ID:2100030408

Lab 2: Tasks on Looping, Arrays and Functions

1. Write a C# code to implement the Tasks on Looping Statements?

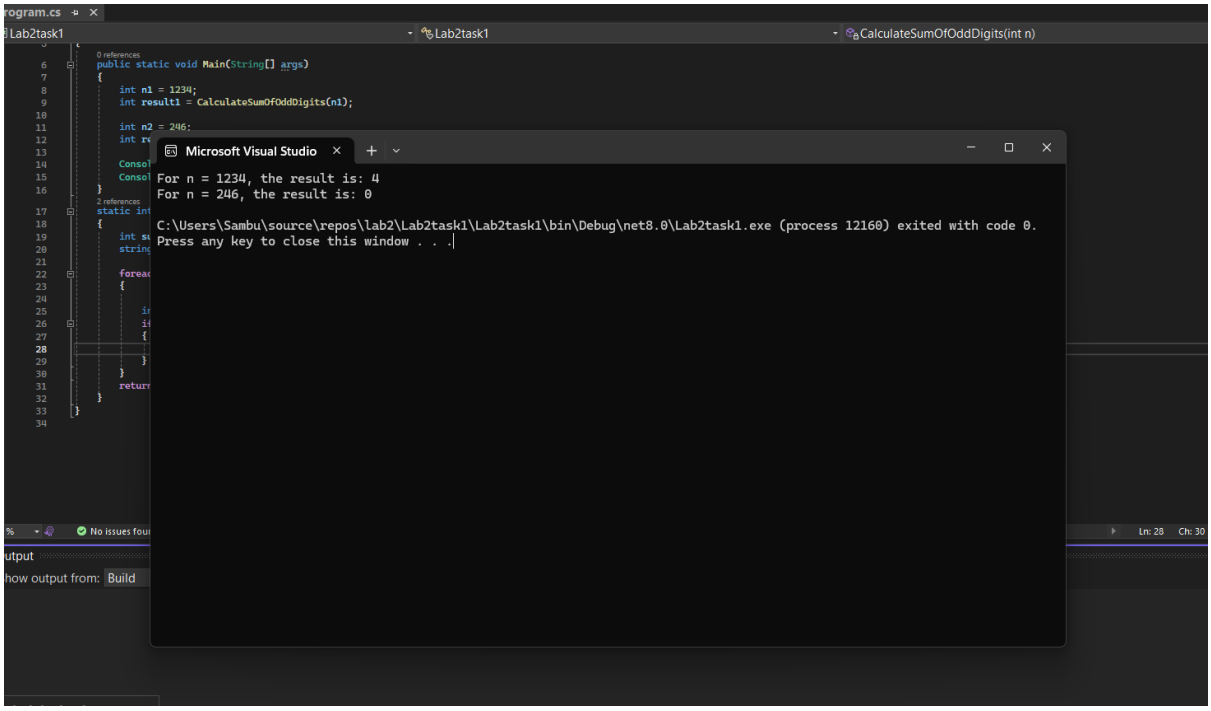
TASK1: For a positive integer *n* calculate the *result* value, which is equal to the sum of the odd numbers in *n*

Example

```
n = 1234    result = 4 (1 + 3)
```

```
n = 246     result = 0
```

answer1:



