**Test Driven Development Report**

The test we chose to show during our demonstration video was a test for our distance calculations using the haversine formula. We chose this test as the method was one of the more difficult ones in the project and it has limitless possibilities for testing it. The method was to calculate the distance between the ZIP code entered by the user and the ZIP codes of the hospitals stored in the database using their respective longitudes and latitudes.

For testing, we decided first to measure a country (Netherlands) from top to bottom. In our example, this returned a variable - of type double - containing the distance calculated in meters. We know that a double is prone to rounding errors. Moreover, we know that the method used in this calculation – the haversine formula – is somewhat *imprecise* for our planet. It is enough for our application now, but perhaps we’ll want to replace it with higher-precision calculation in the future. As stated above, we calculated the distance across the Netherlands, however one test passing isn’t enough to prove the method works. Because of this, we tested the method with the distance for a few more countries.

At this point, we have covered some base cases – the main functionality of the method seems to work. We then decided to go further and stretch our code to the limit. Our planet is more like a sphere as opposed to a plane. This means that there is a place where latitude goes from 180 to -180, leaving a *space* where we should be careful. We decided to test the method further: from one pole to another, max distance on the planet, really small distance, if both points are identical, etc.

Next, we began to think of negative cases: when the method is not to run; like a controlled failure. Latitude is supposed to be defined in exactly [-90, 90] degrees range and longitude in [-180, 180] range. This means that in a case in which we passed an invalid value, our method throws an exception. The same tests were run for both latitude and longitude parameters.

Evidence of all tests that were created during the development of our project can be found here: <https://github.com/RossMitchell1999/Agile_Development_Project1/tree/master/Sprint%202/Test_Driven_Development>.

Most of it was based on sorting and searching as we wanted to implement that as soon as possible, so many sorting algorithms had to be tested(BubbleSort, InsertionSort, MergeSort, QuickSort) in order to figure out the most essential, easy and practical sorting for the project.