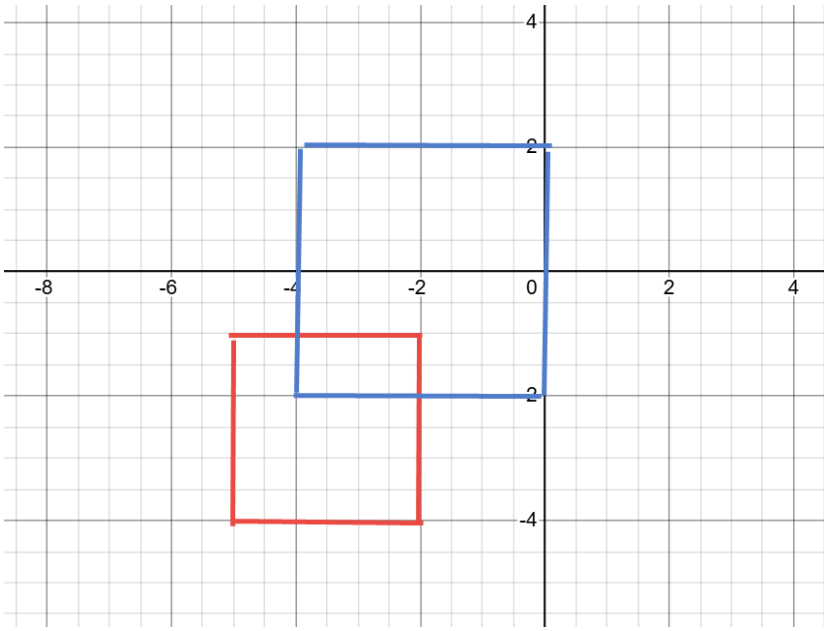
I have devised a test cases below.
Please note, my code has been devised by simply creating test scenarios. And building all the logic outwards again.. But this time, I had visual aids to identify the constraints.

TEST CASE 1:




```
//TEST CASE 1
System.out.println("The overlapping area is: " + overlappingArea(new int[]{-5, -4},  new int[]{-2, -1}, new int[]{-4, -2}, new int[]{0, 2}));
//                              //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

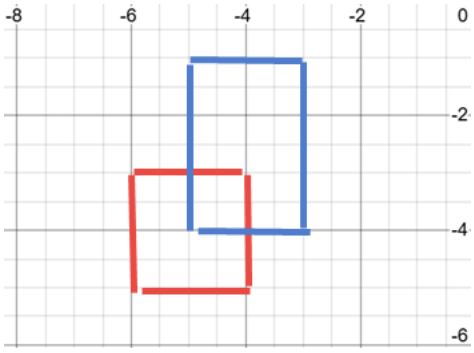
```
WIDTH: 2
HEIGHT: 1
The overlapping area is: 2
```

TEST CASE 2:
I have simply flipped the dimensions for the two rectangles

```
66        //TEST CASE 2 - same as above, but it will flip coordinates of the two rectangles
67        System.out.println("The overlapping area is: " + overlappingArea(new int[]{-4, -2}, new int[]{0, 2},new int[]{-5, -4}, new int[]{-2, -1}));
68        //                              //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
WIDTH1: 2
HEIGHT2: 1
The overlapping area is: 2
```

TEST CASE 3: Now trying two rectangles both in the negative axis




```
//TEST CASE 3
System.out.println("The overlapping area is: " + overlappingArea(new int[]{-6, -5},  new int[]{-4, -3}, new int[]{-5, -4}, new int[]{-3, -1}));
//                              //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
WIDTH: 1
HEIGHT: 1
The overlapping area is: 1
```

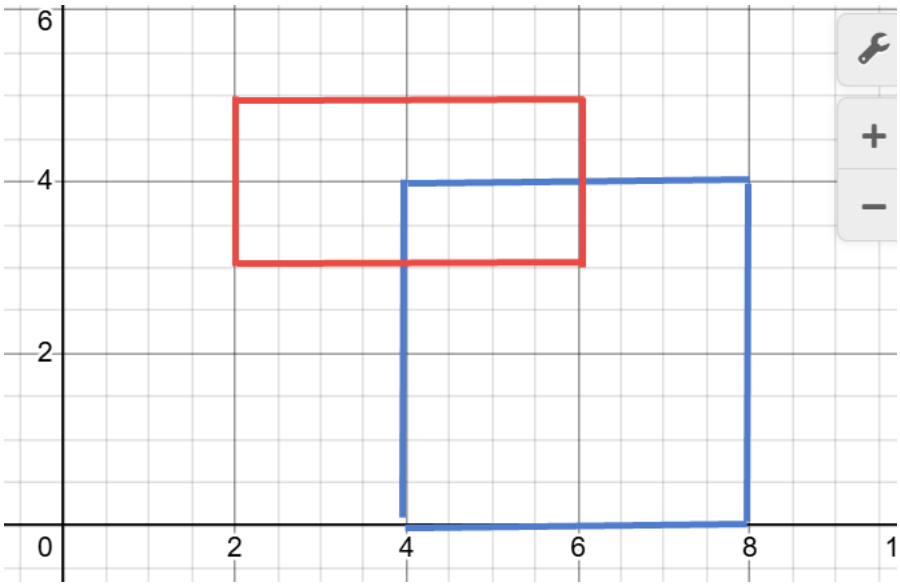
Similarly, I have flipped the rectangles

TEST CASE 4:

```
  //TEST CASE 4
 System.out.println("The overlapping area is: " + overlappingArea(new int[]{-5, -4}, new int[]{-3, -1},new int[]{-6, -5},  new int[]{-4, -3}));
 //                              //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
WIDTH1: 1
HEIGHT2: 1
The overlapping area is: 1
```

TEST CASE 5: Perhaps this should have been the first test FAIL



```
69          //TEST CASE 5
70          System.out.println("The overlapping area is: " + overlappingArea(new int[]{2, 3}, new int[]{6, 5},new int[]{4, 0},  new int[]{8, 4}));
71          //                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
WIDTH1: -6
HEIGHT2: -1
The overlapping area is: 6
```

We were expecting width = 2 height = 1

I will quickly investigate:

I have inputted a substantial amount of code to deal with situation of all dimensions being in positive axis.
It is still quite difficult to imagine how this logic differs to other test cases. But we can see rectangles in relation to each other, one is in top left and other in bottom right.

I have introduced lots more code, it will be difficult to add relevant comments.. But it will be available in my repository.

TEST CASE 5a:

```
8
WIDTH6: -2
HEIGHT6: -1
The overlapping area is: 2
```

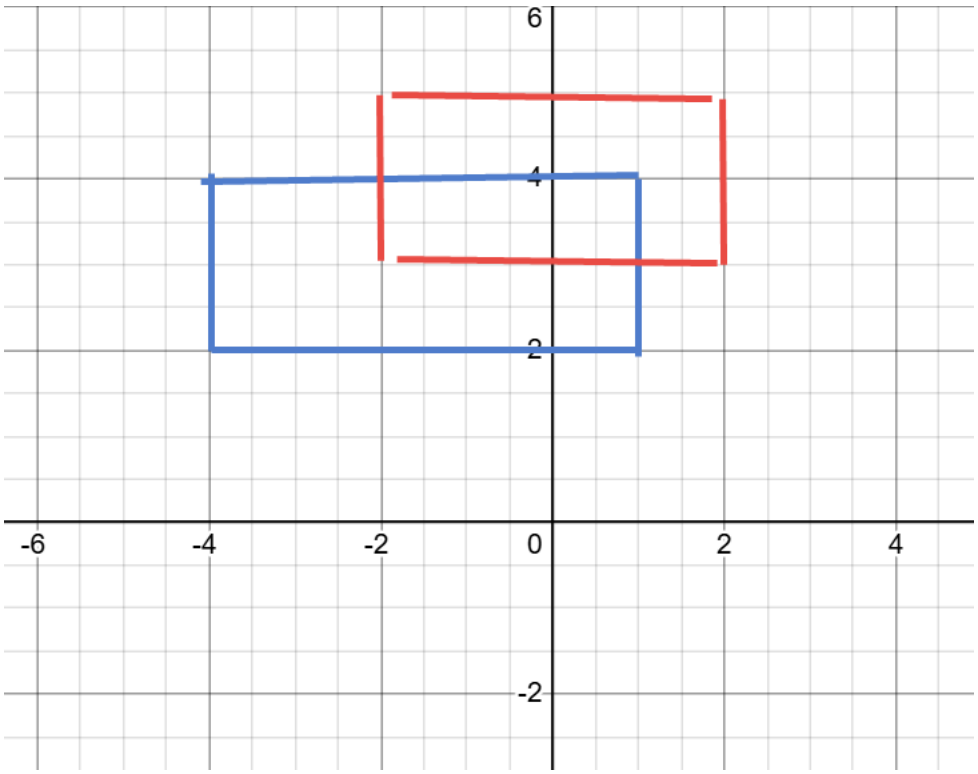
I have now flipped the rectangles as per usual

```
101         //TEST CASE 6
102         System.out.println("The overlapping area is: " + overlappingArea(new int[]{4, 0},  new int[]{8, 4},new int[]{2, 3}, new int[]{6, 5}));
103         //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

TEST CASE 6:

```
WIDTH3: 2
HEIGHT3: -1
The overlapping area is: 2
```

TEST CASE 7:




```
138
139         //TEST CASE 7
140         System.out.println("The overlapping area is: " + overlappingArea(new int[]{-4, 2},  new int[]{1, 4},new int[]{-2, 3}, new int[]{2, 5}));
141         //                                                  //rect1bottomLeft     //rect1TopRight    //rect2BottomLeft    //rect2TopRight
```

```
WIDTH9: 3
HEIGHT9: -1
The overlapping area is: 3
```

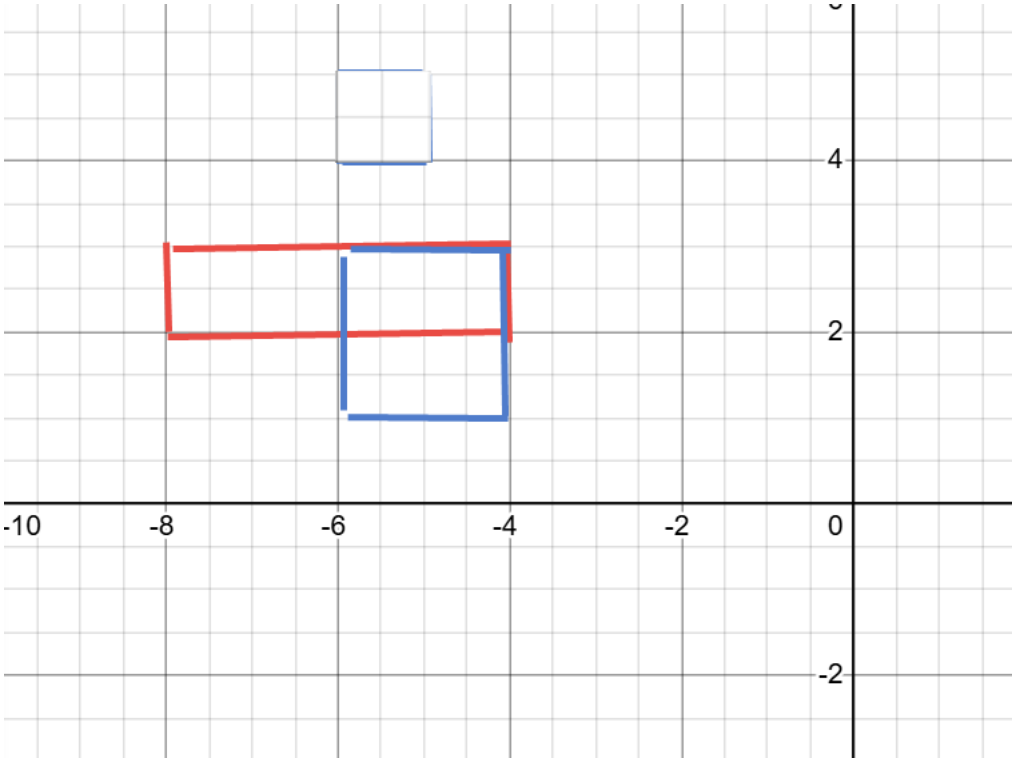
As usual, I have flipped the rectangle dimensions around…

```
143         //TEST CASE 8
144         System.out.println("The overlapping area is: " + overlappingArea(new int[]{-2, 3}, new int[]{2, 5},new int[]{-4, 2},  new int[]{1, 4}));
145         //                                                  //rect1bottomLeft     //rect1TopRight    //rect2BottomLeft    //rect2TopRight
```

TEST CASE 8:

```
-2
1
3
The overlapping area is: 3
```

TEST CASE 9:



I have reached this test case and I can see that I have not explored this at all...
Currently when executing the code we are in this section:

```
26              }
27
28              else
29              {
30                   //used for test case 8
31                  if (rect2BottomLeft[0]<0 || rect1TopRight[0]<0)
32                  {
33                      //System.out.println(rect2BottomLeft[0]);
34                      //System.out.println(rect1TopRight[0]);
35
36                      System.out.println("rect1 bottom left: " + rect1BottomLeft[0]);
37                       System.out.println("rect2 top right: " + rect2TopRight[0]);
38
39
40                      width=Math.abs(rect1BottomLeft[0])+Math.abs(rect2TopRight[0]);
41                      System.out.println("width: " + width);
42                      height = Math.abs(0-rect1BottomLeft[1]) - Math.abs(0-rect2TopRight[1]);
43                  }
44
```

We can see that perform Math.abs(4) + Math.abs(8) will give 12.

```
rect1 bottom left: -8
rect2 top right: -4
12
The overlapping area is: 12
```

But we can see infact, we require width to be different if the shapes overlap exactly at

top right
We expect every scenario tested so far to be susceptible to this.
For now, I will just address this scenario.
But perhaps, we require a method to perform a level of validation… This will save repeat code.

For now, I have just resolved the issue with following code:

```
39                  //used for test case 9
40                  if (rect1TopRight[0]==rect2TopRight[0])
41                  {
42                      System.out.println(rect1BottomLeft[0]);
43
44                      width=Math.abs(rect2BottomLeft[0])-Math.abs(rect1TopRight[0]);
45                      System.out.println("width10: " + width);
46                      height = Math.abs(0-rect1BottomLeft[1]) - Math.abs(0-rect2TopRight[1]);
47                       System.out.println("height10: " + width);
```

TEST CASE 9a:

```
163     //TEST CASE 9
164     System.out.println("The overlapping area is: " + overlappingArea(new int[]{-8, 2}, new int[]{-4, 3},new int[]{-6, 1},  new int[]{-4, 3}));
165     //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
rect1 bottom left: -8
rect2 top right: -4
-8
width10: 2
height10: 2
The overlapping area is: 2
```

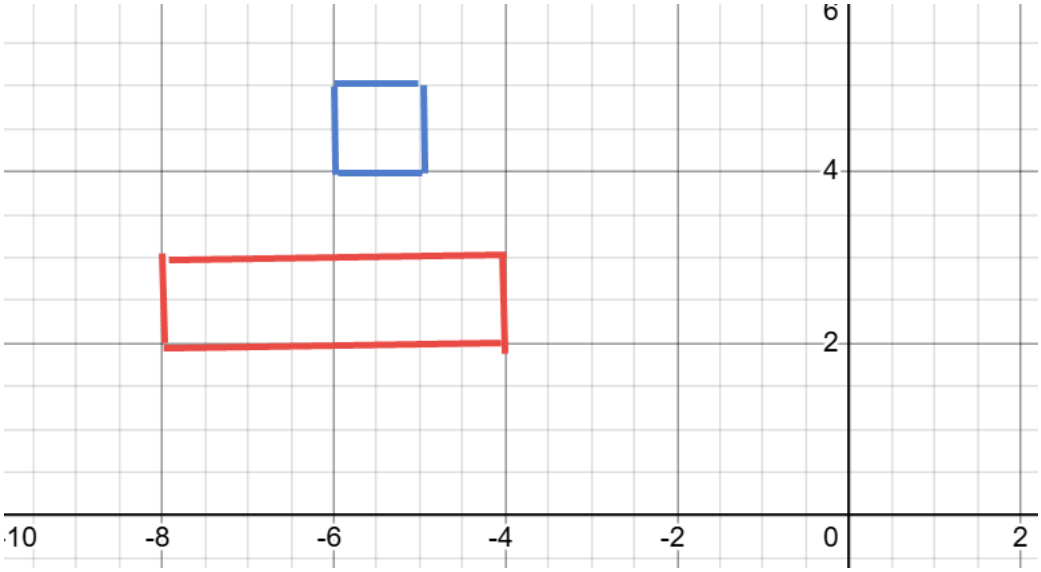
I will now flip the scenario as per usual:

TEST CASE 10:

```
181     //TEST CASE 10
182     System.out.println("The overlapping area is: " + overlappingArea(new int[]{-6, 1},  new int[]{-4, 3},new int[]{-8, 2}, new int[]{-4, 3}));
183     //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
SSSS
-6
width11: 2
height11: 2
The overlapping area is: 2
```

TEST CASE 11: This is to see if it identifies no overlaps



Solution.java

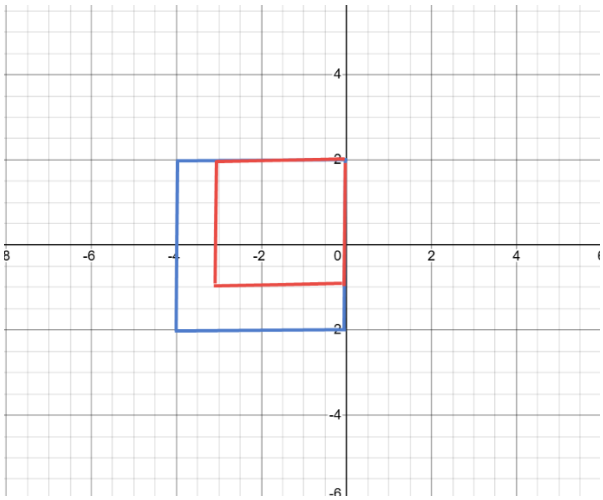
```java
import java.util.*;

public class Solution
{
    public static int overlappingArea(int[] rect1BottomLeft, int[] rect1TopRight, int[] rect2BottomLeft, int[] rect2TopRight)
    {
        int width=0;
        int height=0;

        //used for test case 11
        if (rect1BottomLeft[0]>=rect2TopRight[0]
           || (rect2BottomLeft[0]>=rect1TopRight[0])
            || (rect1BottomLeft[1]>=rect2TopRight[1])
           || (rect2BottomLeft[1]>=rect1TopRight[1]))
           {
               System.out.println("NO OVERLAP FOUND");
               System.exit(0);
           }
```

```java
        //TEST CASE 11
    System.out.println("The overlapping area is: " + overlappingArea(new int[]{-6, 5},  new int[]{-5, 5},new int[]{-8, 2}, new int[]{-4, 3}));
    //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
```

```
NO OVERLAP FOUND
```

I will now need to perform a few tests similar to test case 9.
I will use previous test cases and try to create similar situation...

TEST CASE 12:




```
215
216         //TEST CASE 12
217         System.out.println("The overlapping area is: " + overlappingArea(new int[]{-4, -2},  new int[]{0, 2},new int[]{-3, -1}, new int[]{0, 2}));
218         //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft     //rect2TopRight
```

```
SSSS
-4
TEST
width: 3
height: 3
The overlapping area is: 9
```

Rotating the rectangles as per usual
But I can see I had to disrupt the area of code for Test case 9.
This was not ideal.

TEST CASE 13:

Solution.java

```
48                  System.out.println("rect1 bottom left: " + rect1BottomLeft[0]);
49                   System.out.println("rect2 top right: " + rect2TopRight[0]);
50
51                   //used for test case 13
52                   if (rect1TopRight[0]==rect2TopRight[0])
53                   {
54                       if (rect1BottomLeft[0]>rect2BottomLeft[0])
55                       {
56
57                       width=Math.abs(rect1BottomLeft[0])-Math.abs(rect1TopRight[0]);
58                       height=Math.abs(rect1BottomLeft[1])+Math.abs(rect1TopRight[1]);
59
60                       }
61
62                       else
63                       {
64                           width=Math.abs(rect2BottomLeft[0])-Math.abs(rect2TopRight[0]);
65                           height=Math.abs(rect2BottomLeft[1])+Math.abs(rect2TopRight[1]);
66                           System.out.println("width: " + width);
67                           System.out.println("height: " + height);
68                       }
69
```

```
70
71                      //used for test case 9
72                      //System.out.println(rect1BottomLeft[0]);
73
74                      //width=Math.abs(rect2BottomLeft[0])-Math.abs(rect1TopRight[0]);
75                      //System.out.println("width10: " + width);
76                      //height = Math.abs(0-rect1BottomLeft[1]) - Math.abs(0-rect2TopRight[1]);
77                      // System.out.println("height10: " + width);
78
79                  }
80
```

So although this test case passes 13 (see below):

```
237
238         //TEST CASE 13
239         System.out.println("The overlapping area is: " + overlappingArea(new int[]{-3, -1}, new int[]{0, 2},new int[]{-4, -2},  new int[]{0, 2}));
240         //                                                  //rect1bottomLeft     //rect1TopRight   //rect2BottomLeft    //rect2TopRight
241
```

```
rect1 bottom left: -3
rect2 top right: 0
The overlapping area is: 9
```

Test case 9 now fails..

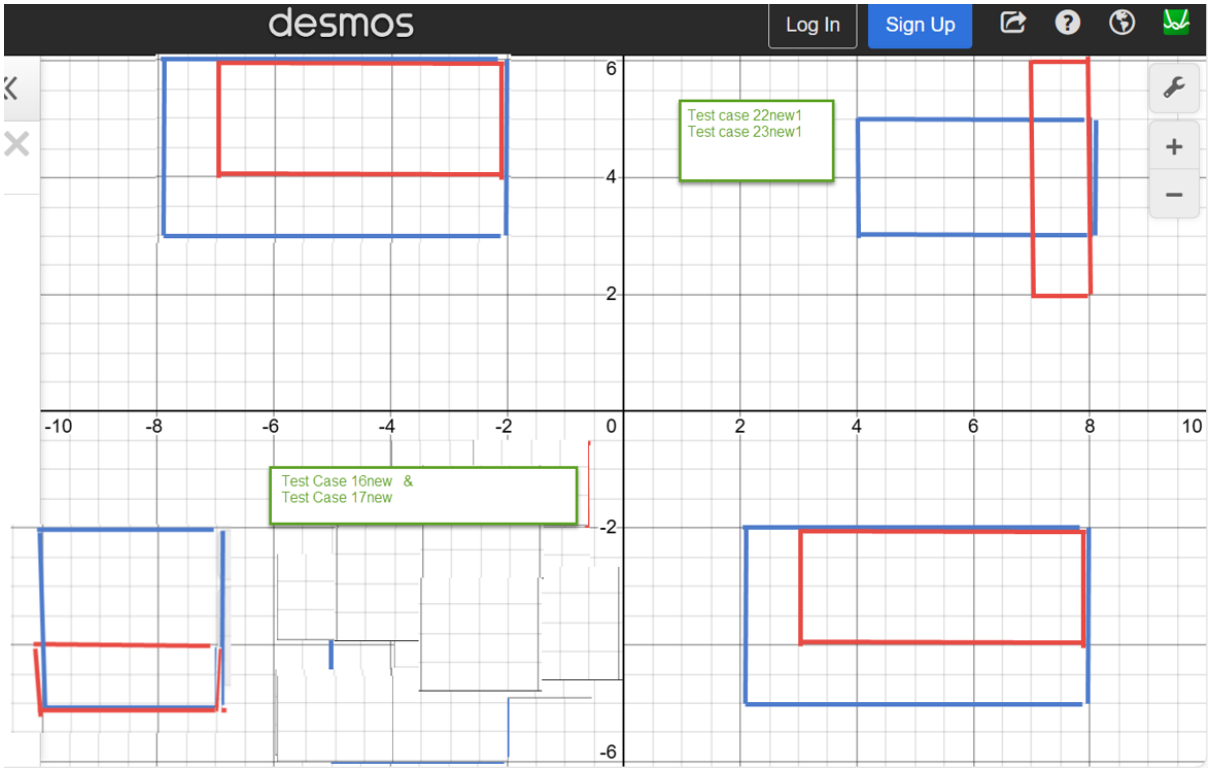
```
rect2 top right: -4
width: 2
height: 4
The overlapping area is: 8
```

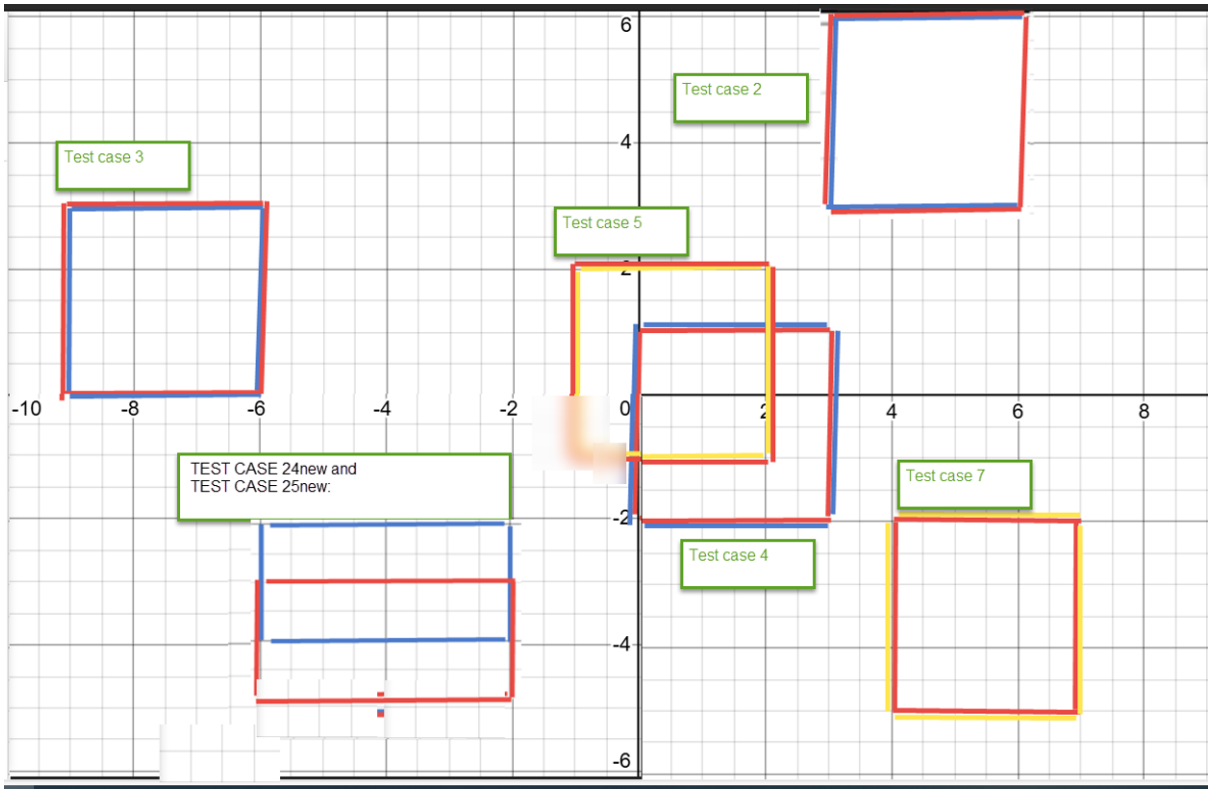
I also need to consider the other possibility of a similar scenario in which the rect1BottomLeft aligns with rect2BottomLeft and examine similar principle to above of one shape fitting exactly inside another (this will be part of test case 14).

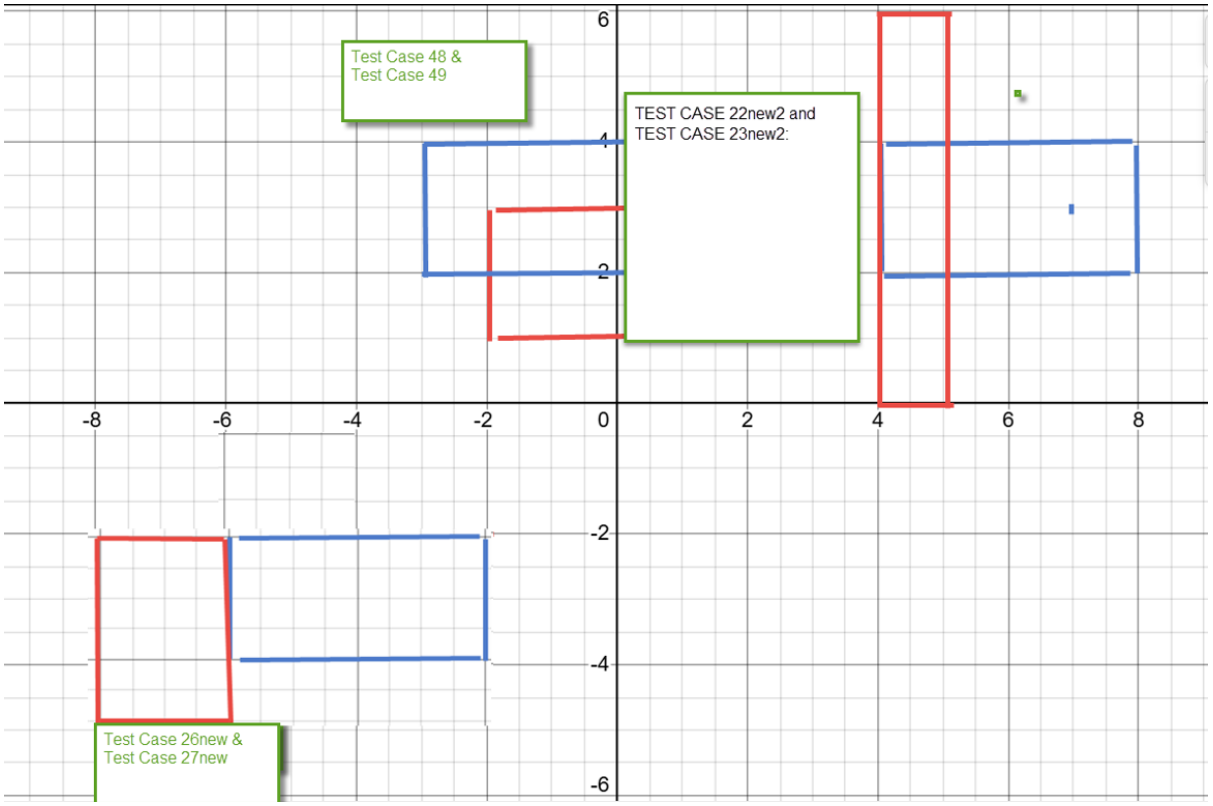
I can imagine this becoming extremely messing in which I will be displacing existing test cases...

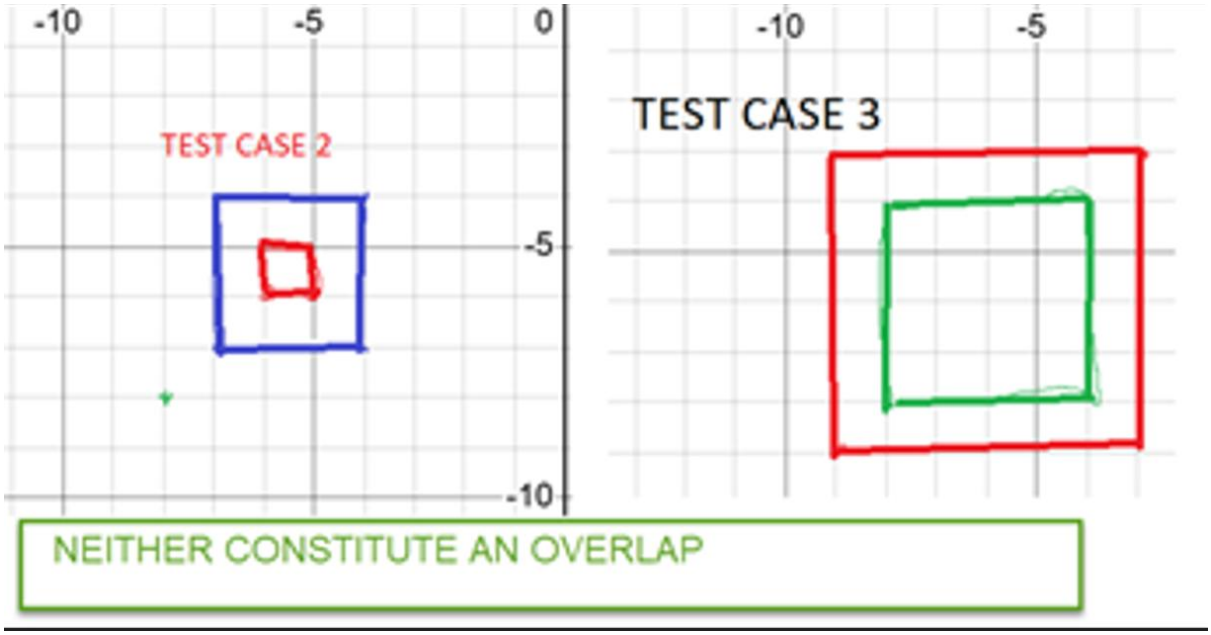
It suggests to me that in this code, my primary objectives need to consider if one shape fits inside another (whether exactly) or fully.
I would then need to consider if there is a partial overlap.
I will not continue with this coding attempt as I can imagine problems will continue to compound.

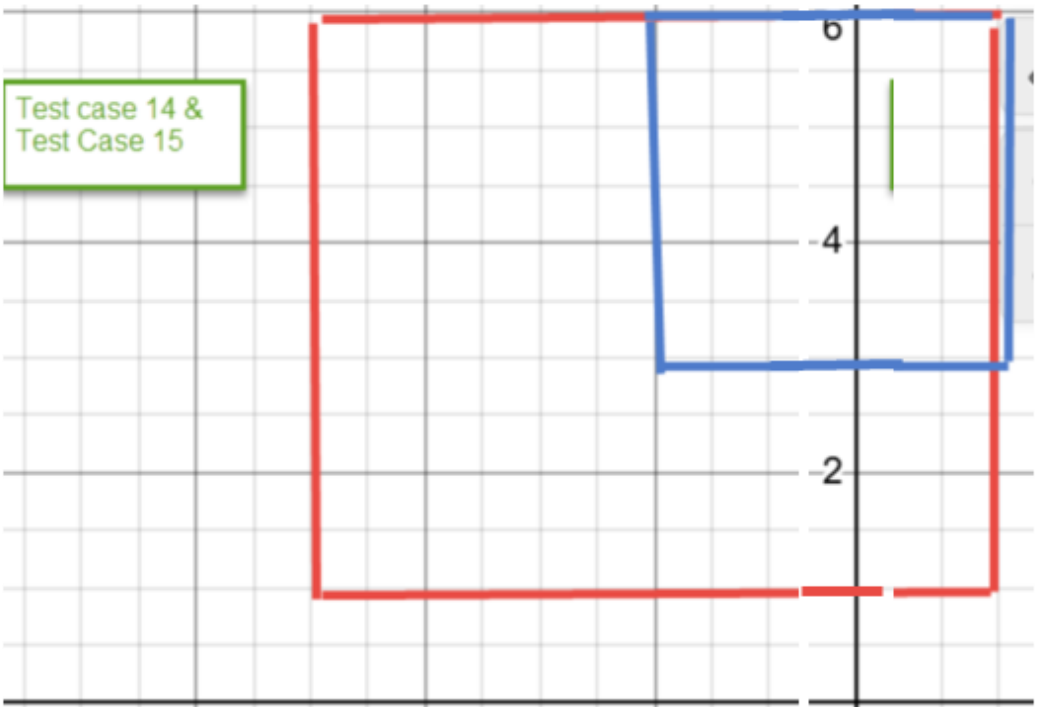
Other miscellaneous photos to try and visualise the challenge:




Test case 2
Test case 3
Test case 5
TEST CASE 24new and
TEST CASE 25new:
Test case 7
Test case 4


Test Case 48 &
Test Case 49
TEST CASE 22new2 and
TEST CASE 23new2:
Test Case 26new &
Test Case 27new


-10
-5
0
-5
-10
TEST CASE 2
-10
-5
TEST CASE 3
NEITHER CONSTITUTE AN OVERLAP


Test case 14 &
Test Case 15
6
4
2