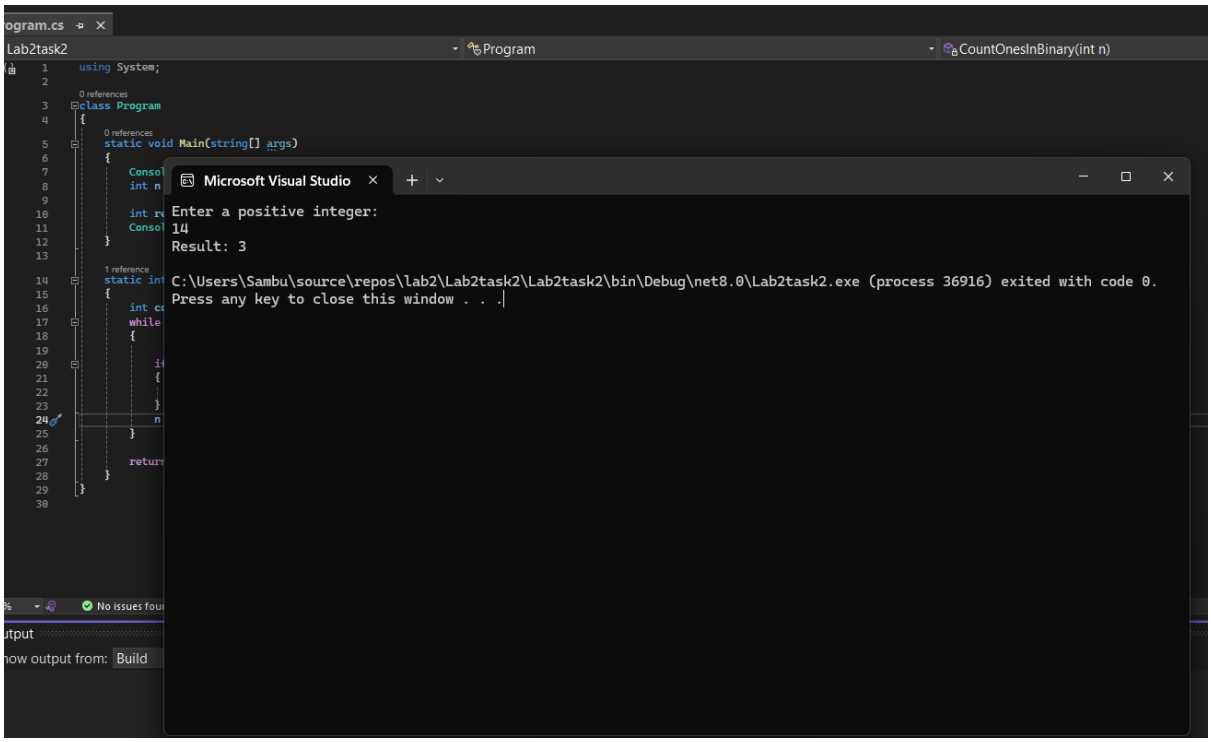
TASK2: For a positive integer *n* calculate the result value, which is equal to the sum of the "1" in the binary representation of *n*.

Example

```
n = 14(decimal) = 1110(binary)       result =  3
```

```
n = 128(decimal) = 1000 0000(binary) result  =  1
```

answer2:



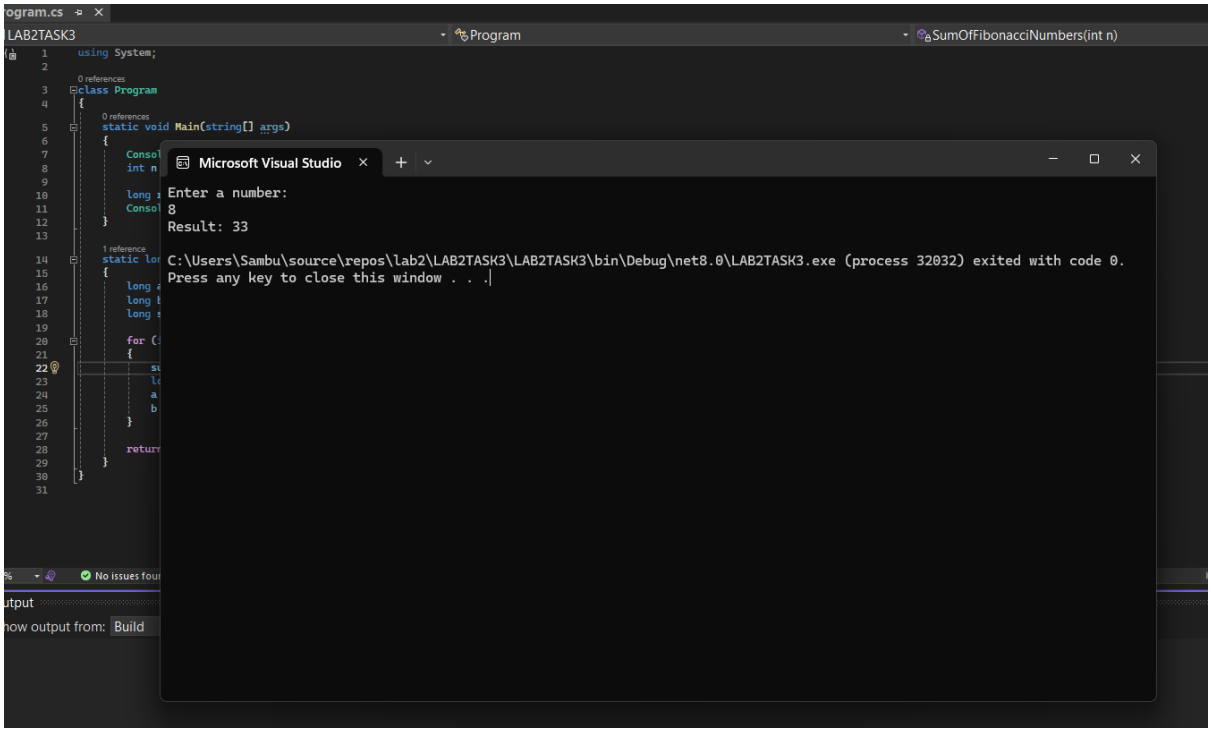
TASK3: For a positive integer *n*, calculate the result value equal to the sum of the first *n* Fibonacci numbers Note: Fibonacci numbers are a series of numbers in which each next number is equal to the sum of the two preceding ones: 0, 1, 1, 2, 3, 5, 8, 13... (F0=0, F1=F2=1, then F(n)=F(n-1)+F(n-2) for n>2)

