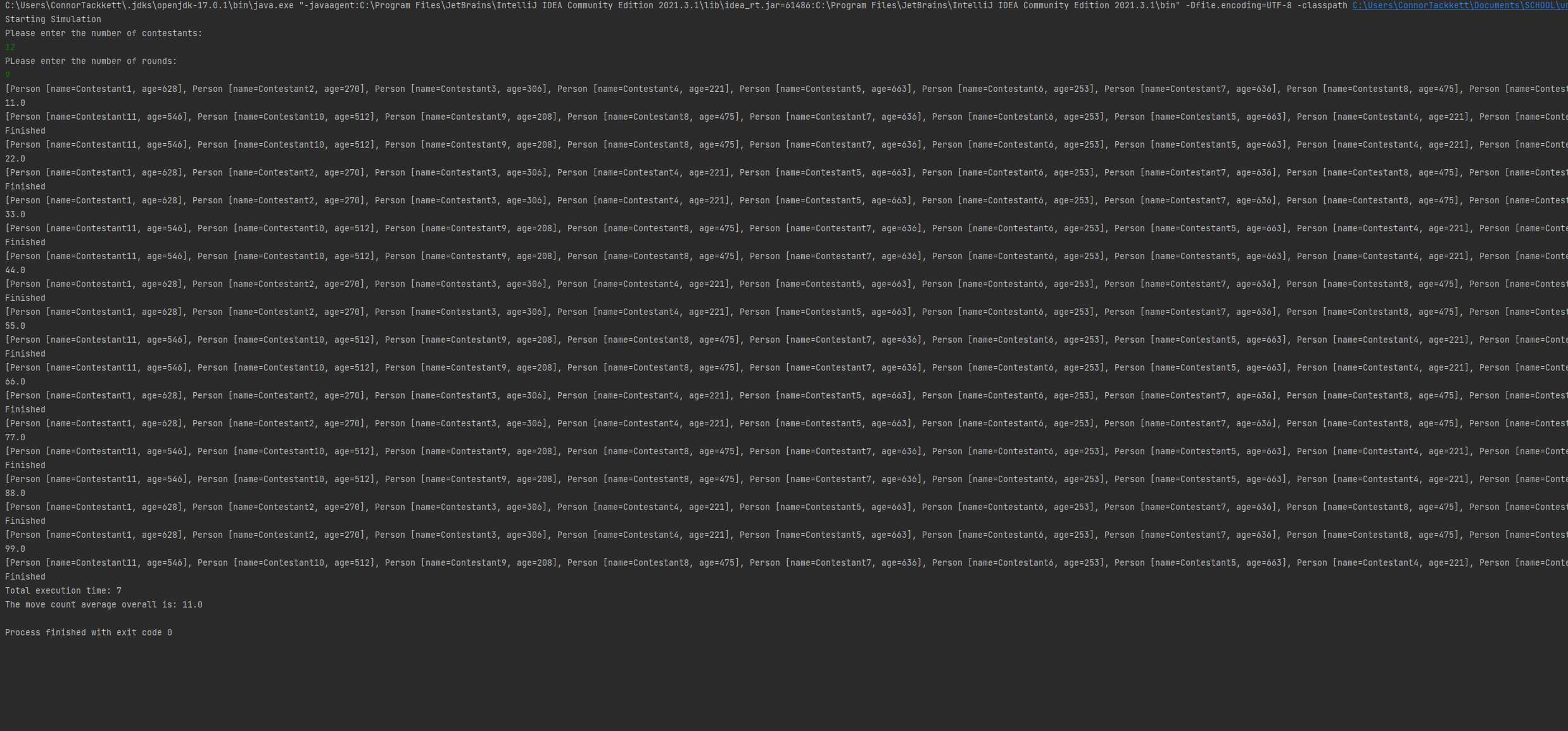
**How I Know It Worked:**

My idea for checking if it was working towards the beginning was by making the numbers of contestants and rounds low, I also used low numbers unlike those in my student numbers. Numbers no higher than 5 work well for the test. I added print statements to see the before and after results to make sure duplicates moved to the back and others moved ahead. I had to go through a lot of trial and error to make sure it was working properly and also just in general.

**Explanation and Showing Data:**



The creation of the counter for the number of moves wasn’t all that difficult. I already had a section of code as well that was an area where things were done if contestants were being moved to the front of the list. This is where I chose to iterate the “totalMoveCount”. The average of the number of times that a contestant was moved to the front can be calculated in different ways. An overall average, like above, can be calculated for example. You could also calculate as an average per contestant and you could even do an average per round. Taking the total count and dividing it by the number of rounds would give you the average. If you wanted the more extreme average moves per round you can divide the count by both the number of contestants and the number of rounds.

