TrianglesAndRectanglesOhMy

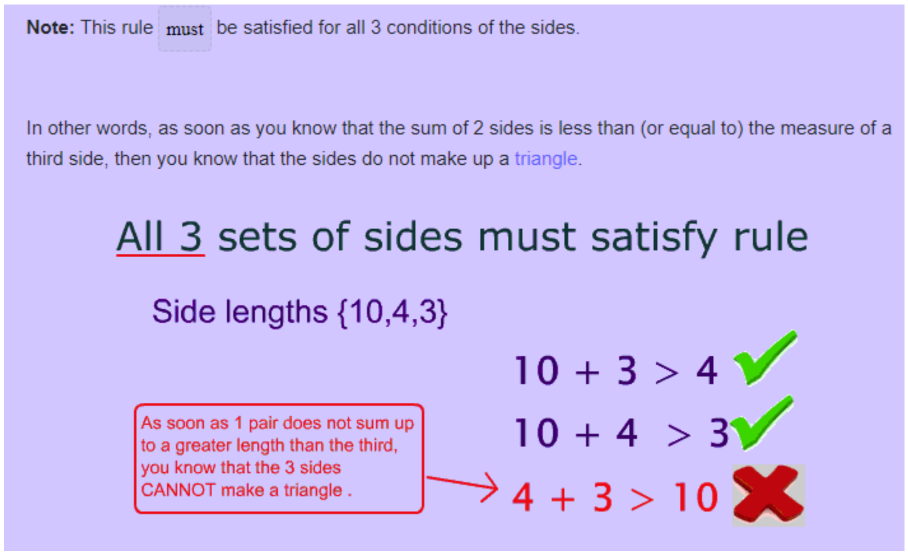
In this problem you will implement five (5) methods dealing with the properties of triangles and rectangles.

The first method to implement is `getNumPossibleTriangle`. Given three arrays of `int`s, how many different ways is it possible to pick 1 number from each array, so that the three numbers could be the lengths of the three sides of a triangle.

According to Wikipedia:

> In mathematics, the triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than the length of the remaining side.

The following clarification is from mathhouse.com:



Note: This rule must be satisfied for all 3 conditions of the sides.

In other words, as soon as you know that the sum of 2 sides is less than (or equal to) the measure of a third side, then you know that the sides do not make up a triangle.

All 3 sets of sides must satisfy rule
Side lengths {10,4,3}
10 + 3 > 4 ✓
10 + 4 > 3 ✓
4 + 3 > 10 ✗
As soon as 1 pair does not sum up to a greater length than the third, you know that the 3 sides CANNOT make a triangle .

For example, given `int[]`:

```
    side1 = {1, 2, 3}     side2 = {3, 4, 5}     side3 = {5, 6, 7}
```

Ten different Triangles can be made by choosing one value from each array. The 10 possible triangles would have sides of length:

1. {1, 5, 5}
2. {2, 4, 5}
3. {2, 5, 5}
4. {2, 5, 6}
5. {3, 3, 5}
6. {3, 4, 5}
7. {3, 4, 6}
8. {3, 5, 5}
9. {3, 5, 6}
10. {3, 5, 7}

The following code shows the results of the `getNumPossibleTriangle` method.

The following code	Returns

`int[] side1 = {1, 2, 3};`
`int[] side2 = {3, 4, 5};`
`int[] side3 = {5, 6, 7};`	

`TrianglesAndRectanglesOhMy.getNumPossibleTriangle(side1, side2, side3);`	10

Another example: given `int[]`:

    side1a = {2, 4, 4}   side2a = {1, 3, 1}   side3a = {5, 2}

Seven different Triangles can be made by choosing one value from each array. The 7 possible triangles would have sides of length:

1. {2, 1, 2}
2. {2, 3, 2}
3. {2, 1, 2} - since side2 contains the value 1 twice, this triangle will get counted twice
4. {4, 3, 5}
5. {4, 3, 2}
6. {4, 3, 2}
7. {4, 1, 2}

And, the following code shows the results of the `getNumPossibleTriangle` method.

The following code	Returns

`int[] side1a = {2, 4, 4};`
`int[] side2a = {1, 3, 1};`
`int[] side3a = {5, 2};`	

`TrianglesAndRectanglesOhMy.getNumPossibleTriangle(side1a, side2a, side3a));`	7

The second and third methods to be implemented are the `possibleMinThirdSideOfTriangle` and `possibleMaxThirdSideOfTriangle`. Both methods have three parameters. The first two parameters are two sides of a triangle and the third parameter is an array of `int`s. The `possibleMinThirdSideOfTriangle` returns the smallest value from the `int` array that can be the third side of the triangle, and `possibleMaxThirdSideOfTriangle` returns the largest value from the `int` array that can be the third side of the triangle.

