The in-class report should provide: i) a summary list of your test cases in Part A and test cases in Part B, ii) execution test results (numbers passed and numbers failed – with your defined severity level), iii) any fixing or re-test performed and iv) a discussion of your experiences (e.g. what was difficult to design; how much effort in person hours did it take to design test cases, execute the test cases and record the material; what was effective and what was not, etc.). (*Look at Lecture Notes on Test Planning for hint of Test Result Report.)*



The full list of our black box test cases can be found in appendix A and the diagram in which the white box test cases can be found in appendix B. A quick breakdown of the black box testing can be seen above in our decision table. This method allowed us to quickly determine which tests would be most likely to produce errors and bust through robust boundary value testing theoretically. Our path (white box) testing was performed later and can be seen in appendix B. It covered all main paths, but not all possible paths due to time constraints. Input validation tests were also performed.

Our levels of severity were as follows: 1. The program fails to complete/run. 2. The program produces improper output. 3. Any formatting problems or any noted lack of optimization.

Our first error that was encountered during input validation testing was a level 1 error. If invalid data was entered, the program would loop infinitely asking for user input. This was due to a failure to clear the input stream. Three level 2 errors were found, and they were found during black box testing. First, there were some round off errors in the tax percentage output e.g. .25 became .25000000001. This was caused by using doubles in Java and was quickly resolved by using decimals instead. Secondly the program didn’t output the amount of money owed, it output only the tax percentage. This was caused by a misinterpretation of the requirements document. The third and final level 2 error found in this program was the tax percentage was not properly calculated if the number of dependents equaled zero. The first of the two level 3 errors was a typo in the printed statement “Please enter you’re annual income”. The word “you’re” was fixed to show the proper word, “your”. The other level 3 error was a memory leak in the output streams, which were not closed once they were no longer needed. All of the previously stated errors were found during black box testing and were submitted to the developer shortly after they were found. The developer would then edit the software to fix the errors submitted to him. Once a new version of the software was released for testing, all of the black box tests were run again to ensure that the errors were no longer present. The white box test cases were completed shortly after the black box testing was done. All of the main paths covered in the white box test cases operated as expected and no defects were found.

Designing the black box test cases using a decision table was fairly simple and fast to do. The decision table was designed and completed in approximately 35 minutes. The expected values for the test cases where manually calculated and recorded in a word document prior to running the test cases. The test cases generated from the table were run and evaluated in approximately 110 minutes, which includes the time from the retests of updated versions of the software. The test cases were run using the JUnit assert function within a wrapper class of the software. After entering the assert commands into the file, running the tests were very simple. All that needed to be done was execute the wrapper class and the test cases would be run. If any of the test cases failed, the program would stop on the specific case that failed and the values were recorded in a word document. In our case, after recording the values of the failed test case, the developer was notified of the issue almost immediately. The white box test cases were generated using a \_\_ path diagram. The test cases generated from the diagram covered the main paths that the typical user would take and they were run using the same method used for the black box test cases. This method of testing, recording, and notifying the developer turned out to be effective for a project of this scale because bugs were fixed just as fast as they were found. All of the test cases and their final results as of the last time running them are included in appendix A and B. Appendix A contains the black box test cases and appendix B contains the diagram in which the white box test cases were generated.