**Time Table Statistics:**

|  |  |  |
| --- | --- | --- |
| **CONTESTANTS** | **ROUNDS** | **TIME(MILLISECONDS)** |
| 12 | 9 | 0 |
| 34 | 10 | 1 |
| 34 | 100 | 3 |
| 34 | 1000 | 15 |
| 34 | 10000 | 47 |
| 47000 | 10 | 4118 |
| 47000 | 100 | 41515 |
| 47000 | 1000 | 418423 |
| 47000 | 10000 | 4217218 |

**LinkedList Table Statistics:**

|  |  |  |
| --- | --- | --- |
| **CONTESTANTS** | **ROUNDS** | **TIME(MILLISECONDS)** |
| 12 | 9 | 0 |
| 34 | 10 | 1 |
| 34 | 100 | 4 |
| 34 | 1000 | 16 |
| 34 | 10000 | 54 |
| 47000 | 10 | 55078 |
| 47000 | 100 | 541358 |
| 47000 | 1000 | 5320972 |
| 47000 | 10000 | 52299480 |

**Reflection Questions:**

* In a sentence or two, what did you learn?

I learned a lot. I remembered how to use the scanner for one, but other than that I learned a lot about working with ArrayLists and mostly just how to think about how to solve this algorithm. I ran into a lot of problems with figuring out how to make the loop only sort on elements it hadn’t touched before. I learned a lot about sorting, how to create the restrains on the numbers the randomID could be, and a lot more.

* In a sentence or two, what did you like about this project?

I actually found a lot about this project very interesting. I liked sorting data. I liked creating restraints for the data, and it was also very fun to watch the data make patterns when I ran the speed tests and there were a lot of print statements still within. I thought it was cool to make an algorithm work and the way that it had to sort the numbers was interesting as well. Also, I just realized something is wrong with the program that is massively bad and I need to go fix it right now.

* In a sentence or two, what did you find confusing or would like to see done differently regarding this project?

Oh I found a great deal of this assignment confusing. It’s funny after being sick and working on it all day I find it hard to think about everything I already did. There were a lot of things I had to learn about and look up though and there was a lot that was confusing. The most confusing part was making the algorithm work. Not necessarily the sorting part of the algorithm but the making sure it stopped once everything was looked at and appropriately moving the index counter. One thing I noticed was that the comments within the code don’t match up with the directions at all, so that bothered me. Also some of the directions were just a little bit ambiguous.

* In a paragraph or two, explain any timing differences from using an ArrayList vs. a LinkedList. Speculate on what might be causing these differences.

I had to retime so many times because I realized how much of an impact my print statements were having on the time. Originally I was printing the original array and the sorted one, and that was slowing it down. Removing them and only leaving the print statements that were in, “start” and “finish” dropped the time a lot.

Some interesting things can be seen by the time tables. I think what is interesting to note is that while the arraylist times steadily go up in the amount of time taken, that is not the case for linkedlists. Once the number of contestants gets higher, the time it takes for a linkedlist to sort exploded. Using an arraylist is all around faster in this case and my idea behind why is because linkedlists cannot randomly access the elements within the list. It has to navigate through it front and back while an arraylist can navigate and sort freely, which doesn’t take as long. The final test for the linked list was taking so long that I had to estimate. I divided the 47000 by 1000 time by the time before that to see what the growth rate was and then multiplied the most recent time by the growth rate to receive the estimated time. If those milliseconds are converted into something easy for us to think about, then the final linkedlist test would have taken 14 hours…

9.82893351247322 this was the rate by which I multiplied.

* In a paragraph or two, if you had another hour or two, what would you like to add to the project or how would you do things differently? Especially consider how you would write the program without the specific constraints of the assignment.

I might have attempted to use multiple array lists to sort, I tried to in the assignment but then I reread the instructions. There were multiple times were I reread the instructions and found something that I had missed earlier on. I think I would try to just condense my code and make it more elegant. There are certainly spots where things could be written better I am sure. For example earlier on when I was working on this I had a while loop that was, while true. I realized that was quite dumb and I changed it to something better and shorter which is on line 27 of the driver. Regarding the two arrays my initial thought was to use one as the sorted array and one as the original array, but other than that I am currently unsure.