Exam Preparation

Test your tasks in the Judge system: https://judge.softuni.org/Contests/4454/Exam-Preparation-I

1. Sum Factorial Even Digits

Write a program that:

- Reads an integer number from the console
- Calculate sum of the factorials only on even digits of the given number
- Print the calculated sum

Example

Input	Output	Comments

4532	26	`First digit is 4, it is even so we calculate factorial: 4! = 4 * 3 * 2 * 1 = 24`
		
		`Second digit is 5, it is odd so we skip it.`
		
		`Third digit is 3, it is odd so we skip it.`
		
		`Forth digit is 2, it is even so we calculate factorial: 2! = 2 * 1 = 2`
		
		`Sum of factorials: 24 + 2 = 26`

468	41064	`First digit is 4, it is even so we calculate factorial: 4! = 4 * 3 * 2 * 1 = 24`
		
		`Second digit is 6, it is even so we calculate factorial:`
		`6! = 6 * 5 * 4* 3 * 2 * 1 = 720`
		
		`Third digit is 8, it is even so we calculate factorial:`
		`8! = 8 * 7 * 6 * 5 * 4* 3 * 2 * 1 = 40320`
		
		`Sum of factorials: 24 + 720 + 40320 = 41064`

2. Middle Elements

Write a program that:

- Reads an array of integer numbers from the console, separated by single space
- Array length will always be even number.

- Calculate the average value of the elements in the middle of the array
- Print the result formatted to the second digit

Example

Input	Output	Comments

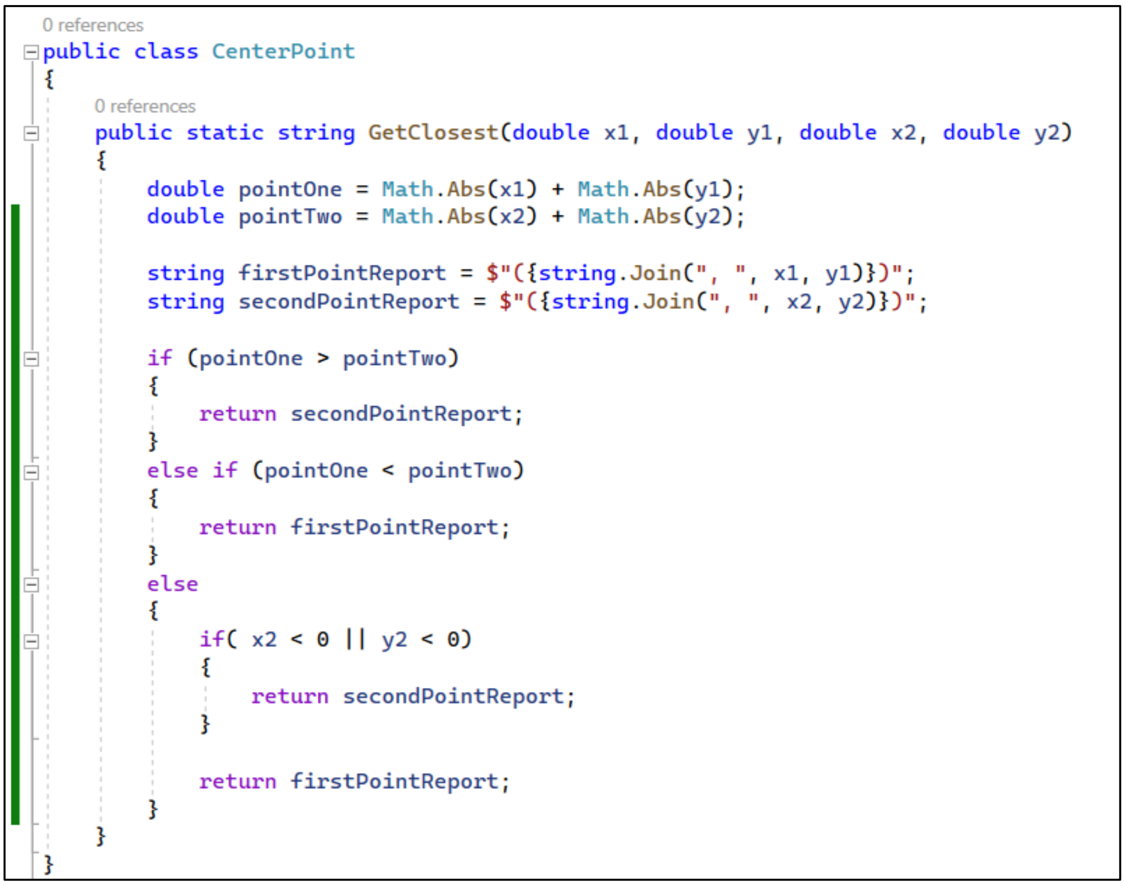
`3 4 6 7 8 9`	`6.50`	`Middle elements are: 6 and 7`
		`Average value: (6 + 7) / 2 = 13 / 2 = 6.50`

