimum window containing all unique characters of a pattern in a log string.
- Terminal:** The terminal shows the command `python "c:/anas_3rd_sem/logic_forge/challenge2passwordrecoverywindow.py"` being run. It prompts for a log string and a pattern, then prints the minimum window substring.
- Bottom Bar:** The status bar indicates the file is Python, has 3.13.5, and is at Line 8, Column 22.

Q3

A screenshot of the Visual Studio Code interface. The top bar shows several tabs: challenge1teamcontributor.py, Workspace Trust, challenge3balancedperformancescore.py (which is the active tab), challenge6TOH.py, challenge2passwordrecoverywindow.py, challenge5brokenexp.py, and challenge4inventorysearch.py. The Explorer sidebar on the left has a single item: medianSortedArrays(scoresA, scoresB). The main editor area contains the following Python code:

```
2 # Ensure A is smaller array
3 if len(scoresA) > len(scoresB):
4     scoresA, scoresB = scoresB, scoresA
5
6 m, n = len(scoresA), len(scoresB)
7 low, high = 0, m
8
9 while low <= high:
10     # Partition positions
11     partA = (low + high) // 2
12     partB = (m + n + 1) // 2 - partA
13
14     # Left and Right values
15     maxLeftA = float("-inf") if partA == 0 else scoresA[partA - 1]
16     minRightA = float("inf") if partA == m else scoresA[partA]
17
18     maxLeftB = float("-inf") if partB == 0 else scoresB[partB - 1]
19     minRightB = float("inf") if partB == n else scoresB[partB]
20
21     # Correct partition found
```

The terminal below the editor shows the execution of the script:

```
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic_forge/challenge3balancedperformancescore.py"
Enter scores of Team A: 12 31 45 56 63
Enter scores of Team B: 56 39 29 52 9
Median Score: 38.0
PS C:\Users\anass>
```

Q5

A screenshot of the Visual Studio Code interface. The top bar shows several tabs: sr.py, Workspace Trust, challenge3balancedperformancescore.py, challenge6TOH.py, challenge2passwordrecoverywindow.py, challenge5brokenexp.py, and challenge4inventorysearch.py (which is the active tab). The Explorer sidebar on the left has a single item: removeInvalidParentheses(expr):. The main editor area contains the following Python code:

```
1 def removeInvalidParentheses(expr):
2     result = set()
3
4     # Step 1: Count extra parentheses
5     left_rem = right_rem = 0
6     for ch in expr:
7         if ch == '(':
8             left_rem += 1
9         elif ch == ')':
10             if left_rem == 0:
11                 right_rem += 1
12             else:
13                 left_rem -= 1
14
15     # Step 2: Backtracking
16     def backtrack(index, left_count, right_count, left_rem, right_rem, path):
17         # End of string
18         if index == len(expr):
19             if left_rem == 0 and right_rem == 0:
20                 result.add(path)
21             return
22
23     backtrack(0, 0, 0, left_rem, right_rem, "")
24
25     return result
```

The terminal below the editor shows the execution of the script:

```
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic_forge/challenge4inventorysearch.py"
Enter expression: 1231(sndrfmw213(dkal(jd,wbo)srf1wdn1))dqnl1(nddk)
Valid expressions with minimum removals:
1231(sndrfmw213(dkal(jd,wbo)srf1wdn1))dqnl1(nddk)
1231(sndrfmw213(dkal(jd,wbo)srf1wdn1))dqnl1(nddk)
1231(sndrfmw213(dkal(jd,wbo)srf1wdn1))dqnl1(nddk)
PS C:\Users\anass>
```

Q6

A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file, search, and other workspace functions. The main editor area contains a Python script named `towerOfHanoi.py`. The code implements the Tower of Hanoi algorithm using recursion:

```
1 def towerOfHanoi(n, from_rod, to_rod, aux_rod):
2     if n == 1:
3         print(f"Disk 1 moved from {from_rod} to {to_rod}")
4         return
5
6     # Step 1: Move n-1 disks from source to auxiliary
7     towerOfHanoi(n - 1, from_rod, aux_rod, to_rod)
8
9     # Step 2: Move nth disk from source to destination
10    print(f"Disk {n} moved from {from_rod} to {to_rod}")
11
12    # Step 3: Move n-1 disks from auxiliary to destination
13    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod)
```

The terminal tab at the bottom shows the execution of the script and the resulting moves:

```
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic_forge/challenge1teamcontributor.py"
Impact Array: [24, 12, 8, 6]
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic_forge/challenge6TOH.py"
Enter number of disks: 4
Disk 1 moved From A to B
Disk 2 moved From A to C
Disk 1 moved From B to C
Disk 2 moved From A to C
Disk 1 moved From B to C
Disk 3 moved From A to B
Disk 1 moved From C to A
Disk 2 moved From C to B
Disk 1 moved From A to B
Disk 4 moved From A to C
Disk 1 moved From B to C
Disk 2 moved From B to A
Disk 1 moved From C to A
Disk 3 moved From B to C
Disk 1 moved From A to B
Disk 2 moved From A to C
Disk 1 moved From B to C
PS C:\Users\anass> []
```

Q4

A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file, search, and other workspace functions. The main editor area contains a Python script named `kthSmallest.py`. The code uses a min-heap to find the k-th smallest element in a matrix:

```
1 import heapq
2
3 def kthSmallest(matrix, k):
4     n = len(matrix)
5
6     # Min-heap banayenge
7     heap = []
8
9     # Step 1: har row ka first element heap mein daalo
10    for row in range(n):
11        heapq.heappush(heap, matrix[row][0])
```

The terminal tab at the bottom shows the execution of the script and the result:

```
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic_forge/challenge4inventorysearch.py"
Kth smallest element is: 13
PS C:\Users\anass> []
```