Example

```
n = 8     result = 33
```

```
n = 11    result = 143
```

answer3:



1. Write a C# code to implement the Tasks on Arrays?

TASK 1: In a given array of integers ***nums*** swap values of the first and the last array elements, the second and the penultimate etc., if the two exchanged values are even

Example

```
{ 10 , 5, 3, 4}                          =>  {4, 5, 3, 10}

{100, 2, 3, 4, 5}                      => {100, 4, 3, 2, 5}

{100, 2, 3, 45, 33, 8, 4, 54} => {54, 4, 3, 45, 33, 8, 2, 100}
```

TASK 2: In a given array of integers ***nums*** calculate integer ***result*** value, that is equal to the distance between the first and the last entry of the maximum value in the array.

Example

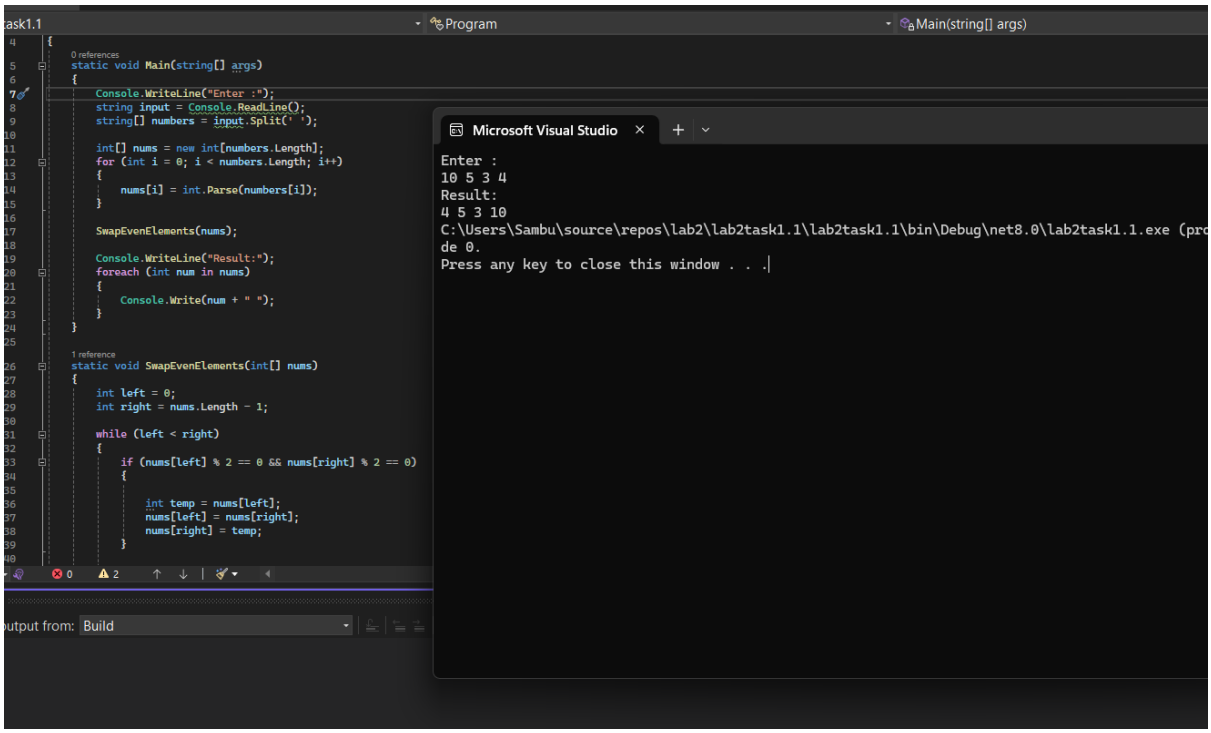
```
{4, 100!, 3, 4}                result = 0

{5, 50!, 50!, 4, 5}          result = 1

{5, 350!, 350, 4, 350!} result = 3

{10!, 10, 10, 10, 10!}   result = 4
```

