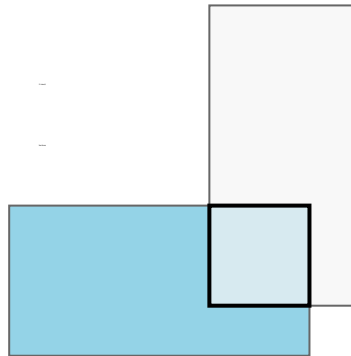


A crack team of love scientists from OkEros (a hot new dating site) have devised a way to represent dating profiles as rectangles on a two-dimensional plane.

They need help writing an algorithm to find the intersection of two users' love rectangles. They suspect finding that intersection is the key to a matching algorithm so *powerful* it will cause an immediate acquisition by Google or Facebook or Obama or something.



Write a function to find the rectangular intersection of two given love rectangles.

As with the example above, love rectangles are always "straight" and never "diagonal." More rigorously: each side is parallel with either the x-axis or the y-axis.

They are defined as instances of the Rectangle class:

```
class Rectangle
{
private:
    // coordinates of bottom left corner
    int leftX_;
    int bottomY_;

    // dimensions
    int width_;
    int height_;

public:
    Rectangle() :
        leftX_(0),
        bottomY_(0),
        width_(0),
        height_(0)
    {
    }

    Rectangle(int leftX, int bottomY, int width, int height) :
        leftX_(leftX),
        bottomY_(bottomY),
        width_(width),
        height_(height)
    {
    }

    int getLeftX() const
    {
        return leftX_;
    }

    int getBottomY() const
    {
        return bottomY_;
    }

    int getWidth() const
    {
```

```
        return width_;
    }

    int getHeight() const
    {
        return height_;
    }

    bool operator==(const Rectangle& other) const
    {
        return
            leftX_ == other.leftX_
            && bottomY_ == other.bottomY_
            && width_ == other.width_
            && height_ == other.height_;
    }

    bool operator!=(const Rectangle& other) const
    {
        return !(*this == other);
    }
};
```

Your output rectangle should be a Rectangle object as well.

Gotchas

What if there is *no* intersection? Does your function do something reasonable in that case?

What if one rectangle is entirely contained in the other? Does your function do something reasonable in that case?

What if the rectangles don't really intersect but share an edge? Does your function do something reasonable in that case?

Do some parts of your function seem very similar? Can they be refactored so you repeat yourself less?

Breakdown

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now → (/upgrade)

Solution

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now → (/upgrade)

Complexity

$O(1)$ time and $O(1)$ space.

Bonus

What if we had a vector of rectangles and wanted to find *all* the rectangular overlaps between all possible pairs of two rectangles within the vector? Note that we'd be returning *a vector of rectangles*.

What if we had a vector of rectangles and wanted to find the overlap between *all* of them, if there was one? Note that we'd be returning *a single rectangle*.

What We Learned

This part requires full access (/upgrade)

You've already used up your free full access questions!

If you subscribe to our weekly newsletter (/free-weekly-coding-interview-problem-newsletter), you'll get another free full access question every week.

To see the full solution and breakdown for *all* of our questions, you'll have to upgrade to full access (/upgrade).

Upgrade now → (/upgrade)

Want more coding interview help?

Check out **interviewcake.com** for more advice, guides, and practice questions.