For example, given the two sides 7 and 19 and the array containing the values: 13, 6, 9, 37, 11, 5, 2, 23, 17. 13 is the smallest value in the array that forms a triangle along with sides 7 and 19. And 23 is the largest value in the array that forms a triangle with sides 7 and 19.

And, the following code shows the results of the `possibleMinThirdSideOfTriangle` method.

The following code	Returns
`int[] possibleSides = {13, 6, 9, 37, 11, 5, 2, 23, 17};`	
`TrianglesAndRectanglesOhMy.possibleMinThirdSideOfTriangle(7, 19, possibleSides);`	13

And, the following code shows the results of the `possibleMaxThirdSideOfTriangle` method.

The following code	Returns
`int[] possibleSides = {13, 6, 9, 37, 11, 5, 2, 23, 17};`	
`TrianglesAndRectanglesOhMy.possibleMaxThirdSideOfTriangle(7, 19, possibleSides)`	23

The final two methods in this problem concern properties of rectangles. The first method is the `getNumPossibleRectangleWithArea`. This method has three parameters; the first two parameters are arrays of `int` containing possible lengths of the sides of a rectangle. The third parameter is an `int` representing the desired area of the rectangle. This method returns the number of rectangles that can be formed by choosing one value from each of the two `int` array parameters such that the area of this rectangle is equal to the third parameter (the desired area of the rectangle).

For example, given `int[]` and a desired area of 24:

```
     sideA = {2, 4, 8, 11}        sideB = {1, 3, 6, 7, 8}
```

Two different rectangles can be made by choosing one value from each array.

1. `4 from first array and 6 from second array`
2. `8 from first array and 3 from second array`

The following code shows the results of the `getNumPossibleRectangleWithArea` method.

The following code	Returns

`int[] sideA = {2, 4, 8, 11};`
`int[] sideB = {1, 3, 6, 7, 8};`	

`TrianglesAndRectanglesOhMy.getNumPossibleRectangleWithArea(sideA, sideB, 24)`	2

And in another example, given `int[]`and a desired area of 24:

```
     sideA1 = {2, 4, 8, 2, 11}        sideB1 = {12, 6, 3, 7, 8}
```

Four different rectangles can be made by choosing one value from each array.

1. `2 from first array and 12 from second array`
2. `4 from first array and 6 from second array`
3. `8 from first array and 3 from second array`
4. `2 from first array and 12 from second array`

And, the following code shows the results of the `getNumPossibleRectangleWithArea` method.

The following code	Returns

`int[] sideA1 = {2, 4, 8, 2, 11};`
`int[] sideB1 = {12, 6, 3, 7, 8};`	

`TrianglesAndRectanglesOhMy.getNumPossibleRectangleWithArea(sideA1, sideB1, 24)`	4

The final method in this problem is the `rectangleWithAreaAndMinPerimeter`. This method has the same three parameters as `getNumPossibleRectangleWithArea`: the first two parameters are arrays of `int` containing possible lengths of the sides of a rectangle. The third parameter is an `int` representing the desired area of the rectangle. This method returns the an `int[]` containing two values. The two values are the sides of rectangles that can be formed by choosing one value from each of the two `int` array parameter such that the area of this rectangle is equal to the third parameter (the desired area of the rectangle) with the minimum Perimeter. The value from the first parameter is stored in the first element (index == 0) and the value from the second parameter is second element (index == 1) of the array being returned.

Example continues on next page

For example, given `int[]` and a desired area of 24:

```
      sideA1 = {2, 4, 8, 2, 11}         sideB1 = {12, 6, 3, 7, 8}
```

Two different rectangles can be made by choosing one value from each array.

1. `4 from first array and 6 from second array`
2. `8 from first array and 3 from second array`

The perimeter of the first triangle is 2*4+2*6 = 20 and the perimeter of the second triangle is 2*8+2*3 = 22 so an array of length 2 with a 4 in index 0 and 6 in index 1 is returned.

You may assume `getNumPossibleRectangleWithArea(sideA1, sideB1, area) > 0`

The following code shows the results of the `rectangleWithAreaAndMinPerimeter` method.

The following code	Returns

`int[] sideA1 = {2, 4, 8, 2, 11};`
`int[] sideB1 = {12, 6, 3, 7, 8};`	

`int[] stu =`
`       TrianglesAndRectanglesOhMy.rectangleWithAreaAndMinPerimeter(`
`             sideA1, sideB1, 24);`	

`stu[0]`	4
`stu[1];`	6