TASK 3: In a predetermined two-dimensional integer array (square matrix) ***matrix*** insert 0 into elements to the left side of the main diagonal, and 1 into elements to the right side of the diagonal.

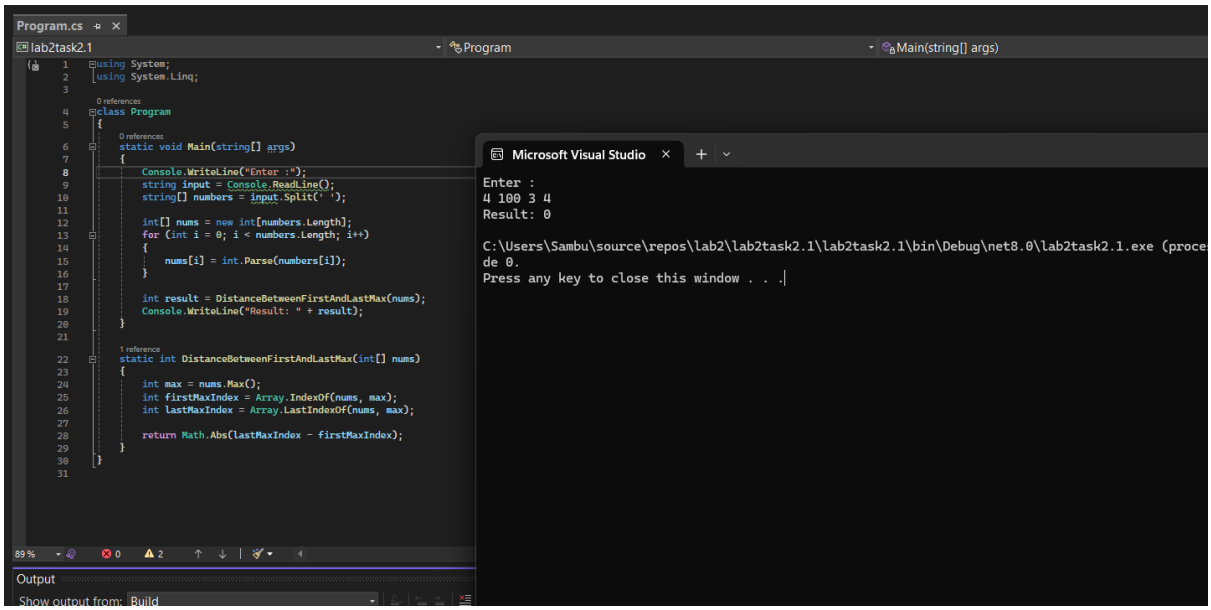
Example

```
{{2, 4, 3, 3},          {{2, 1, 1, 1},
 {5, 7, 8, 5},     =>    {0, 7, 1, 1},
 {2, 4, 3, 3},           {0, 0, 3, 1},
 {5, 7, 8, 5}}           {0, 0, 0, 5}}
```

