After one round, contestants are moved at random. moving through the list, contestants get moved to first [0] if answer the question right(depends on coin flip) and move to last position [contestant.size()-1] if answer question wrong (coin flip). I know my code worked perfectly because there was no bugs (I have vision issue so there might have been a typo here and there but the program ran and no bugs were annouced so I believe it worked), and I printed the initial positions to compare them to the after 1 round positions, since they are very different, my code worked.

Average positions after 12 rounds:

B: 3.642857142857143

G: 3.6

H: 3.142857142857143

D: 2.6363636363636362

A: 2.375

C: 3.875

F: 4.071428571428571

E: 4.0

I ran 12 rounds with 8 contestants. Since the average positions after 12 rounds are pretty evenly distributed and the averages are roughly close to the middle (around the value n/2, which is 8/2 = 4), we know that this algorithm isn't biased against certain Contestants, that is, no Contestant gets stuck at one end of the line or the other for most of the game.
Time took (in nanoseconds): 501125.

rounds	10	100	1000	10000
time when n = 43 (nanoseconds)	898125	5698958	11236708	107943375
time when n = 45000 (nanoseconds)	3088195125	32342276459	317492644167	

8. Alternative approach: Just run through the entire list for each round, ask each contestant a question, if correct (random coinflip), then swap with person before them If incorrect, then swap with person behind.

```
public static void simulateRound(ArrayList<Contestant> contestants) {
for (int i = 0; i < contestants.size(); i++) {
boolean correctAnswer = flipCoin(); // Flip coin for answer
if (correctAnswer && i > 0) {
//Swap with the person before if correct
swap(contestants, i, i - 1);
} else if (!correctAnswer && i < contestants.size() - 1) {
// Swap with the person after if incorrect
swap(contestants, i, i + 1);
}
}
}


//Swap method
public static void swap(ArrayList<Contestant> list, int i, int j) {
Contestant temp = list.get(i);
list.set(i, list.get(j));
list.set(j, temp);
}
```

It's easier to implement since you don't have to remove and add everytime, and the time would be shorter since you wouldn't have to move up and down the list everytime.

9.

- I learned methods for list such as swap, remove, add; and looping through each items of list using `for (Contestant c : contestants).`
- I liked that it was ArrayList, I've been doing a lot of leetcode with ArrayList so everything was fresh on my mind.
- It was just so long and the last number of contestants was so large and heavy for the program.
- I realized at the end that I was supposed to commit after each step, if I had more time, I would love to go back and take codes out and redo everything.