CS 142 Assignment 2
Engineers and Mangers River Crossing Puzzle
See Canvas for due date!

For this assignment, you will program the logic for a river crossing puzzle.

Three Puget Sound area tech companies (Company 1, Company 2, Company 3) have decided to hold an event to improve relations. Each company sends one engineer and one manager to go on a hike. We refer to the six people by the letters A, B, C, D, E, and F. The three engineers and three managers hike for an hour and eventually reach a river which they need to cross. There is a small boat where at most two people can fit at a time. Any person can pilot the boat. There is one caveat: If a manager is in the presence of an engineer of a different company when that engineer's manager is not present (on the same side of the river), the manager would feel obligated to recruit the engineer to join their company. However, this would ruin the event's synergy. Let's model the rules of this problem so that a player could try to solve the puzzle without violating the laws of physics or ruining the event.

Create a class called `CompaniesCrossing`. Inside that class, create a method with the following signature:

```
public static boolean isMoveOkay(int beforeBoat, int beforeA, int beforeB, int beforeC, int beforeD, int beforeE, int beforeF, int afterBoat, int afterA, int afterB, int afterC, int afterD, int afterE, int afterF)
```

The boat and each of the six people have their own "Before" position parameter variable which is 1 if the position is on the left (original) side of the river and 2 if it is on the right (desired) side before the move. In addition, the boat and each of the six people have their own "After" position parameter which is 1 if the player wants the next position to be on the left side and 2 if they want it to be on the right side. Only a few positions can change in one move!

Running `CompaniesCrossingTest` will tell you the role and employer for each of the 6 people.

Inside the `isMoveOkay` method, you'll implement logic to check whether the After positions are compatible with the rules of the game compared with the Before positions. An overview of the logic follows. **Note that at most one error message should be printed, and the order of the checks determines which error message to print.**

First, make sure that all parameter variables are 1 or 2. If not, print the following error message and return false:
`All positions must be 1 or 2!`

Second, make sure that the boat is changing position. If not, print the following error message and return false:
`The boat must move!`

Third, make sure that each person you're trying to move is on the same side as the boat. If not, print the following error message and return false:
`You may not move a person who is not with the boat!`

Fourth, make sure you're moving one or two people. If not, print the following error message and return false:
`You must move one or two people!`

Fifth, make sure that no manager would recruit anyone in the **after** positions. There will be three separate checks, one for the managers from companies 1, 2, and 3, in that order. For each check, if a recruitment would happen, print the following error message, replacing ***X*** with 1, 2, or 3 as appropriate to the manager's company and return false:
`Company X manager would try to recruit someone!`

Finally, if none of the errors above occurred, check to see if the after positions solve the puzzle. If so, print:
`You solved the puzzle!`

If no errors occurred, you should return true to indicate that the move is okay. **A tester program will test your code and provide helpful messages if there are problems!**