Hey there,

I just wanted to say a few things about the assessment and how I approached things.

First, I have to ask, is this a normal assessment that you guys always do? Or was this specially made for the start of the F1 season?

If you didn’t know, the Formula 1 season started this weekend. So, this was a very perfectly timed assessment. (Did you watch the race? It’s going to be an interesting season)

Anyways, on to the assessment.

There are a couple of points on the instructions that I felt could be taken a couple ways. I am going to quickly explain my logic behind what I did

1. The outcome of a race is represented by the class **RaceResult**, which in turn contains a list of **RoundResult** objects.
   1. A RoundResult is an object with two elements, a round number and a cars position array. The cars position array represents the position on the track of each car on a given round.

So, my cars position array is just a plain ol’ array. The indexes represent the cars, and the values are the positions. First, I assume that is what is asked in the instructions, but it could be an array of objects that represent the positions. This would make it easier for a human to read, but I did create it with the idea that this information could be fed to any other program (front end) and it would still be easily read.

1. The first car that arrives at the last element wins. If two or more cars arrive at the last element at the same time, it's a draw.

I felt this could be taken two ways. One, if multiple cars get to the final element, it is considered a tie (both in first). Or two, if multiple cars get to the final element, check who actually got there first.

Translating this to the real world for a moment.

Let’s say two cars are racing.

Car A is at total distance 1995 with speed of 15, Car B is at 1985 with a speed of 16. Both of these cars finish the race on the ‘same round’ in our game, but it seems pretty obvious that car A should be the winner, as they would have crossed the finish line first. As a result of this thinking, I have designed the positions to be accurate down to the element level.

Also, as the results are just a RaceResult, there is no indicator of a winner other than position.

I do make sure all Cars finish their round even if a winner is found. I also do a check every round if there are cars tied for any position (element accuracy). I move cars up the order for any missing positions (ie, 1 car in first, 2 cars in second, 1 car in third… etc).

So, in the final results, if there is a tie, it will show two or more Cars with a position of 1. (and will show a tie for any other position as well)

A quick word on how I designed my solution.

(more below)

Regarding the track, I chose to make a Track object, which contains a list of TrackSections. The TrackSections have a length property. The length is 40, there are 50 TrackSections in the Track. This makes the 2000 Elements required. The Track selects between 23 and 27 tracks of one type, and the remaining spots are filled with the other type. Then is shuffled.

The cars do not start on the track. Initial placement they are at null (behind the start line). Movement is then added from there.

I created a Settings class to hold many options that I felt could be changeable settings. This could be improved, but is quite useful.

The last thing I want to talk about is the carMove function in the Race class.

I use multiple return statements in that function. I know some people are really against that sort of thing, but in my opinion, sometimes it is more readable to have the multiple returns, than a very nested statement. I had to mention that as I did consciencely choose to do multiple returns rather than a very nested statement

Thanks for taking the time to read this. I hope to discuss this further in an interview.

Kyle Crawford.