Matthew Kennedy
CS 5800
Testing Document
Due by 4/7/2021

Overview
For this program, there are several sections of tests to be run. Each test will have an intended functionality targeted, and will showcase the method in which this functionality was tested. The sections to be targeted for testing are:

1. Input tab
    a. States input
    b. Alphabet input
    c. Transition function input
    d. Final states input
    e. File import
    f. Convert Button
    g. Exit Button
2. Simulator tab
    a. String to Process Input
    b. Graphviz output
    c. Minimized DFA output
    d. Exit button

States input:

Test: 0
Target: GUI States Input
Functionality to test: Verify normal input is accepted
Input: q0, q1, q2
Pass/Fail: Pass

Test: 1
Target: GUI States Input
Functionality to test: Put in various whitespace amounts, verify whitespace is trimmed
Input: q0, q1,q2
Pass/Fail: Pass (result: states are q0, q1, q2

Test: 2
Target: GUI States Input
Functionality to test: Verify all states are unique
Input: q0, q1, q1
Pass/Fail: Pass: program catches duplicate states, prompts user for re-input

Test: 3
Target: GUI States Input
Functionality to test: After states are provided, populate the starting state options with current states
Input: q0, q1, q1
Pass/Fail: Pass: Program repopulates the drop-down box with new options as the states are updated

Test: 4
Target: GUI States Input
Functionality to test: User attempts to enter special characters, such as . or !
Input: q0., q1, q1
Pass/Fail: Pass: Program has validator that prevents this input

Test: 5
Target: GUI States Input
Functionality to test: User enters special characters with file input
Input: q0., q1, q1)
Pass/Fail: Pass: Program detects special characters, prompts user to re-enter states

Test: 6
Target: GUI States Input
Functionality to test: User enters nothing between commas
Input: q0,,q1
Pass/Fail: Pass: Program detects empty state, prompts user for re-entry

Alphabet input

Test: 0
Target: GUI Alphabet Input
Functionality to test: Verify normal input is accepted
Input: a, b, c
Pass/Fail: Pass

Test: 1
Target: GUI Alphabet Input

Functionality to test: Put in various whitespace amounts, verify whitespace is trimmed
Input: a, b,c
Pass/Fail: Pass (result: alphabet is a, b, c)

Test: 2
Target: GUI Alphabet Input
Functionality to test: Verify alphabet input is unique from states input
Input: a, b, q0 (assuming q0 is a state)
Pass/Fail: Pass: program catches duplicate state value, prompts user for re-input

Test: 3
Target: GUI Alphabet Input
Functionality to test: Verify alphabet input is unique from states input
Input: a, b, b
Pass/Fail: Pass: program catches duplicate alphabet items, prompts user for re-input

Test: 4
Target: GUI Alphabet Input
Functionality to test: User attempts to enter special characters, such as . or !
Input: a., b, c!
Pass/Fail: Pass: Program has validator that prevents this input

Test: 5
Target: GUI Alphabet Input
Functionality to test: User enters special characters with file input
Input: a., b, c)
Pass/Fail: Pass: Program detects special characters, prompts user to re-enter alphabet

Test: 6
Target: GUI Alphabet Input
Functionality to test: User enters nothing between commas
Input: a,,b
Pass/Fail: Pass: Program detects empty state, prompts user for re-entry

Transition function input

Test: 0
Target: GUI Transition Function Input
Functionality to test: Verify normal input is accepted
Input: (q0, a, q0), (q1, a, q1)

Pass/Fail: Pass

Test: 1
Target: GUI Transition Function Input
Functionality to test: Put in various whitespace amounts, verify whitespace is trimmed
Input: (q0, a, q0) , (q1, a, q1)
Pass/Fail: Pass (result: transition function is {(q0, a): q0, (q1, a): q1}

Test: 2
Target: GUI Transition Function Input
Functionality to test: Put varying amounts of '(' and ')'
Input: ((((q0, a, q1))
Pass/Fail: Pass (result: transition function is {(q0, a): q0}

Test: 3
Target: GUI Transition Function Input
Functionality to test: For each transition element, verify starting state is part of states provided
Input: states are q0, q1, input for transition function is (q0, a, q0), (q2, a, q0)
Pass/Fail: Pass: program catches wrong start state, prompts user for re-input

Test:4
Target: GUI Transition Function Input
Functionality to test: For each transition element, verify ending states are part of states provided
Input: states are q0, q1, q2, input for transition function is (q0, a, q0), (q2, a, q0, q3)
Pass/Fail: Pass: program catches wrong ending state, prompts user for re-input

