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;**

**}**

Външният цикъл ще се изпълни N на брой пъти. Вътрешният ще се изпълни също N на брой пъти – при всяка итерация интервала намалява с единица докато start и end не се изравнят. Следователно сложността на този метод е N2.

1. **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;**

**}**

Външият цикъл се изпълнява M на брой пъти, вътрешният само тогава, когато числото в първата колона на матрицата се дели на нула без остатък, тоест е четно. Ако броят четни числа в първата колона означим с К то сложността на метода става O((M-K) + K\*N). В зависимост от константата К сложността варира от линейна до квадратична – при 0 четни числа метода ще се изпълни само M пъти, а при М четни числа – M\*N пъти.

1. **\* 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));**

Тук чрез рекурсия се обхождат и сумират всички клетки на матрицата. Метода събира всички клетки на дадения ред и вика рекурсивно себе си за следващия ред, ако такъв съществува. Сложността отново е N\*M – тоест квадратична сложност.

П.С. – Има грешка при вземане на размерите на матрицата – при колоните трябва да се взема Getlength(1), а при редовете - Getlength(0).