Final Test Report

To succeed in this project, we have decided to do our testing by first doing Blackbox Unit Testing, then Whitebox Unit Testing, next Integration Testing, and lastly, Acceptance Testing. When we did Blackbox Unit Testing for all the functions we created, we considered the possible inputs the user can enter and compared the expected and actual outputs. For example, we thought of all the possible inputs the user can enter for the destination, such as negative numbers, or numbers greater than 25 since there is no existing coordinate. We also used different scenarios such as creating our own destination point as well as assuming the weight and volume of each truck and package and determining what the output would be. After Blackbox Unit Testing, our next testing was Whitebox Unit Testing, which was looking at the actual function definition, rather than the prototype. This was like Blackbox Unit Testing, but we made our tests in detail that is related to the definition itself. After Whitebox Unit Testing, we did our Integration Unit Testing. This testing allows us to combine multiple functions and see if they're communicating properly. We know that each individual function works perfectly fine, we should also combine a few functions to see if they're working together properly and making sure no unexpected changes in the output are shown. Lastly, after Integration Testing is done, we do the Acceptance Testing. This testing is done by the user who requested to do this. This will also verify to see if the code is ready by knowing if it compiles and gives the correct output. These are the four tests done in steps to achieve the goal of knowing which truck to take based on the route, package's weight, and volume.

Throughout this project, we have discovered some major bugs during Unit Testing. We first discovered a bug in the calculateUtilizationScore() function since it did not have an if statement to check for a negative value. To fix this bug, we added an if statement to this function. Another major bug was found during the Integration Test in the read() function. This bug was preventing the program from exiting when entering 0 0 x. To fix this bug, we modified the code and made the program check for the exit code first. getAllTruckPath() function was also producing a bug since it was returning an empty route struct so to fix this bug, we used the addPtToRoute() function to add the destination point to the result struct. This allowed the constainDestination() function to check the structure and ensure that it can reach its destination. Also, a bug was discovered in the shortestPath() function which was adding more than MAX_ROUTE (100) to the structure that caused the program to crash. Therefore, to prevent shortestPath() from exceeding the MAX_ROUTE, we updated the while loop.