1. **What is the expected running time of the following C# code? Explain why. Assume the array's size is n.**

**long Compute(int[] arr)**

**{**

**long count = 0;**

**for (int i=0; i<arr.Length; i++)**

**{**

**int start = 0, end = arr.Length-1;**

**while (start < end)**

**if (arr[start] < arr[end])**

**{ start++; count++; }**

**else**

**end--;**

**}**

**return count;**

**}**

The expected running time is **O(n^2).** In the worst case scenario, the array will be sorted ascending, so arr[start] will always be smaller than arr[end] 🡪 the required steps for traversal will be ~n for the inner loop.

2. **What is the expected running time of the following C# code? Explain why. Assume the input matrix has size of n \* m.**

**long CalcCount(int[,] matrix)**

**{**

**long count = 0;**

**for (int row=0; row<matrix.GetLength(0); row++)**

**if (matrix[row, 0] % 2 == 0)**

**for (int col=0; col<matrix.GetLength(1); col++)**

**if (matrix[row,col] > 0)**

**count++;**

**return count;**

**}**

The running time is quadratic **O(n\*m).** In the worst case scenario, each matrix[row, 0] is an even number, so the operation will be performed **n** times. The inner loop is always performed **m** times (the conditional check is done at each iteration).

3. **What is the expected running time of the following C# code? Explain why. Assume the input matrix has size of n \* m.**

**long CalcSum(int[,] matrix, int row)**

**{**

**long sum = 0;**

**for (int col = 0; col < matrix.GetLength(0); col++)**

**sum += matrix[row, col];**

**if (row + 1 < matrix.GetLength(1))**

**sum += CalcSum(matrix, row + 1);**

**return sum;**

**}**

**Console.WriteLine(CalcSum(matrix, 0));**

In the method body, the for-loop will run **m times**. After that the recursive call is made exactly 1 time for each method call (pay attention to the missing scope brackets). From the code it seems that, in the worst-case scenario, this call will be made **n times.** Thus, the running time becomes **O(n\*m).**