

Challenge 1:

The screenshot shows a VS Code editor with a file named `Challenge1_TeamContributionMultiplier.py`. The code implements a function `TeamContribution` that calculates the impact of team members' contributions. The terminal output shows the execution of the script, which prompts the user to enter the number of team members and their individual contributions. The output displays the impact after the prefix part and the final impact for each team member.

```
1 #Task 1
2
3 def TeamContribution(contributions):
4     length = len(contributions)
5     prefix = 1 #to calculate stuff to the left of current index
6     suffix = 1 #to calculate stuff to the right of current index
7
8     impact = [1]*length #impact vector, currently all 1s, will be updated in loops
9
10    #Calculating prefix part
11    for i in range(length):
12        impact[i] = impact[i] * prefix # [[1*1], [1*2], [1*2], [1*6]]
13        prefix = prefix * contributions[i] #prefix will update like: initial(1) > 1 > 2 > 6 > 24 (for 4 iterations)
14
15    print(f"Impact after prefix part: {impact}")
16    print(f"Prefix: {prefix}")
17
18    #Calculating suffix part
```

Terminal Output:

```
PS C:\Users\VARComputers\Desktop\VS code stuff\winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\winter\Challenge1\Challenge1_TeamContributionMultiplier.py"
Enter the number of team members: 4
Enter the contribution of team member 1: 1
Enter the contribution of team member 2: 2
Enter the contribution of team member 3: 3
Enter the contribution of team member 4: 4
Impact after prefix part: [1, 1, 2, 6]
Prefix: 24
Impact of each team member: [24, 12, 6, 6]
PS C:\Users\VARComputers\Desktop\VS code stuff\winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\winter\Challenge1\Challenge1_TeamContributionMultiplier.py"
Enter the number of team members: 5
Enter the contribution of team member 1: -1
Enter the contribution of team member 2: 1
Enter the contribution of team member 3: 0
Enter the contribution of team member 4: -3
Enter the contribution of team member 5: 3
Impact after prefix part: [1, -1, -1, 0, 0]
Prefix: 0
Impact of each team member: [0, 0, 0, 0, 0]
PS C:\Users\VARComputers\Desktop\VS code stuff\winter>
```

Challenge 2:

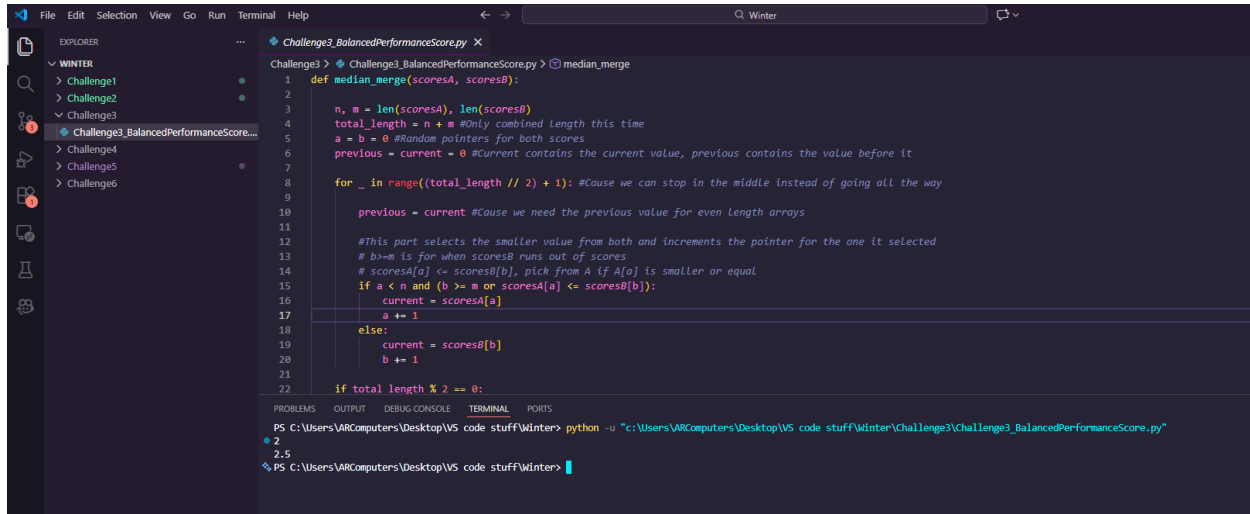
The screenshot shows a VS Code editor with a file named `Challenge2_PasswordRecovery.py`. The code implements a function `minWindow` that finds the smallest substring of a log string that matches a given pattern. The terminal output shows the execution of the script, which prompts the user to enter a log string and a pattern. The output displays the smallest substring that matches the pattern.

```
1 def minWindow(log: str, pattern: str) -> str:
2     if not log or not pattern:
3         return ""
4
5     freq = [0] * 128 #Creates frequency array for ASCII characters so like "A" will be "65" in integer value
6     for ch in pattern:
7         freq[ord(ch)] += 1 #ord() gets int value of the ASCII char, and this part increments count for each character in pattern
8
9     left = 0 #left pointer for the sliding window, this will shrink when we find req. window
10    needed = len(pattern) #total characters still needed to match
11    result = ""
12
13    for right in range(len(log)): #going right first
14        if freq[ord(log[right])] > 0: #if char is needed
15            needed -= 1
16            freq[ord(log[right])] -= 1 #reduce freq for current char
17
18        while needed == 0: #runs after we have valid window
19            if not result or (right - left + 1) < len(result): #checks if the current window is smaller than previous or not
20                result = log[left:right+1]
21
22        #removing left char from window
```