Test:5
Target: GUI Transition Function Input
Functionality to test: For each transition element, verify strings to process are part of alphabet provided
Input: alphabet is a, b, input for transition function is (q0, a, q0), (q2, c, q0, q3)
Pass/Fail: Pass: program catches wrong string to process, prompts user for re-input

Final states input

Test: 0
Target: GUI Final states Input
Functionality to test: Verify normal input is accepted
Input: q0, q1, q2

Pass/Fail: Pass

Test: 1
Target: GUI Final states Input
Functionality to test: Put in various whitespace amounts, verify whitespace is trimmed
Input: q0, q1,q2
Pass/Fail: Pass (result: final states are a, b, c)

Test: 2
Target: GUI Final states Input
Functionality to test: Verify all states provided are part of provided Q
Input: states are q0, q1, input to final states is q0, q1, q2
Pass/Fail: Pass: program catches erroneous final state, prompts user for re-input

File import

Test: 0
Target: File input
Functionality to test: Verify file dialog appears, allows user to only select text documents
Input: N/A
Pass/Fail: Pass

Test: 1
Target: File input
Functionality to test: Verify file path is recorded by program, and is openable. If not, program does not attempt to process file
Input: N/A
Pass/Fail: Pass

Test: 2
Target: File input
Functionality to test: After opened, file will go line-by-line, cleaning the whitespace from each line, and if the line starts with a tuple element (Q, S, D, F, q0), then it will read this line in as input. Lastly, the program will verify all inputs after reading lines.
Input: N/A
Pass/Fail: Pass: program reads in lines as intended

Test: 3
Target: File input
Functionality to test: User enters multiple colon's (:) in a given line, instead of just 1 after the tuple element
Input: N/A

Pass/Fail: Pass: program ignores lines with multiple colon's

Test: 4
Target: File input
Functionality to test: User selects a completely random file, not intended to be used with this program
Input: N/A
Pass/Fail: Pass: program validation detects erroneous input, preventing conversion until errors are fixed for whatever was accidentally detected as input

Test: 5
Target: File input
Functionality to test: User selects cancel instead of a file
Input: N/A
Pass/Fail: Pass: program will just return instead of trying to parse file

Verify Button

Test: 0
Target: Verify Button
Functionality to test: If the user presses this button, runs a verification check before attempting to convert
Input: N/A
Pass/Fail: Pass - if there are any outstanding issues, conversion stops until user fixes issue and re-presses the convert button

Test: 1
Target: Verify Button
Functionality to test: If all input is correct, program converts input provided into a DFA, then minimzes the DFA. Lastly, program will populate the simulator tab with minimized DFA information.
Input: N/A
Pass/Fail: Pass

Exit Button (Input tab)

Test: 0
Target: Exit Button (input tab)

Functionality to test: If the user presses this button, the program will exit.
Input: N/A
Pass/Fail: Pass

String to Process Input

Test: 0
Target: String to Process Input
Functionality to test: Verify normal input is accepted
Input: N/A
Pass/Fail: Pass

Test: 1
Target: String to Process Input
Functionality to test: Verify all characters provided are part of Alphabet specified
Input: N/A
Pass/Fail: Pass - if any characters are not part of specified alphabet, user is prompted to re-enter this string

Graphviz Output Button

Test: 0
Target: Graphviz Output Button
Functionality to test: If this button is pressed, output is copied to clipboard for user to paste into graphviz website, only if Pyperclip module is installed.
Input: N/A
Pass/Fail: Pass - if pyperclip module is installed, copies with no issues.

Test: 1
Target: Graphviz Output Button
Functionality to test: If button is pressed and pyperclip not installed, messagebox is display for user to copy text from.
Input: N/A
Pass/Fail: Pass - if pyperclip module is not installed, messagebox is displayed for user to copy text from.

Minimized DFA output

Test: 0
Target: Graphviz Output Button
Functionality to test: After minimized DFA is created, there are 5 fields, similar to the Input tab, that are auto-populated with the new minimized DFA tuple elements
Input: N/A
Pass/Fail: Pass - all fields are auto populated by program after conversion

Exit Button (Simulator tab)

Test: 0
Target: Exit Button (simulator tab)
Functionality to test: If the user presses this button, the program will exit.
Input: N/A
Pass/Fail: Pass