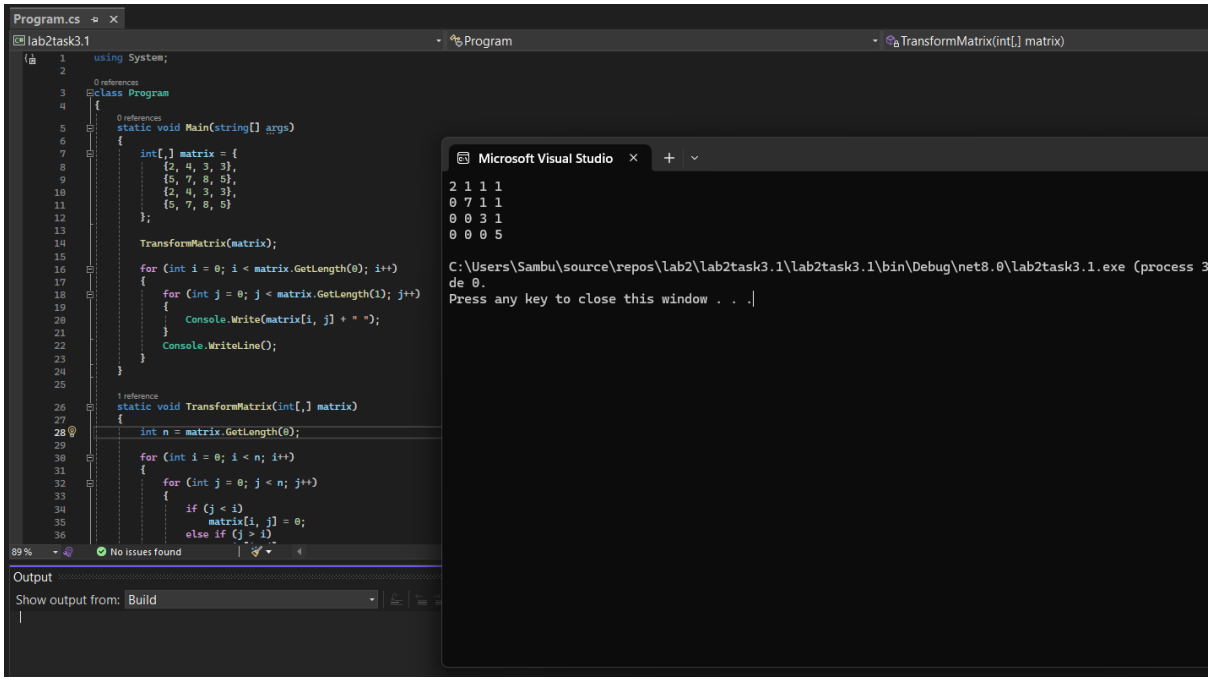
Answer1:



Answer2:



Answer3:



1. Write a C# code to implement the Tasks on Functions?

TASK 1: **Create function** *IsSorted***, determining whether a given** *array* **of integer values of arbitrary length is sorted in a given** *order* **(the order is set up by enum value** *SortOrder***). Array and sort order are passed by parameters. Function does not change the array**

TASK 2: Create function `Transform`, replacing the value of each element of an integer ***array*** with the sum of this element value and its index, only if the given ***array*** is sorted in the given ***order*** (the order is set up by enum value ***SortOrder***). Array and sort order are passed by parameters. To check, if the array is sorted, the function ***IsSorted*** from the Task 1 is called.

Example

```
For {5, 17, 24, 88, 33, 2} and "ascending" sort order values in the array do not change;

For {15, 10, 3} and "ascending" sort order values in the array do not change;

For {15, 10, 3} and "descending" sort order the values in the array change to {15, 11, 5}
```

TASK 3: Create function ***MultArithmeticElements***, which determines the multiplication of a given number of first ***n*** elements of arithmetic progression of real numbers with a given initial element of progression ***a(1)*** and progression step ***t***. a(n) is calculated by the formula a(n+1) = a(n) + t.