Terminal Output:

```
PS C:\Users\VARComputers\Desktop\VS code stuff\winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\winter\Challenge2\Challenge2_PasswordRecovery.py"
Enter log string: ADDBCCDEBANK
Enter pattern string: ABC
Smallest substring: ABC
PS C:\Users\VARComputers\Desktop\VS code stuff\winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\winter\Challenge2\Challenge2_PasswordRecovery.py"
Enter log string: a
Enter pattern string: a
Smallest substring: a
PS C:\Users\VARComputers\Desktop\VS code stuff\winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\winter\Challenge2\Challenge2_PasswordRecovery.py"
Enter log string: aa
Enter pattern string: aa
Smallest substring: ""
PS C:\Users\VARComputers\Desktop\VS code stuff\winter>
```

Challenge 3:



The screenshot shows a VS Code editor with a file named `Challenge3_BalancedPerformanceScore.py`. The code implements a `median_merge` function that takes two sorted arrays, `scoresA` and `scoresB`, and returns the median of the merged array. The function uses a two-pointer approach to merge the arrays in sorted order. The terminal shows the execution of the script, which outputs `2.5`.

```
def median_merge(scoresA, scoresB):
    n, m = len(scoresA), len(scoresB)
    total_length = n + m #Only combined length this time
    a = b = 0 #Random pointers for both scores
    previous = current = 0 #Current contains the current value, previous contains the value before it

    for _ in range((total_length // 2) + 1): #Cause we can stop in the middle instead of going all the way
        previous = current #Cause we need the previous value for even length arrays

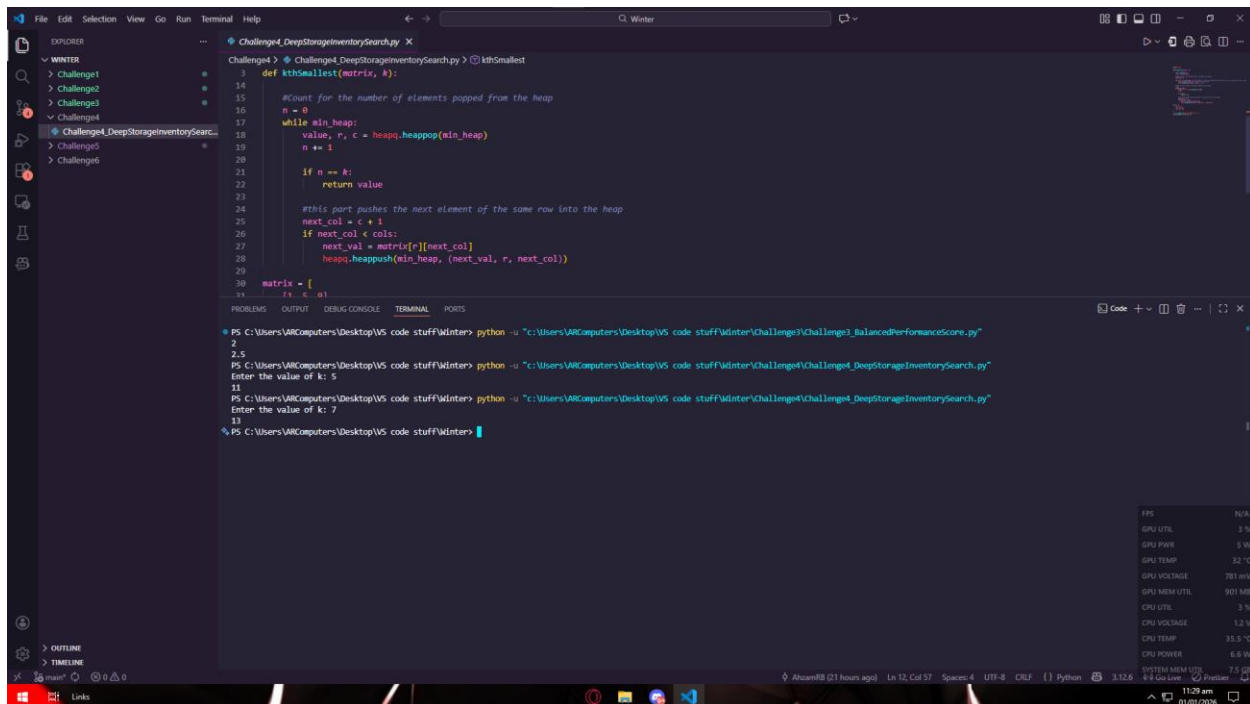
        #This part selects the smaller value from both and increments the pointer for the one it selected
        # b>m is for when scoresB runs out of scores
        # scoresA[a] <= scoresB[b], pick from A if A[a] is smaller or equal
        if a < n and (b >= m or scoresA[a] <= scoresB[b]):
            current = scoresA[a]
            a += 1
        else:
            current = scoresB[b]
            b += 1

    if total_length % 2 == 0:
        return (previous + current) / 2
    else:
        return current
```

