Software Test Plan

# Test Plan Identifier

Turing Machine Assignment 6 Test Plan – Unit Testing, Class Diagram based testing and Combinational testing for TM ver. 1.1

# Test Items

- Version 1.1 of the Turing Machine application code

- Unit testing for the main Turing Machine classes, in other words the classes which do more than simply act as collections for data

- Code Coverage report from the Unit testing suite

- Combinational testing of the Turing Machine to be run manually

- Class Diagrams for the Turing Machine application

# Features to be Tested

- Unit testing for the Turing Machine, in particular the alphabets, tape, Turing Machine proper and Turing Machine transitions

- Combinational testing for the application as a whole

- Class Diagrams developed for the application

# Approach

For this test plan, we will take the approach of creating unit tests to perform a method of whitebox testing on the pieces of the application. Further, we will manually perform combination testing as a way of black box testing. As a group, we decided that we would exclude from testing classes which only serve as collections for information that is used later in the application as these simply have setter and getter functionality.

# Test Log

Unit Tests are in the UnitTestFolder inside the tmTester folder. This is a Visual Studio 2013 project which works in conjunction with the project file in the turingMachine folder on the github repository developed by both Chris Wilkins and Michael Degan.

Code Coverage is in the main folder, inside CodeCoverageReport.txt derived from the unit tests in UnitTestFolder, created by Visual Studio 2013 Ultimate built-in tool and run by Chris Wilkins

Combinational testing is in \_\_\_\_\_ and was derived and run by Daniel Martinez

Class diagrams and test cases derived from them are available in the ClassDiagrams folder, as developed by Cody Curry

# Test Summary

Unit Test results were run and all passed successfully using Visual Studio 2013. If all tests return success, then the tests are successful.

Code Coverage results show (in order) number of blocks not covered, % of blocks not covered, number of blocks covered and % of blocks covered. Code coverage showed 55% coverage from the unit tests