`12 34 98 42 65 12`	`70.00`	`Middle elements are: 98 and 42`
		`Average value: (98 + 42) / 2 = 140 / 2 = 70`

3. Unit Test Method: Center Point

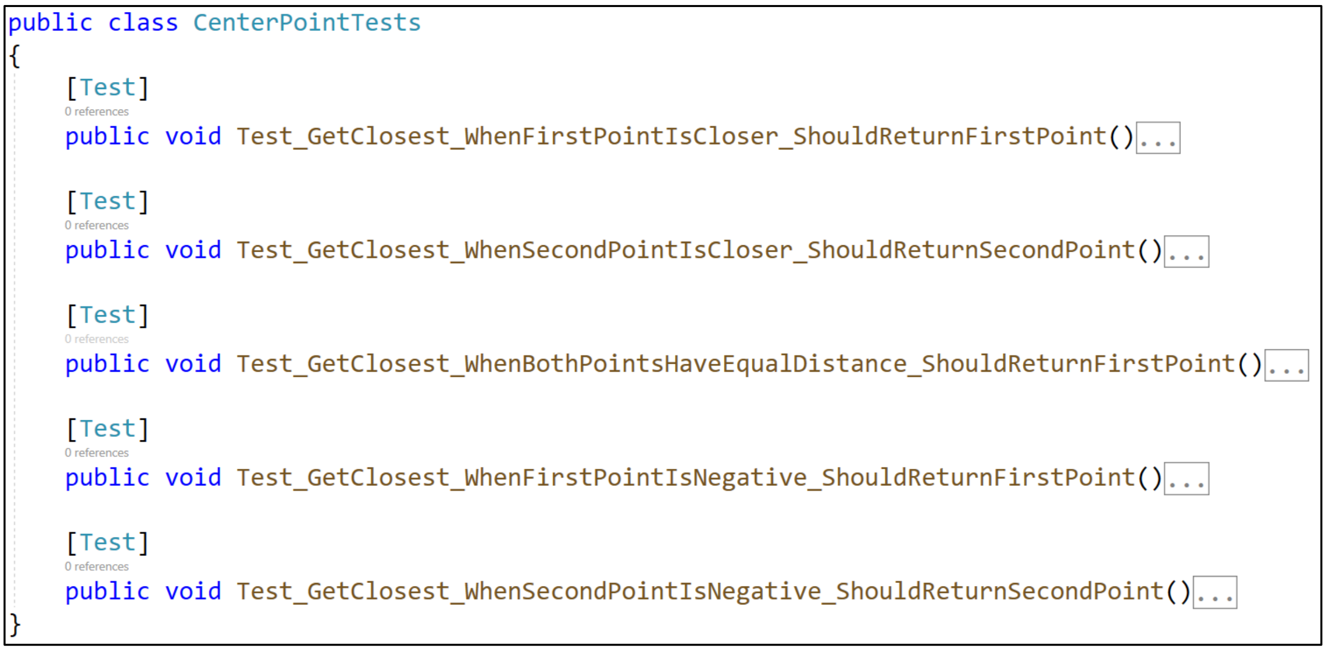
Test a given method which takes in **coordinates of 2 points** and determines which point is **closer to the center (0,0)**.

The method is found in the **`CenterPoint.cs`** file:



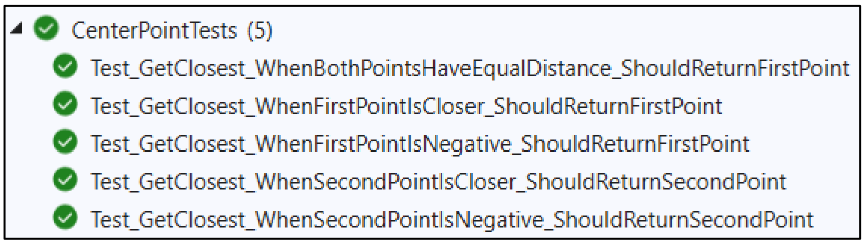
```
0 references
public class CenterPoint
{
    0 references
    public static string GetClosest(double x1, double y1, double x2, double y2)
    {
        double pointOne = Math.Abs(x1) + Math.Abs(y1);
        double pointTwo = Math.Abs(x2) + Math.Abs(y2);

        string firstPointReport = $"({string.Join(", ", x1, y1)})";
        string secondPointReport = $"({string.Join(", ", x2, y2)})";

        if (pointOne > pointTwo)
        {
            return secondPointReport;
        }
        else if (pointOne < pointTwo)
        {
            return firstPointReport;
        }
        else
        {
            if( x2 < 0 || y2 < 0)
            {
                return secondPointReport;
            }

            return firstPointReport;
        }
    }
}
```

