Unit 2:

3: To test my program, I first ensured that all classes (Person, Contestant, and ContestDriver) were properly implemented and compiled without errors. I used print statements to verify the initial state of the contestants and checked their positions after each round of simulation. I ran the program with various numbers of contestants and rounds to ensure the correct simulation behavior, including verifying that the positions were updated as expected. Additionally, I tested the handling of both ArrayList and LinkedList to ensure compatibility with the program logic.

rounds	10	100	1000	10000
time when n = 43 (nanoseconds)	850166	4411667	10074625	60726875
time when n = 45000 (nanoseconds)	68347075458	58603365087	317492644167	~3B

In this approach, each contestant swaps positions based on whether they answer correctly or not. Using an ArrayListmakes these swaps faster because accessing and modifying elements by index is efficient. However, swapping elements involves shifting data, which can become slower if the list is large. On the other hand, a LinkedList would be slower for this algorithm because it requires traversing nodes to reach a specific index before performing swaps. Since this algorithm frequently accesses and modifies elements by index, an ArrayList is generally a better choice for this task.

What did you learn? I learned that ArrayLists generally offer better performance for random access and resizing, while LinkedLists are more efficient for insertions and deletions in the middle of the list. The trade-off between the two depends on the specific operations performed on the list.

What did you like about this project? I liked that the project provided an opportunity to compare and contrast the performance of different data structures, such as ArrayLists and LinkedLists, in a real-world context, helping to better understand their strengths and weaknesses.

What did you find confusing or would like to see done differently regarding this project? I found the timing of the different operations a bit tricky since the sample sizes were so large.

If you had another hour or two, what would you like to add to the project or how would you do things differently?If I had more time, I would further optimize the algorithm for rearranging contestants by incorporating more advanced techniques (like a priority queue or different sorting strategies)