Problem Set 8 - Computational Geometry

Question C

This question ALSO wasn't the worst thing in the world! Only question D was really the super hard one in this problem set. But man that one was hard. At first when reading this question, the naive train of thought was to check if every B point was inside the A polygon, but (with the means that I know) that would have taken way too long with our 10^5 and 10^4 bounds.

The actual method I came up with was simply running a convex hull over ALL the points, and checking if any of the points of the second set of points lie on the convex hull. This wasn't too hard to implement either. Just needed a way to tag points as A, and tag points as B, which I did by storing a bool alongside the pair of doubles denoted as pt - all in a pair. I'd then parse this into the convex hull function - edit the full hull and the half hull functions to take in this new pair<pt, bool> data type, and then at the end, I would cycle through the hull and check if there are any trues.

At first, I actually changed the hull function to add one to a count everytime we pushed a polygon B dot into the hull, and subtract one everytime we popped a polygon B dot into the hull, but it wasn't working for some reason, and was more complicated than what I finally came up with.

I suspect that the reason the first implementation wasn't working was not because of the actual method, but because I had missed a change I needed to make to the ccw function, that I eventually did with my working solution.

When going around and checking for points on the convex hull, we would pop if any consecutive three points run counterclockwise or straight. However, in the final solution this would prevent a set of collinear points from ending up on the hull, and only include the endpoints of the collinear line segment in the hull. If we are trying to check if there are ANY B points on the convex hull, if a bunch of points all lie on the same straight line on the hull, we need them all to be included on the hull, and thus, the ccw check needed to be relaxed to only return true if the points are strictly counterclockwise (and not straight).

Just changing:

```
bool ccw(pt a, pt b, pt c) {
 return cross(b - a, c - a) >= 0;
}
```

To:

```
bool ccw(pt a, pt b, pt c) {
 return cross(b - a, c - a) > 0;
}
```

And once that was fixed, the solution passed, the problem sets (or at least what I wanted to get done of them) but done!!!