You are given a **test file `CenterPointTests.cs`** containing **5 empty tests**. Implement all the unit tests:



```
public class CenterPointTests
{
    [Test]
    0 references
    public void Test_GetClosest_WhenFirstPointIsCloser_ShouldReturnFirstPoint()...

    [Test]
    0 references
    public void Test_GetClosest_WhenSecondPointIsCloser_ShouldReturnSecondPoint()...

    [Test]
    0 references
    public void Test_GetClosest_WhenBothPointsHaveEqualDistance_ShouldReturnFirstPoint()...

    [Test]
    0 references
    public void Test_GetClosest_WhenFirstPointIsNegative_ShouldReturnFirstPoint()...

    [Test]
    0 references
    public void Test_GetClosest_WhenSecondPointIsNegative_ShouldReturnSecondPoint()...
}
```

When you are ready make sure your **tests run:**

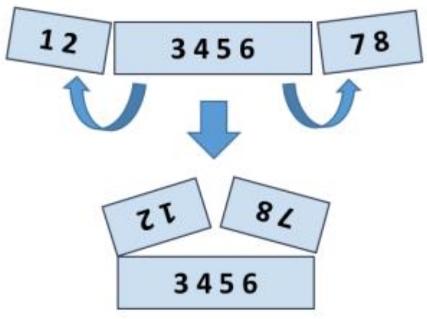


```
◢ ✓ CenterPointTests (5)
    ✓ Test_GetClosest_WhenBothPointsHaveEqualDistance_ShouldReturnFirstPoint
    ✓ Test_GetClosest_WhenFirstPointIsCloser_ShouldReturnFirstPoint
    ✓ Test_GetClosest_WhenFirstPointIsNegative_ShouldReturnFirstPoint
    ✓ Test_GetClosest_WhenSecondPointIsCloser_ShouldReturnSecondPoint
    ✓ Test_GetClosest_WhenSecondPointIsNegative_ShouldReturnSecondPoint
```

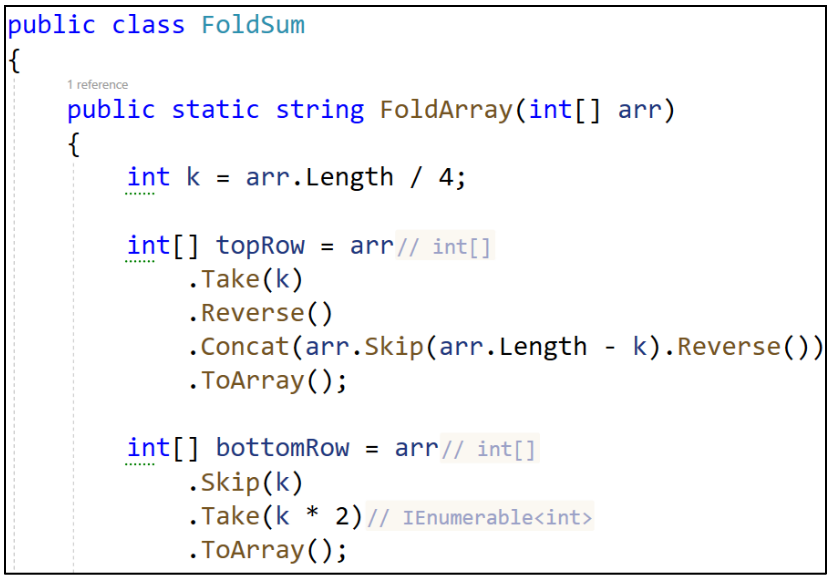
IMPORTANT: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.

