**1. Are any of your tests inconclusive?**

All tests serve to determine an expected outcome. None of the tests were designed to remain undetermined after execution.

**2. Do any of the tests fail? If so, do these failed tests detect failures that cause the program to become inoperable?**

We had seven tests result in a fail status upon first execution. Of the seven tests that discovered an error with the Turing Machine application none of them were failures that would result in an inoperable program. Two errors showed that appropriate error messages were not displayed. Three resulted in conditional bounds not be checked when loading, inserting, and deleting stings in the Input String list. One error detected the inability to save the input string file, if edited during operation, back to the C:\ drive upon exit. The final error discovered was that the application did not terminate upon Exit of the application as defined by program requirements.

**3. Are some of the requirements ambiguous or too high level such that deriving test cases from them was difficult?**

Our situation may not be the norm as the system requirements used to derive the requirements for test were methodically crafted by Dr. Neil Corrigan for use in CPT\_S 322. Dr. Corrigan has been teaching the course for many years and would boast about the specificity of the requirements that he presented to the class. During Team 2’s evaluation of the requirements documentation it was easy to derive requirements that would govern our tests.