Software Project Report

For Stage 1, the first problem that I tackled was forming the structure of my code. I created additional separate classes, one for the Rides and one for the Cars. This was done to incorporate the attributes that each ride or car object possessed such as the 'earliest start of a ride' or the 'number of rides for a particular vehicle'. This was an example of how I used refactoring to improve the design of my code. I felt it was much simpler to separate the rides and cars into their own classes rather than focus on the rides in the 'WorldAndRides' file and focus on the cars in the 'Allocation' file. I stored both the Rides and Car were stored as ArrayLists because the number of cars and rides can vary, so an ArrayList was the most appropriate structure to store the data. In this stage, there was a simulation file that took the worldAndRides files and allocation files as input. For stage 2, my approach was based on reusing the components that I had already made in stage 1 to try and produce a more refined allocation strategy. The structure was slightly different as instead of a simulation file, which takes both the worldAndRides and allocation files as inputs, the allocation file was the output file, and it took only one worldAndRides file as an input.

I approached stage 1 by storing the components of the world and rides in ArrayLists. It was important that these two components were stored separately as they had different uses. After, I stored the Car components in a Car ArrayList. This was then used in the allocation class to store each line of the allocation file as a car and assign the numbers in each line to the number of rides and the id of the ride that is assigned to the car. For the Allocation, my strategy at first was to store the id of the rides as integers in the Cars. However, after assessing my strategy, I realised that I had to store the ids of each ride as elements of an ID ArrayList in the Car class. After following this route, the rest of stage 1 became simple for me to complete. For the Simulation, it was now easy for me to access all components of the separate classes and files. My approach to the simulation was to access each ride of a car through the ids stored in the Car ArrayList. After this I could find the three key components of the simulation, the start length (the time it took for a car to reach its start point from its current position), the ride length (the time it took for the car to reach its end position from the start position) and the wait time (the time it took for a car to start after it had reached its start position). Additionally, I had to store a clock that would reset for each car in the simulation. Lastly, I had to compute scenarios for when a car finished too early or when a car finished too late and for when a car started too early or started too late. Firstly, for stage 2, I had to sort my rides. I had to sort the rides because I realised that the best way to allocate rides would be to allocate the

rides in the order of which rides had started earliest. Therefore, I sorted the rides in order of their earliest start. Secondly, I attempted to use the strategy of assigning each ride to a vacant car. This worked perfectly fine as each car had an equal number of rides.

In terms of my testing process, I thoroughly used to debug my allocation strategy to find out how I had to store the IDs of each ride. Debugging allowed me to read each line and see that it was not possible to store the IDs of each ride as just integers with no data structure to hold the integers. I did not see this problem at first and I went ahead to try to complete my simulation, but after debugging, I was able to locate the problem in the allocation file. This allowed me to pass all the given tests. Despite all tests working, I was faced with a hidden problem that I did not face with the tests that I was given. This problem was that the wait time was not taken into consideration. Therefore, I wrote my own test that would force my solution to compute a wait time. As a result, I was able to incorporate a wait time into my solution. For stage 2, despite my solution being able to allocate 1 ride to 1 car, through testing and evaluation, I understood that the number of points that my solution would produce was limited in terms of the points I could gave from this. Therefore, the quality of my solution was not great. I tried to improve this by increasing the number of rides that each car would be allocated. By increasing the number of rides allocated to each car, I sacrificed points in the “a_example.in” and “b_should_be_easy.in” tests but it was worth it as the number of cars and rides in the other 3 tests was much higher meaning there was more points to be gained in those tests. Although this did increase the number of points, it was not enough for me to get over 10 million to gain another 5 marks for my assignment.

As a result of this assignment, the aspect that I found challenging was understanding certain details of this actual task. For example, I did not understand at first, how to structure task 2 as I was not sure whether a simulation file was needed or not. However, the aspect of the assignment that I am most proud about is my quickness in carrying out certain parts of the tasks after understanding the content. Overall, I felt I worked at a good pace; however, I could have done better with my time as Stage 3 is still incomplete and Stage 2 could still have a higher score. Altogether, I worked on the project for an hour a day each week, sometimes two. This totals up to roughly 10-14 hours a week. So, into total, taking away the weeks where I did not commit anything, I worked on this for roughly 100-110 hours.