Software Test Cases

**Definition File**

Section 4.2 Tests

4.2.2

Test 1

* Description: User will run the Turing Machine Application with a valid definition file in the same directory as the TMA. This test checks to see if the Turing Machine looks for the definition file in the same directory as the TMA.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: The user should be taken to the “command:” prompt with no errors displayed.

4.2.2.3

Test 2

* Description: User will run the Turing machine with a valid Turing Machine definition file as in Test 1, but with tabs and spaces inserted between the tokens. This test checks to see if the Turing Machine can handle any number of white space characters between tokens as specified in requirement 4.2.2.3.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full valid turing machine definition file with tabs and spaces inserted between tokens of file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: The user should be taken to the “command:” prompt with no errors displayed.

4.2.3

Test 3

* Description: User will run the Turing machine with a valid Turing Machine definition file as in Test 1, but the order of the FINAL\_STATES and TAPE\_ALPHABET headings swapped. This test checks to see if the Turing Machine checking to see if the headings of the Turing Machine are in the correct order.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full valid turing machine definition file with the FINAL\_STATES and TAPE\_ALPHABET headings reversed.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: The user should get an error when the TMA is run on this definition file. The user should not be taken back to the command prompt. The user should be taken back to their terminal.

4.2.4

Test 4

* Description: User will run the Turing machine with a valid Turing Machine definition file as in Test 1, then he or she should run the View Command to see the description of the Turing Machine Definintion. This test checks to see if the Turing Machine Application is retaining the formatting on the description field of the definition file.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full valid turing machine definition file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: The user should be able to load the TMA, definition file, and View command without error. The View command should display the description of the definition exactly how it is formatted in the TM definition file.

4.2.5.1

Test 5

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain a reserved character in the name of one of the states in the “STATES” section. This test checks to see if the Turing Machine is checking for reserved characters in the states.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with a reserved character in one of the states.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that a state contains a reserved character. The user should be taken back to their terminal.

4.2.5.2

Test 6

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain no states listed in the STATES section. The next token in the file after the STATES: heading should be the INPUT\_ALPHABET. This test checks to see if the TMA is checking whether or not the STATES section contains at least one state.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file without any states in the STATES section of the file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that there are no states in the definition file. The user should be taken back to their terminal.

4.2.6.1.1

Test 7

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain a string in the INPUT\_ALPHABET section. This string is any set of more than one characters without any white space separating them. This test checks to see whether or not the TMA is checking that the INPUT\_ALPHABET only contains characters and no strings.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file containing a string in the INPUT\_ALPHABET section. This string is any set of more than one characters without any white space separating them.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that the INPUT\_ALPHABET contains a string. The user should be taken back to their terminal.

4.2.6.3

Test 8

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain no characters listed in the INPUT\_ALPHABET section. The next token in the file after the INPUT\_ALPHABET: heading should be the TAPE\_ALPHABET. This test checks to see if the TMA is checking whether or not the INPUT\_ALPHABET section contains at least one character.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file without any characters in the INPUT\_ALPHABET section of the file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that there are no characters in the input alphabet of the definition file. The user should be taken back to their terminal.

4.2.6.2

Test 9

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain a reserved character in the name of one of the states in the “INPUT\_ALPHABET” section. This test checks to see if the Turing Machine is checking for reserved characters in the input alphabet.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with a reserved character in one of the input alphabet.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that an input alphabet character is a reserved character. The user should be taken back to their terminal.

4.2.7.1

Test 10

* Description: User will run the Turing machine with a valid Turing Machine definition file as in Test 1. The definition file should contain the blank character in the TAPE\_ALPHABET section. This test checks to see if the Turing Machine Application will accept the blank character, which is a reserved character, as one the characters in the TAPE\_ALPHABET.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with the blank character in the TAPE\_ALPHABET section.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: The user should be taken to the “command:” prompt with no errors displayed.

4.2.7.3

Test 11

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain a reserved character in the name of one of the states in the “TAPE\_ALPHABET” section. This test checks to see if the Turing Machine is checking for reserved characters in the tape alphabet.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with a reserved character in one of the tape alphabet characters.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that a tape alphabet character is a reserved character. The user should be taken back to their terminal.

4.2.7.4

Test 12

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain no characters listed in the TAPE\_ALPHABET section. The next token in the file after the TAPE\_ALPHABET: heading should be the INITIAL\_STATE. This test checks to see if the TMA is checking whether or not the TAPE\_ALPHABET section contains at least one character.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file without any characters in the TAPE\_ALPHABET section of the file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that there are no characters in the tape alphabet of the definition file. The user should be taken back to their terminal.

4.2.8.1

Test 13

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain multiple states in the INITAL\_STATE section. This test checks to see if the TMA allows more than one initial state to be specified in the definition.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with more than one state in the INITIAL\_STATE section.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that multiple states were found in the INITIAL\_STATE section. The user should be taken back to their terminal.

4.2.8.2

Test 14

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The definition file should contain no characters listed in the INITIAL\_STATE section. The next token in the file after the INITIAL\_STATE: heading should be the BLANK\_CHARACTER. This test checks to see if the TMA is checking whether or not the INITAL\_STATE section contains at least one state.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file without any states in the INITIAL\_STATE section of the file.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that there are no states in the INITIAL\_STATE section of the definition file. The user should be taken back to their terminal.

4.2.9.2

Test 15

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The BLANK\_CHARACTER section should be empty. The next token after the BLANK\_CHARACTER heading in the file should be the FINAL\_STATES heading. The purpose of this test is to check to see if the TMA checks to see if the blank character section is empty.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with an empty blank character section.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that the blank character section is empty. The user should be taken back to their terminal.

4.2.10.1

Test 16

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The FINAL\_STATES section should contain a state name containing a character that is a reserved character. The purpose of this test is to check whether or not the TMA checks to see if the FINAL\_STATES section contains reserved characters.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with an a final state containing a reserved character.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that a final state contains a reserved character. The user should be taken back to their terminal.

4.2.10.2

Test 17

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The FINAL\_STATES section should contain the name of a state that is not listed in the STATES section of the TM definition file. The purpose of this test is to check to see if the TMA allows states that have not been defined in the STATES section.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with an a final state that is not in the states section of the TM definition.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that a final state is not a state defined in the states of the definition. The user should be taken back to their terminal.

4.2.10.3

Test 18

* Description: User will run the Turing machine with an invalid Turing Machine definition file as in Test 1. The file should contain a heading after the FINAL\_STATES section. The purpose of this test is to check to see if the TMA allows sections after the FINAL\_STATES section. The FINAL\_STATES section should be the last section in the file.
* Component
  + Type: class
  + Name: TuringMachine
* Input Condition: A full turing machine definition file with a section after the FINAL\_STATES section.
* Input State or Configuration: The turing machine application is run with the name of the definition file without extension as a command line argument.
* Expected Result: An error should be displayed telling the user that the FINAL\_STATES section is not the last section in the file, as it should be. The user should be taken back to their terminal.