Terminal output:

```
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\Winter\Challenge3\Challenge3_BalancedPerformanceScore.py"
2
2.5
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter>
```

Challenge 4:



The screenshot shows a VS Code editor with a file named `Challenge4_DeepStorageInventorySearch.py`. The code implements a `kthSmallest` function that takes a 2D matrix and a value `k`, and returns the `k`-th smallest element in the matrix. The function uses a min-heap to store the elements of the matrix in sorted order. The terminal shows the execution of the script, which outputs `11`.

```
def kthSmallest(matrix, k):
    #Count for the number of elements popped from the heap
    n = 0
    while min_heap:
        value, r, c = heapq.heappop(min_heap)
        n += 1

        if n == k:
            return value

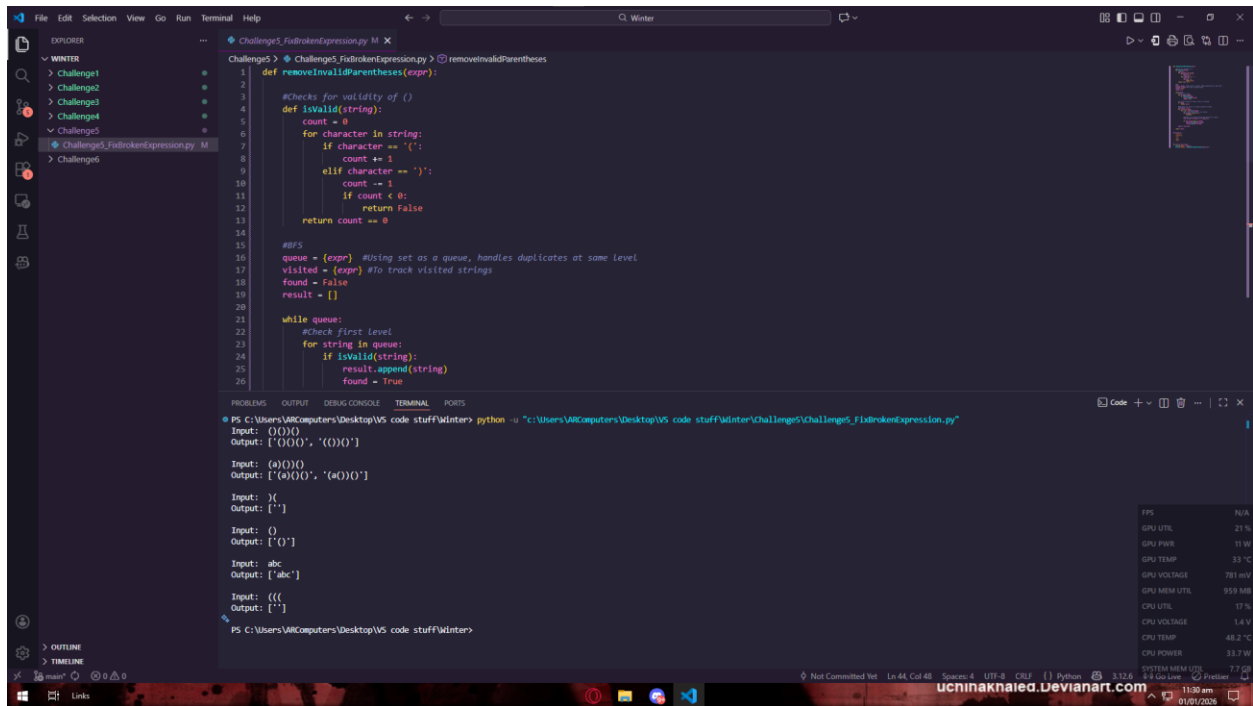
        #This part pushes the next element of the same row into the heap
        next_col = c + 1
        if next_col < cols:
            next_val = matrix[r][next_col]
            heapq.heappush(min_heap, (next_val, r, next_col))

    matrix = [
        [1, 5, 9],
        [10, 11, 13],
        [12, 14, 16]
    ]
```

Terminal output:

```
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\Winter\Challenge3\Challenge3_BalancedPerformanceScore.py"
2
2.5
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\Winter\Challenge4\Challenge4_DeepStorageInventorySearch.py"
Enter the value of k: 5
11
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VARComputers\Desktop\VS code stuff\Winter\Challenge4\Challenge4_DeepStorageInventorySearch.py"
Enter the value of k: 7
13
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter>
```

Challenge 5:



Challenge 6:

