Question - 1

Problem: Minimum Lead Blocks to Contain Radiation

You are given an integer array `radiation` where `radiation[i]` represents the **initial radiation level** at index `i`. Radiation attenuates (decreases) by **1 unit per step** as it spreads to neighboring indices.

You are also given an integer `m`. You may place **lead blocks** between two consecutive indices in the array. A lead block **prevents radiation from spreading further** across that boundary.

Your task is to determine the **minimum number of lead blocks** required such that, after radiation attenuation and blocking, the effective radiation level at **every index** is strictly less than `m`.

Example

Input

Radiation = [3, 1, 2, 1, 3]
M = 4

Output

2

Explanation

Step 1: Compute total radiation without blocks

Radiation at each index (summing contributions from every source):

- Index 0 → 3
- Index 1 → 4
- Index 2 → 4
- Index 3 → 4
- Index 4 → 3

So total = [3, 4, 4, 4, 3]

Step 2: Compare with the requirement (strictly less than m = 4)

- Index 0 → 3 < 4 ✅
- Index 1 → 4 ❌ (not strictly less)
- Index 2 → 4 ❌
- Index 3 → 4 ❌
- Index 4 → 3 < 4 ✅

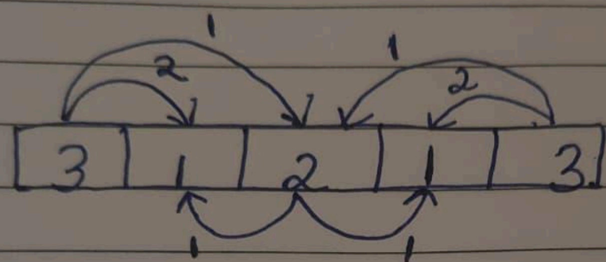
Indexes 1, 2, 3 are invalid.

Step 3: Place lead blocks to reduce radiation

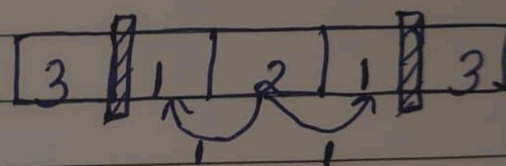
- Place a block between index 0 and 1:
  New radiation = [3, 2, 2, 2, 3]
- Check again with m = 4:
  - Index 0 → 3 < 4 ✅
  - Index 1 → 2 < 4 ✅
  - Index 2 → 2 < 4 ✅
  - Index 3 → 2 < 4 ✅
  - Index 4 → 3 < 4 ✅

Now all are strictly less than 4.

<u>IMAGE GIVEN BELOW FOR REFERENCE</u>



Radiation => 3 4 4 4 3

Using Lead Blocks => 

Radiation => 3 2 2 2 3

<u>Question - 2</u>

Problem: Maximum Sum of Differences of Two Subarrays

You are given an integer array `nums` and an integer `m`.

Your task is to select **two non-overlapping subarrays of size `m`** from `nums`. Define the "difference sum" as the sum of absolute differences of corresponding elements between the two subarrays.

Formally, if `A = nums[i .. i+m-1]` and `B = nums[j .. j+m-1]` (with `i+m-1 < j` or `j+m-1 < i` so they don't overlap), then the score is:

$$\text{score}(A, B) = \sum_{k=0}^{m-1} |A[k] - B[k]|$$

Return the **maximum possible score**.

Example 1:

Input: `nums = [1, 5, 2, 9, 6, 3], m = 2`
Output: `10`
Explanation:
Choose subarray `A = [2, 9]` and subarray `B = [6, 3]`.
The sum of absolute differences is:

∣2−6∣+∣9−3∣ = 4+6 = 10

<u>NOTE</u> => BRUTE FORCE IS TOO INEFFICIENT AS COMPLEXITY = O((N^2)*M).

TRY USING SLIDING WINDOW OR DYNAMIC PROGRAMMING

Question - 3

Based on Greedy Knapsack