

Outputs of challenges

Q1

```
challenge1teamcontributor.py X Workspace Trust challenge3balancedperformancescore.py challenge6TOH.py challenge2passwordrecoverywindow.py challenge5brokenexp.py che ▶ ▢ ...  
C:\> anas_3rd_sem > logic forge > challenge1teamcontributor.py > ...  
1 def teamContribution(contributions):  
13     # Step 2: Right product  
14     right_product = 1  
15     for i in range(n - 1, -1, -1):  
16         impact[i] *= right_product  
17         right_product *= contributions[i]  
18  
19     return impact  
20  
21  
22 # Driver / Client Code  
23 if __name__ == "__main__":  
24     contributions = [1, 2, 3, 4]  
25     result = teamContribution(contributions)  
26     print("Impact Array:", result)  
27  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
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>
```

Q2

```
challenge1teamcontributor.py X Workspace Trust challenge3balancedperformancescore.py challenge6TOH.py challenge2passwordrecoverywindow.py X challenge5brokenexp.py che ▶ ▢ ...  
C:\> anas_3rd_sem > logic forge > challenge2passwordrecoverywindow.py > minWindow  
1 def minWindow(log, pattern):  
2     if not log or not pattern:  
3         return ""  
4  
5     from collections import Counter  
6  
7     # Frequency of characters in pattern  
8     pattern_count = Counter(pattern)  
9  
10    required = len(pattern_count) # unique chars needed  
11    formed = 0 # chars matched with required frequency  
12  
13    window_count = {}  
14  
15    left = 0  
16    min_len = float("inf")  
17    min_window = ""  
18    for right in range(len(log)):  
19        char = log[right]  
20        window_count[char] = window_count.get(char, 0) + 1  
21  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\anass> & C:\Users\anass\AppData\Local\Programs\Python\Python313\python.exe "c:/anas_3rd_sem/logic forge/challenge1teamcontributor.py"  
Kth Smallest Element: 13  
PS C:\Users\anass> & C:\Users\anass\AppData\Local\Programs\Python\Python313\python.exe "c:/anas_3rd_sem/logic forge/challenge2passwordrecoverywindow.py"  
Enter log string: anskdnadfanafaisalanswaerdfknfkwsan  
Enter pattern string: san  
Minimum Window Substring: ans  
PS C:\Users\anass>  
OVR Ln 8, Col 22 Spaces: 4 UTF-8 CRLF {} Python 3.13.5
```

Q3

The screenshot shows a VS Code editor with a workspace containing several Python files. The active file is `challenge3balancedperformancescore.py`. The code implements a function `findMedianSortedArrays` that takes two sorted arrays, `scoresA` and `scoresB`, and returns the median of the combined array. The algorithm uses a binary search approach to find the partition point where the number of elements on the left is equal to the number of elements on the right. The terminal output shows the execution of the script, where the user enters scores for Team A and Team B, and the program outputs the median score.

```
def findMedianSortedArrays(scoresA, scoresB):
    # Ensure A is smaller array
    if len(scoresA) > len(scoresB):
        scoresA, scoresB = scoresB, scoresA

    m, n = len(scoresA), len(scoresB)
    low, high = 0, m

    while low <= high:
        # Partition positions
        partA = (low + high) // 2
        partB = (m + n + 1) // 2 - partA

        # Left and Right values
        maxLeftA = float("-inf") if partA == 0 else scoresA[partA - 1]
        minRightA = float("inf") if partA == m else scoresA[partA]

        maxLeftB = float("-inf") if partB == 0 else scoresB[partB - 1]
        minRightB = float("inf") if partB == n else scoresB[partB]

        # Correct partition found
        if maxLeftA < minRightB and maxLeftB < minRightA:
            return (maxLeftA + minRightB) / 2
        elif maxLeftA > minRightB:
            low = partA + 1
        else:
            high = partA - 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

The screenshot shows a VS Code editor with a workspace containing several Python files. The active file is `challenge4inventorysearch.py`. The code implements a function `removeInvalidParentheses` that takes a string `expr` and returns a list of valid expressions with the minimum number of parentheses removed. The algorithm uses a two-step process: first, it counts the number of extra left and right parentheses, and then it uses backtracking to generate all valid expressions by removing the minimum number of parentheses. The terminal output shows the execution of the script, where the user enters an expression, and the program outputs the valid expressions with the minimum number of removals.

```
def removeInvalidParentheses(expr):
    result = set()

    # Step 1: Count extra parentheses
    left_rem = right_rem = 0
    for ch in expr:
        if ch == '(':
            left_rem += 1
        elif ch == ')':
            if left_rem == 0:
                right_rem += 1
            else:
                left_rem -= 1

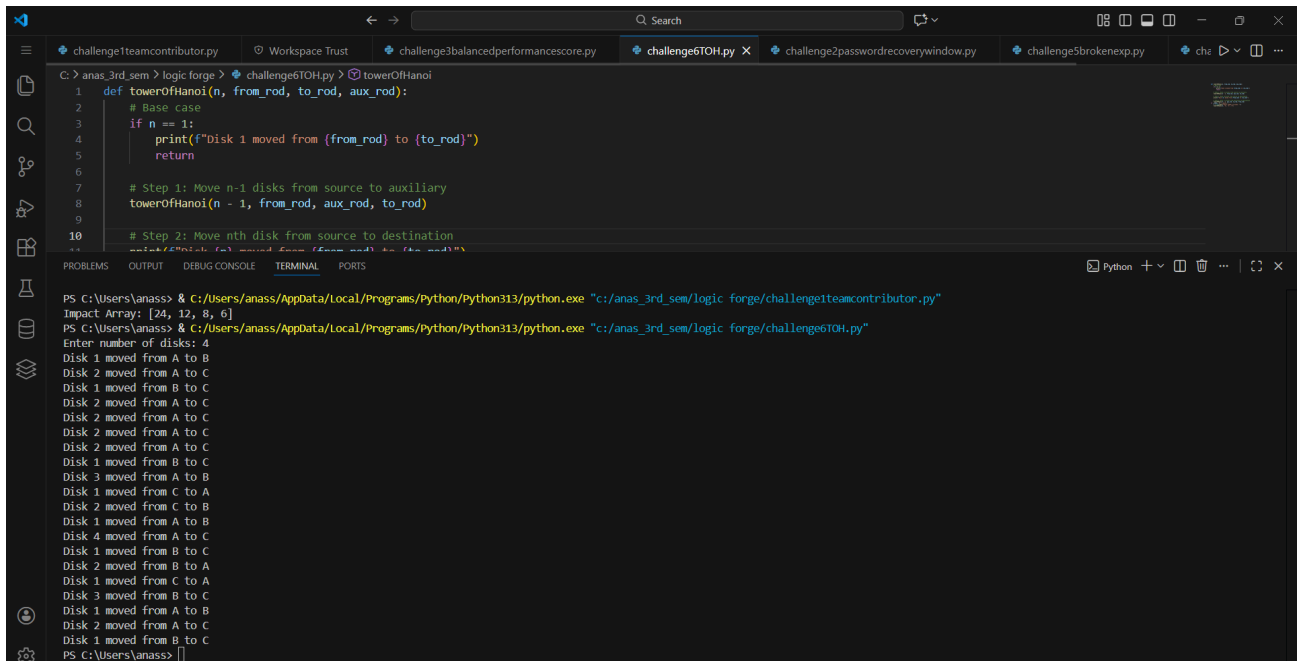
    # Step 2: Backtracking
    def backtrack(index, left_count, right_count, left_rem, right_rem, path):
        # End of string
        if index == len(expr):
            if left_rem == 0 and right_rem == 0:
                result.add(path)
            return

        if left_count < left_rem:
            backtrack(index + 1, left_count + 1, right_count, left_rem, right_rem, path + '(')
        if right_count < right_rem:
            backtrack(index + 1, left_count, right_count + 1, left_rem, right_rem, path + ')')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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(sndnfbmw1213(dkal(jd,wbo)snflwdn1))dqnlD(nddk)
Valid expressions with minimum removals:
1231(sndnfbmw1213(dkal(jd,wbo)snflwdn1))dqnlD(nddk)
1231(sndnfbmw1213(dkal(jd,wbo)snflwdn1))dqnlD(nddk)
1231(sndnfbmw1213(dkal(jd,wbo)snflwdn1))dqnlD(nddk)
PS C:\Users\anass>

