

## Challenge 1:

The screenshot shows the VS Code interface with the file `Challenge1_TeamContributionMultiplier.py` open in the editor. The code implements a prefix sum algorithm to calculate team contributions. It defines a function `TeamContribution` that takes a list of contributions and returns a list where each element is the product of all previous elements in the input list. The terminal window shows the execution of the script with sample input and output.

```
def TeamContribution(contributions):
    impact = [1]*length #impact vector, currently all 1s, will be updated in loops
    #calculating prefix part
    for i in range(length):
        impact[i] = impact[i] * prefix # [[1*1], [1*1], [1*2], [1*6]]
        prefix = prefix * contributions[i] #prefix will update like: initial(1) > 1 > 2 > 6 > 24 (for 4 iterations)
    print("Impact after prefix part: (impact)")
    print("Prefix: (prefix)")

    #calculating suffix part
```

```
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:
Suffix: 24
Impact of each team member: [24, 12, 8, 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:
Suffix: 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 the VS Code interface with the file `Challenge2_PasswordRecovery.py` open in the editor. The code implements a sliding window approach to find the smallest substring of a log string that matches a given pattern. It uses a frequency array to count occurrences of each character in the pattern and then slides a window across the log string to find a match. The terminal window shows the execution of the script with sample inputs and outputs.

```
def minWindow(log, patern, size):
    if not log or not patern:
        return ""

    freq = [0] * 128 #Creates frequency array for ASCII characters so like "A" will be "65" in integer value
    for ch in patern:
        freq[ord(ch)] += 1 #ord() gets int value of the ASCII char, and this part increments count for each character in pattern

    left = 0 #left pointer for the sliding window, this will shrink when we find req. window
    needed = len(patern) #total characters still needed to match
    result = ""

    for right in range(len(log)): #going right first
        if freq[ord(log[right])] > 0: #if char is needed
            needed -= 1
        freq[ord(log[right])] -= 1 #reduce freq for current char

        while needed == 0: #means after we have valid window
            if right - left + 1 < len(result): #Checks if the current window is smaller than previous or not
                result = log[left:right]
            left += 1 #Removing Left char from window

    return result
```

```
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: ABCDECODEBANC
Enter pattern string: ABC
Smallest substring: BANC
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: a
Enter pattern string: aa
Smallest substring: "
```

### Challenge 3:

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

- File Explorer:** Shows a tree view with a folder named "WINTER" containing sub-folders "Challenge1", "Challenge2", "Challenge3", and "Challenge4" through "Challenge6".
- Editor:** A Python file named "Challenge3\_BalancedPerformanceScore.py" is open. The code implements a "median\_merge" function that takes two sorted arrays, "scoresA" and "scoresB", and merges them into a single sorted array while maintaining a balanced performance score. The code uses two pointers, "a" and "b", to track the current elements being compared. It handles cases where one array has run out of elements by continuing with the other. The code is annotated with comments explaining its logic.
- Terminal:** At the bottom, the terminal window shows the command "python -u "c:/Users/ARComputers/Desktop/VS code stuff/Winter/Challenge3/Challenge3\_BalancedPerformanceScore.py"" being run, with the output "2" and "2.5" displayed.

### Challenge 4:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward navigation bar. The left sidebar has sections for Explorer, Search, Problems, Output, Debug Console, Terminal, and Ports. The main area displays a Python file named `Challenged_DeepStorageInventorySearch.py`. The code implements a function `kthSmallest` that finds the k-th smallest element in a matrix using a min-heap. It also includes logic to push the next element of the same row into the heap after popping an element. The terminal below shows the command `python -u "c:/Users/VARComputers/Desktop/VS code stuff/Winter/Challenged/Challenged4_DeepStorageInventorySearch.py"` being run, followed by the user inputting the value of `k` as 7. The status bar at the bottom indicates the file is `Alcamo8 (21 hours ago)`, the terminal has 1129 lines, and the Python version is 3.12.6.

