Software Test Plan

# Test Plan Identifier

Turing Machine Assignment 4 Test Plan – Fault model testing TM ver. 1.0

Specify the unique identifier assigned to this test plan.

# Test Items

- Version 1.0 of the Turing Machine application code as submitted to Neil Corrigan in May 2014

- Valid Definition file for a Turing Machine that accepts strings of a number of characters “a” followed by the same number “b” by default titled “valid.def”

- Invalid Definition file, only differing from the Valid definition file by missing a single transition by default titled “invalid.def

- Input string file containing the strings “aaabbb”, “ab”, “a”, and “\” by default titled “valid.str”

- Invalid Input string file containing the above strings and the invalid string “c” by default titled “invalid.str”

Identify the test items including their version/revision level. Also specify characteristics of their transmittal media that impact hardware requirements or indicate the need for logical or physical transformations before testing can begin (e.g., programs must be transferred from tape to disk)

# Features to be Tested

The major features of the application available to the user as user input is one of the most error prone areas of this application. This includes the following methods as detailed in the Requirements Specification document:

1. Insert Command
   1. Attempt to insert a valid string
   2. Attempt to insert an invalid string
   3. Attempt to insert a duplicate string
2. Delete Command
   1. Deletion of a valid string index
   2. Invalid attempt to delete a string (user selects 0, -1, length of list + 1, length of list + 2)
3. Set Command
   1. Valid selection of configuration setting
   2. Attempt to select current setting
   3. Attempt to select an invalid setting (user selects 0, -1, “a”, “abcd”)
4. Truncate Command
   1. Valid selection of configuration setting
   2. Attempt to select current setting
   3. Attempt to select an invalid setting (user selects 0, -1, “a”, “abcd”)
5. Run Command
   1. Attempt to run on a valid string selection
      1. Test for string to be accepted by Turing Machine
      2. Test for string to be rejected by Turing Machine
   2. Attempt to run on an invalid string selection (user selects 0, -1, “a”, “abcd”)
   3. Use of command when string is already selected, i.e. not enough transitions took place the first call of the run method for a valid string
6. Quit Command
   1. Prior to ever running on a string
   2. When currently running on a string
   3. Following acceptance of a string
   4. Following rejection of a string

In addition, it will test the output of the Instantaneous Description, and invocation of the application for both valid and invalid definition files and for valid, invalid and absent input string files.

1. Valid Def File
   1. Valid string file
      1. No errors, no mention of string file loaded
   2. Invalid string file
      1. No errors on .def loading, mentions strings that are not valid for the Turing Machine
   3. No string file
      1. No errors, no mention of string file loaded
2. Invalid Def File
   1. Valid string file
      1. Errors loading the Turing machine, does not execute program
   2. Invalid string file
      1. Errors loading the Turing machine, does not execute program
   3. No string file
      1. Errors loading the Turing machine, does not execute program

Test cases are detailed in the Test Cases document accompanying this test plan.

Identify all software features and combinations of software features to be tested. Identify the test design specification associated with each feature and each combination of features.

# Features Not to be Tested

Features not to be tested in this test plan is the Unit Testing for the application. This is due to the product being accepted as complete by Neil Corrigan and so for this first test plan, we are assuming the specific methods are correct and any defects found due to testing the updated requirements are to be noted for the next test plan.

Identify all features and significant combinations of features that are in the requirements document that will not be tested and the reasons.

# Approach

For this test plan, we will take the approach of fault model testing to test for bugs in areas we believe are most prone to defects. As a group, we decided that this area was going to be the areas centered around user input.

For this test plan, all testing will take place in a manual fashion as requested by Kary Cook. This means that no Test Drivers or automated testing will take place.

For this test plan, all testing will take place with actual files detailed above in the Test Items section, in order to simulate an actual scenario. Test environment will be Linux, Windows and Mac platforms, though this should not be a major focus for the testing as the application was developed without platform specific requirements.

Describe the overall approach to testing: extent of verification of requirements, use of test methods, use of test input files, simulated versus actual use scenarios, differences between test environment and actual use environment,

# Item Pass/Fail Criteria (optional based on risk level)

Specify the criteria to be used to determine whether each feature has passed or failed testing.

# Test Deliverables (optional based on risk level)

Minimum deliverables:

* This document
* Actual test results

# Procedure Steps (optional based on risk level)

The steps required to perform the testing. Typically documents in a checklist. See the attached table below as one example.

# Test Log

Provide an entry for each test run including the version of the software tested, date, tester, associated files used or generated.

# Test Summary

Summary of results with references to any unresolved problem reports.