Q6



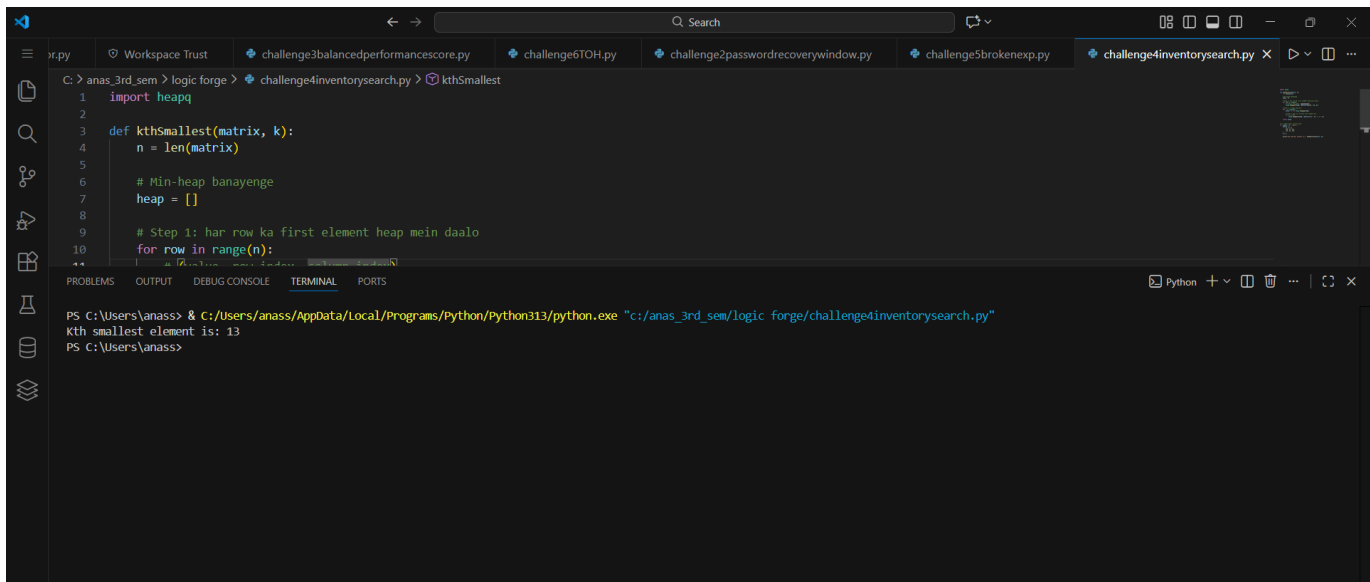
The screenshot shows a VS Code editor with a Python script for the Tower of Hanoi problem. The script is named `challenge6TOH.py` and is located in the `challenge6TOH.py` tab. The script defines a function `towerOfHanoi(n, from_rod, to_rod, aux_rod)` that takes the number of disks `n`, the source rod `from_rod`, the destination rod `to_rod`, and the auxiliary rod `aux_rod` as arguments. The function implements the recursive algorithm for the Tower of Hanoi problem. The terminal output shows the execution of the script, which prints the sequence of moves for 4 disks.

```
def towerOfHanoi(n, from_rod, to_rod, aux_rod):  
    # Base case  
    if n == 1:  
        print(f'Disk 1 moved from {from_rod} to {to_rod}")  
        return  
    # Step 1: Move n-1 disks from source to auxiliary  
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod)  
    # Step 2: Move nth disk from source to destination  
    print(f'Disk {n} moved from {from_rod} to {to_rod}")  
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod)
```

Terminal Output:

```
PS C:\Users\anass> & C:/Users/anass/AppData/Local/Programs/Python/Python313/python.exe "c:/anas_3rd_sem/logic forge/challenge6TOH.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 2 moved from A 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



The screenshot shows a VS Code editor with a Python script for finding the kth smallest element in a matrix. The script is named `challenge4inventorysearch.py` and is located in the `challenge4inventorysearch.py` tab. The script defines a function `kthSmallest(matrix, k)` that takes a matrix and a value `k` as arguments. The function implements a min-heap algorithm to find the kth smallest element in the matrix. The terminal output shows the execution of the script, which prints the kth smallest element in the matrix.

```
import heapq  
  
def kthSmallest(matrix, k):  
    n = len(matrix)  
    # Min-heap banayenge  
    heap = []  
    # Step 1: har row ka first element heap mein daalo  
    for row in range(n):  
        heapq.heappush(heap, matrix[row][0])  
    # Step 2: kth smallest element find karne ke liye k-1 element nikal denge  
    for i in range(1, k):  
        heapq.heappop(heap)  
    # Step 3: kth smallest element return kar denge  
    return heapq.heappop(heap)
```

Terminal Output:

```
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>
```