```
File Edit Selection View Go Run Terminal Help ← → C:\Winter EXPLORER ... Challenged > Challenged_DeepStorageInventorySearch.py ⓘ kthSmallest
  def kthSmallest(matrix, k):
    min_heap = []
    n = len(matrix)
    if n == 0:
      return None
    for i in range(n):
      heapq.heappush(min_heap, (matrix[i][0], i, 0))
    count = 0
    while min_heap:
      value, r, c = heapq.heappop(min_heap)
      count += 1
      if count == k:
        return value
      next_col = c + 1
      if next_col < n:
        next_val = matrix[r][next_col]
        heapq.heappush(min_heap, (next_val, r, next_col))
    matrix = [
      [r[c] for c in range(n)]
    ]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:/Users/VARComputers/Desktop/VS code stuff/Winter/Challenged/Challenged4_DeepStorageInventorySearch.py"
2
5
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:/Users/VARComputers/Desktop/VS code stuff/Winter/Challenged/Challenged4_DeepStorageInventorySearch.py"
Enter the value of k:
5
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> python -u "c:/Users/VARComputers/Desktop/VS code stuff/Winter/Challenged/Challenged4_DeepStorageInventorySearch.py"
Enter the value of k:
7
13
PS C:\Users\VARComputers\Desktop\VS code stuff\Winter> 
```

FPS N/A  
GPU UTIL 3 %  
GPU PWR 5 W  
GPU TEMP 32 °C  
GPU VOLTAGE 781 mV  
GPU MEM UTIL 901 MB  
CPU UTIL 3 %  
CPU VOLTAGE 1.2 V  
CPU TEMP 35.5 °C  
CPU POWER 6.8 W  
SYSTEM MEM UTIL 75.0 %  
GPU MEM UTIL 901 MB  
CPU UTIL 3 %  
CPU VOLTAGE 1.2 V  
CPU TEMP 35.5 °C  
CPU POWER 6.8 W  
OUTLINE  
CPU POWER  
TERMINAL  
TIMELINE  
main.py 3.12.6 1129 lines 09/06/2024  
Links

## Challenge 5:

## Challenge 6:

The screenshot shows a Windows desktop environment with a code editor and a terminal window open.

**Code Editor:** The left pane shows the file `Challenge6_TowerofHanoi.py` with the following content:

```
def tower_of_hanoi(n, from_rod, to_rod, aux_rod):
    if n == 1:
        print("Moved disk 1 from (%s) to (%s)" % (from_rod, to_rod))
        return
    tower_of_hanoi(n - 1, from_rod, aux_rod, to_rod)
    print("Moved disk (%d) from (%s) to (%s)" % (n, from_rod, to_rod))
    tower_of_hanoi(n - 1, aux_rod, to_rod, from_rod)

disks = int(input("Enter the number of disks: "))
tower_of_hanoi(disks, 'A', 'C')
```

The right pane shows the terminal window with the following session:

```
PS C:\Users\VAComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VAComputers\Desktop\VS code stuff\Winter\Challenges\Challenge6_TowerofHanoi.py"
Enter the number of disks: 3
Moved disk 3 from A to B
Moved disk 1 from C to A
Moved disk 2 from A to B
Moved disk 1 from A to B
Moved disk 4 from A to C
Moved disk 1 from B to C
Moved disk 2 from B to A
Moved disk 1 from A to C
Moved disk 3 from B to C
Moved disk 1 from A to B
Moved disk 2 from A to C
Moved disk 1 from C to B
Moved disk 3 from A to C
Moved disk 1 from A to B
Moved disk 2 from A to C
Moved disk 1 from C to A
PS C:\Users\VAComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VAComputers\Desktop\VS code stuff\Winter\Challenges\Challenge6_TowerofHanoi.py"
Enter the number of disks: 3
Moved disk 1 from A to C
Moved disk 2 from A to B
Moved disk 1 from B to A
Moved disk 3 from A to C
Moved disk 1 from B to A
Moved disk 2 from B to C
Moved disk 1 from A to C
PS C:\Users\VAComputers\Desktop\VS code stuff\Winter> python -u "c:\Users\VAComputers\Desktop\VS code stuff\Winter\Challenges\Challenge6_TowerofHanoi.py"
Enter the number of disks: 2
Moved disk 1 from A to B
Moved disk 2 from A to C
Moved disk 1 from B to C
PS C:\Users\VAComputers\Desktop\VS code stuff\Winter>
```

**System Status:** The bottom right corner shows system status icons for battery, signal, and volume.