4. Unit Test Array: Fold Array

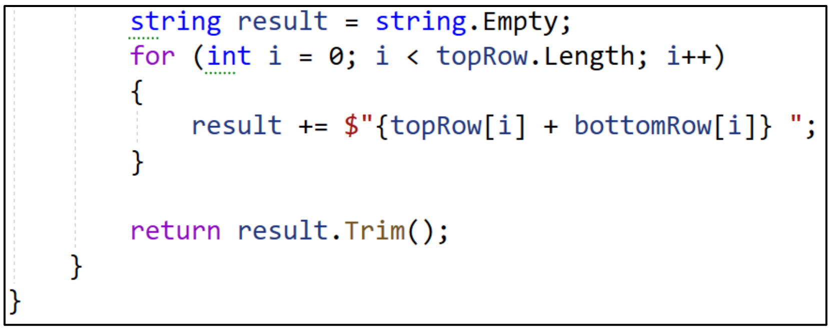
Test a given method which takes in **an integer array** of 4*k integers, then folds it like shown below, and returns the sum of the **upper** and **lower** two rows (each holding 2*k integers):



The method is found in the **`FoldSum.cs`** file:

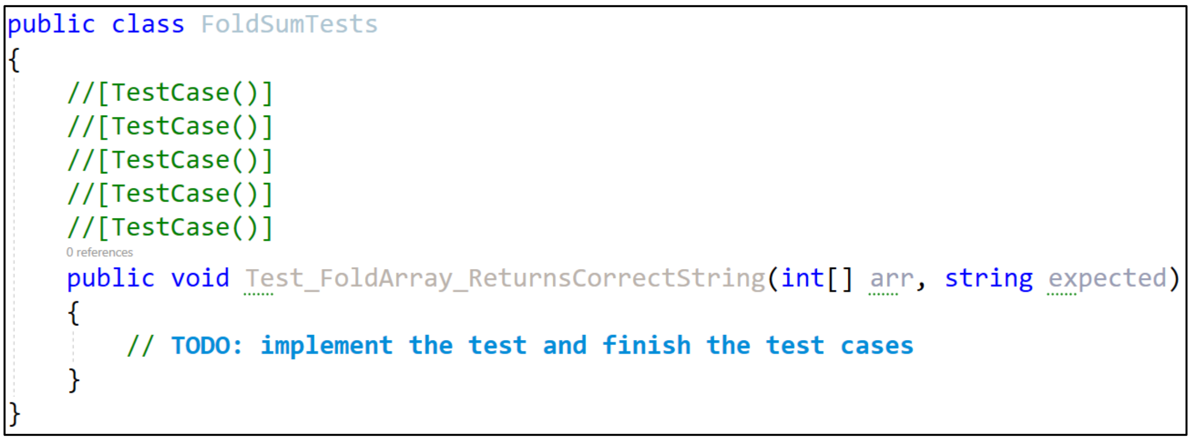


```
public class FoldSum
{
    1 reference
    public static string FoldArray(int[] arr)
    {
        int k = arr.Length / 4;

        int[] topRow = arr// int[]
            .Take(k)
            .Reverse()
            .Concat(arr.Skip(arr.Length - k).Reverse())
            .ToArray();

        int[] bottomRow = arr// int[]
            .Skip(k)
            .Take(k * 2)// IEnumerable<int>
            .ToArray();
```



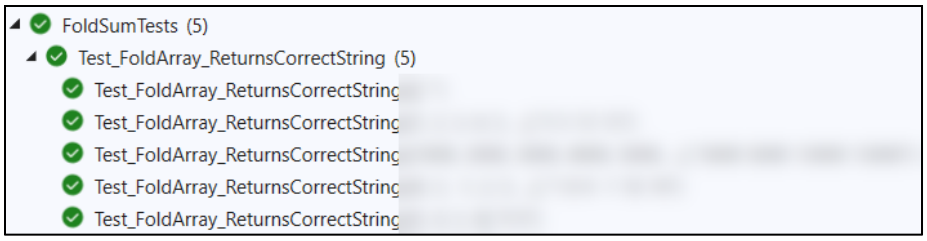
```
        string result = string.Empty;
        for (int i = 0; i < topRow.Length; i++)
        {
            result += $"{topRow[i] + bottomRow[i]} ";
        }

        return result.Trim();
    }
}
```

You are given a **test file `FoldSumTests.cs`** containing **5 empty test cases**. Implement all the cases:



```
public class FoldSumTests
{
    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    //[TestCase()]
    0 references
    public void Test_FoldArray_ReturnsCorrectString(int[] arr, string expected)
    {
        // TODO: implement the test and finish the test cases
    }
}
```

When you are ready make sure your **tests run:**



```
◢ ✓ FoldSumTests (5)
  ◢ ✓ Test_FoldArray_ReturnsCorrectString (5)
      ✓ Test_FoldArray_ReturnsCorrectString
      ✓ Test_FoldArray_ReturnsCorrectString
      ✓ Test_FoldArray_ReturnsCorrectString
      ✓ Test_FoldArray_ReturnsCorrectString
      ✓ Test_FoldArray_ReturnsCorrectString
```

<u>IMPORTANT</u>: DO NOT REMOVE OR CHANGE ANY NAMESPACES AND USINGS.