Example

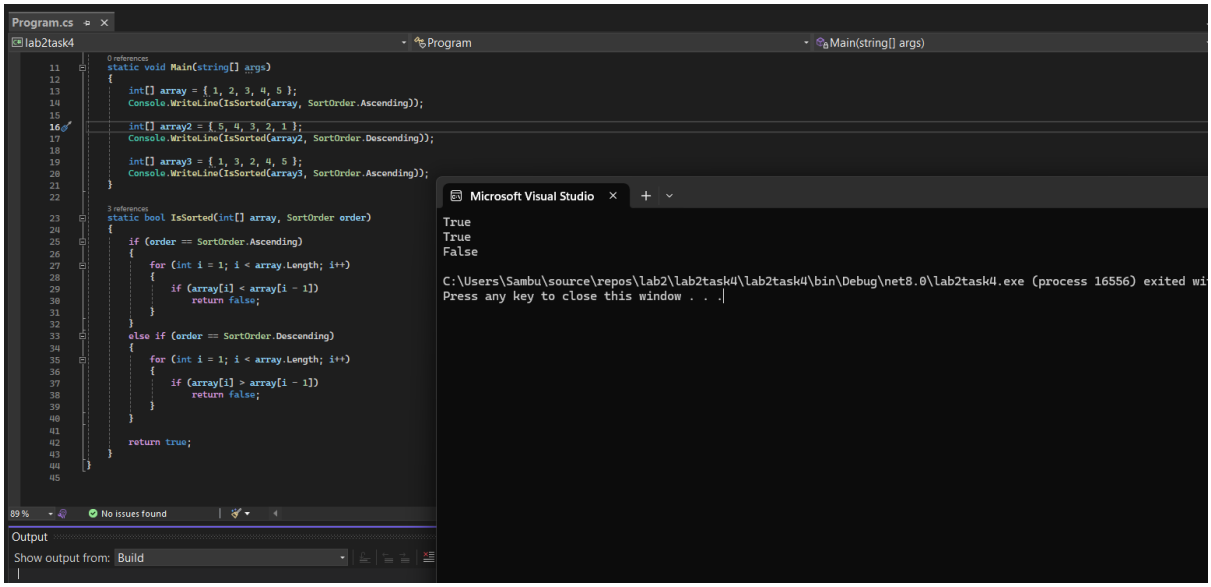
```
For a(1) = 5, t = 3, n = 4  multiplication equals to 5*8*11*14 = 6160
```

TASK 4: Create function ***SumGeometricElements***, determining the sum of the first elements of a decreasing geometric progression of real numbers with a given initial element of a progression ***a(1)***) and a given progression step ***t***, while the last element must be greater than a given ***alim***. an is calculated by the formula a(n+1) = a(n) * t, 0<t<1.

