Towers of Hanoi

You have four files:

- `Hanoi.java` is yours. It's the only file you change.
- `HanoiView.java` draws the pegs and animates the moves. You don't need to read it.
- `TestMove.java` and `TestIsLegal.java` check two of your methods without opening the window.

The pegs are numbered 0, 1, 2. Each peg is an `ArrayList<Integer>` of disk sizes, with the **bottom disk at index 0** and the **top disk at the end**. Disks are numbered 1 (smallest) to n. So with 3 disks, the starting position is

```
pegs.get(0)  →  [3, 2, 1]
pegs.get(1)  →  []
pegs.get(2)  →  []
```

The goal is to move the whole tower to peg 2. You may move one disk at a time, always the top disk of a peg, and you may never put a disk on top of a smaller one.

Run `Hanoi` now. It compiles, but nothing works yet.

Part 1: Playing by hand

Fill in `isLegal` and `move` , testing each one before going on. A test program sets up the pegs in several different ways, calls your method, and prints `pass` or `FAIL` for each. A `FAIL` line shows the pegs it started with and what went wrong.

Write `isLegal`

`isLegal(from, to)` returns whether the top disk of `from` may go on `to`. Think about what happens when either peg is empty. `isLegal` only answers the question; it must not change the pegs. The first two lines of the method get the ArrayLists for `from` and `to`. What must you check to know if the move is legal? What do you need to know about `from` and what do you need to know about `to`? Write your code and run `TestIsLegal` to check it.

Write the missing lines of isLegal here

Run `TestIsLegal`. Get all 11 tests passing.

Write `move`

`move(from, to)` takes the top disk off peg `from` and puts it on peg `to`. (There is a call to `view.show(from, to)` is already there. Leave it as the last line: it must come **after** you change the pegs, because it checks

that your pegs match the move.) `move` doesn't check whether the move is legal; that's the next method's job.

How do you move a disk? What happens to the `from` ArrayList and what happens to the `to` ArrayList? Write your code. (Note you may need to borrow code from isLegal)

Write the missing lines of move here

Run `TestMove`. Get all 7 tests passing.

Play by hand

When both work, you can play: click a peg to pick up its top disk, then click another peg to drop it. An illegal drop does nothing.

1. Solve the 3-disk puzzle by hand. How many moves did you use? Can you do it in fewer?
2. Solve the 4-disk puzzle by hand. What was the hardest part?

Part 2: The recursion

Fill in `hanoi(n, a, b, c)`, which moves the top `n` disks from peg `a` to peg `b`, using peg `c` for temporary storage. Then press **Solve**.

With any recursion you need to think about the base case and the recursive case. We discussed both in class. What is the smallest problem you can solve? That's the base case. And you know `hanoi` calls `hanoi` twice.

Write your recursive solution HERE

Part 3: Exploring

Check your code by running it. If it solves each size in the right number of moves, you must be correct!