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;
}
```

Отговор: Външният цикъл се изпълнява O(n) пъти, като на всяка итерация изпълнява вътрешния по O(n) пъти (всеки път разликата end - start намалява с 1, докато не стане 0) => $O(n^2)$

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;
```

```
}
```

Отговор: O(n-z + z*m), където z е броя четни числа в първата колона на matrix. Причина: външният цикъл се изпълнява n пъти, а във вътрешния се влиза точно когато първото числото в реда е четно => z пъти. Ако z се приеме за константа, имаме O(n + m); ако z е от порядъка на n => O(n*m), като разликата между average и worst case е константен фактор, ~ n*m/2 срещу n*m, но и в двата случая имаме O(n*m) асимптотична сложност.

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));
```

Отговор: Очевидно, програмата написана така ще крашва при неквадратна матрица, тъй като проверките за размер са разменени (IndexOutOfRangeException). Ако броят редове n е по-голям от броя колони m, това ще се случи в първия цикъл; иначе програмата ще го изпълни n пъти, след което ще рекурсира - общият брой рекурсивни извиквания ще е n (докато row не излезе извън матрицата) => $O(n^2)$.

По същата логика, ако оправим проверките, програмата ще изпълни n извиквания и цикълът ще е с m итерации => ще имаме O(n*m) операции, което е и логично, тъй като кодът представлява сумиране на матрица.

Впрочем, мисля че ще е полезно да се покаже как изглежда програмата след стандартно елминиране на рекурсията (и след размяна на проверките):

```
long CalcSum(int[,] matrix, int row)
{
    long sum = 0;

    while(true)
    {
        for (int col = 0; col < matrix.GetLength(1); col++)
            sum += matrix[row, col];

        // tail recursion replacement
        // base case
        if (!(row + 1 < matrix.GetLength(0)))
```

```
            break;

        // recursive case
        row += 1;
    }
    return sum;
}
```