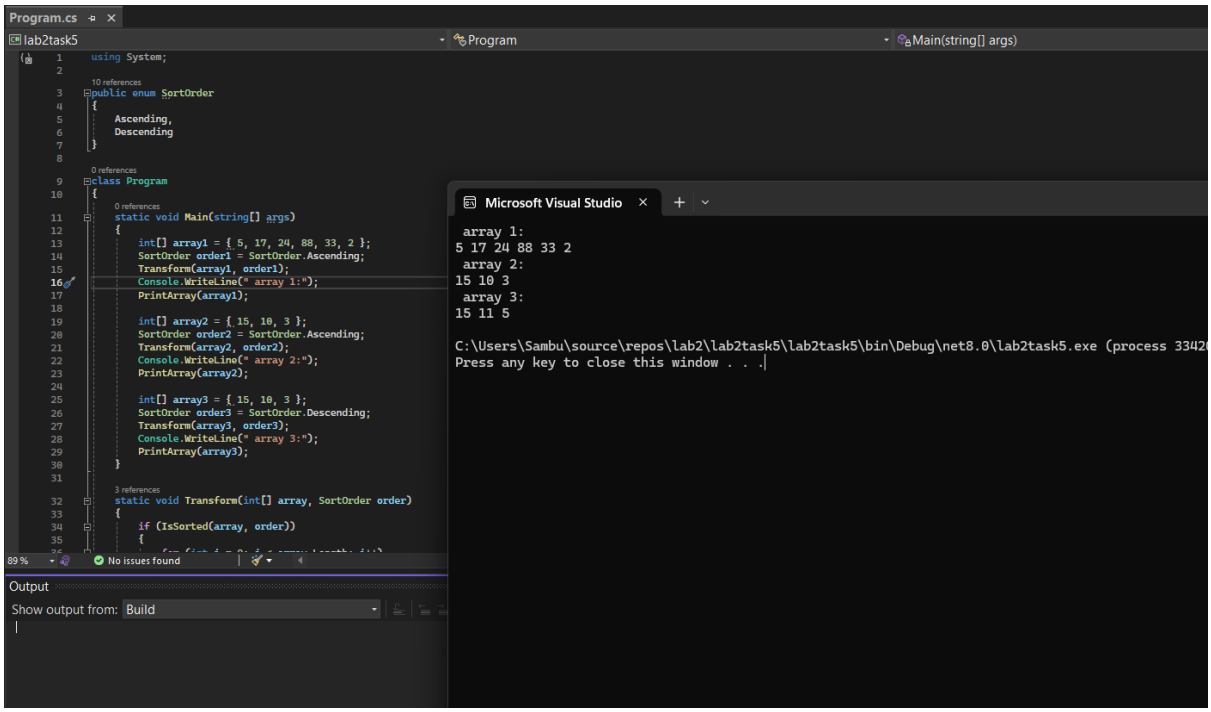
Example

```
For a progression, where a(1) = 100, and t = 0.5, the sum of the first elements, grater than alim = 20, equals to 100+50+25 = 175
```

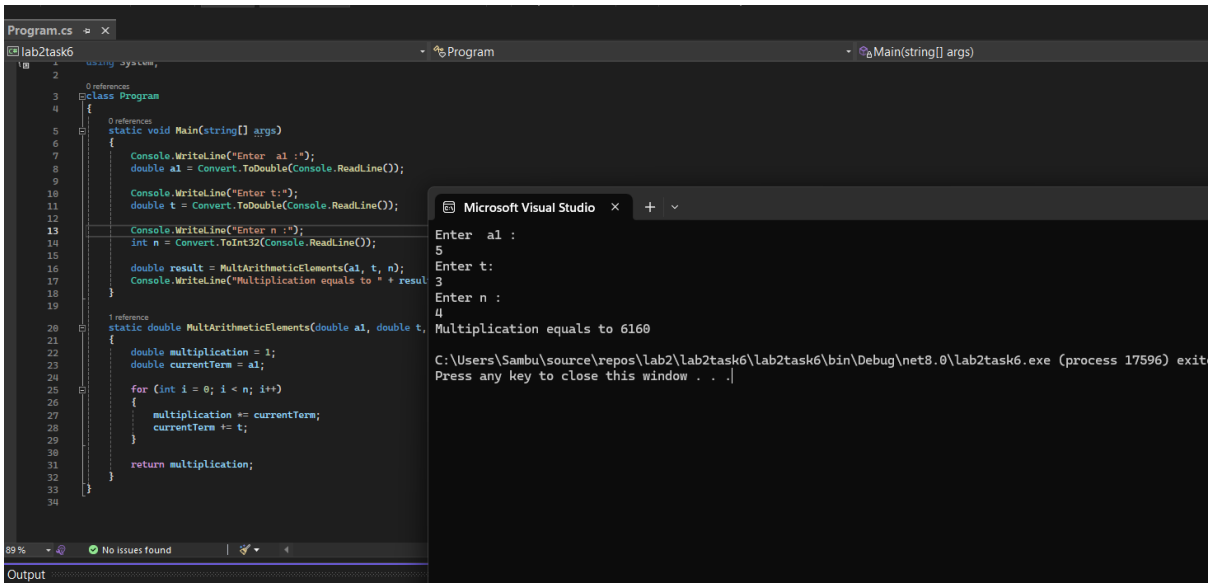
Answer1:



Answer2:



Answer3:



Answer4: