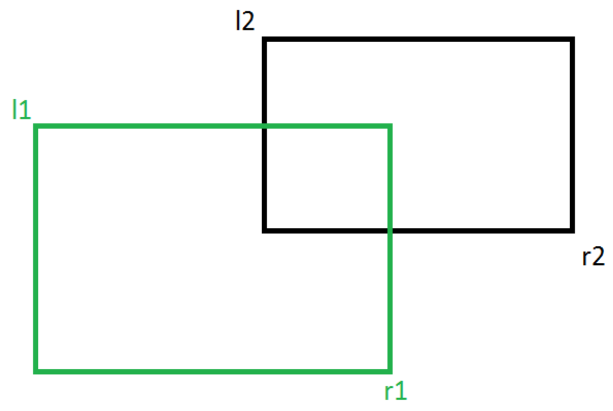
Find if two rectangles overlap

Given two rectangles, find if the given two rectangles overlap or not.

Note that a rectangle can be represented by two coordinates, top left and bottom right. So mainly we are given following four coordinates.
l1: Top Left coordinate of first rectangle.
r1: Bottom Right coordinate of first rectangle.
l2: Top Left coordinate of second rectangle.
r2: Bottom Right coordinate of second rectangle.



We need to write a function *bool doOverlap(l1, r1, l2, r2)* that returns true if the two given rectangles overlap.

One solution is to one by one pick all points of one rectangle and see if the point lies inside the other rectangle or not. This can be done using the algorithm discussed here.
Following is a simpler approach. Two rectangles **do not** overlap if one of the following conditions is true.
1) One rectangle is above top edge of other rectangle.
2) One rectangle is on left side of left edge of other rectangle.

We need to check above cases to find out if given rectangles overlap or not. Following is C++ implementation of the above approach.

```
#include<stdio.h>

struct Point
{
    int x, y;
};

// Returns true if two rectangles (l1, r1) and (l2, r2)
bool doOverlap(Point l1, Point r1, Point l2, Point r2)
{
    // If one rectangle is on left side of other
    if (l1.x > r2.x || l2.x > r1.x)
        return false;

    // If one rectangle is above other
    if (l1.y < r2.y || l2.y < r1.y)
        return false;

    return true;
}

/* Driver program to test above function */
int main()
{
    Point l1 = {0, 10}, r1 = {10, 0};
    Point l2 = {5, 5}, r2 = {15, 0};
    if (doOverlap(l1, r1, l2, r2))
        printf("Rectangles Overlap");
    else
        printf("Rectangles Don't Overlap");
    return 0;
}
```

Output:

```
Rectangles Overlap
```

Time Complexity of above code is O(1) as the code doesn’t have any loop or recursion.