CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

Project Name: JBefunge
People in the group: Siming Zheng
Date: Sep 20, 2017

Introduction

This document will contain test plans, the traceability matrix, and defects report surrounding the program “JBefunge” found in https://github.com/laboon/JBefunge, by the requirements found in https://github.com/laboon/CS1632_Fall2017/blob/master/deliverables/1/requirements.md.

There are several difficulties with testing the requirements. For example, FUN-BEFUNGE states that the program shall be able to run any valid Befunge-93 program. PERF-EXECUTION-TIME states that on any given computer, the system shall be able to complete execution of a reference FizzBuzz implementation (i.e., the one listed in the README.md file of the JBefunge repository), in less than 30 seconds (30,000,000 microseconds). There will be not enough test cases to cover up any valid Beunge-93 program, as well as execution time on every computer.

Difficulties also include preciseness. For example, FUN-RUN-SPEED states that The system shall have three standard execution speeds, Run (no pauses in execution), Walk (50 ms pause after each opcode), and Mosey (500 ms pause after each opcode). All of these shall be reachable by pressing a button of the appropriate name. However, it is difficult to time for exactly 50 ms or 500 ms, given that runtime varies differently every time the program is run.

To develop test cases, I try to break up requirements into different parts. For example, FUN-STOP states that when no program is being executed, the Stop button shall be disabled. When a program is being executed, the Stop button shall be enabled. Upon clicking the Stop button, the currently running program shall stop execution. So I divide them up into cases where the stop button shall be enabled, and cases where it shall be disabled, and test both. So when a defect is found, it can be refer to a more specific test case.

There is also concern in how the program handles infinite loop or invalid opcode and how those affect the test cases. They may not be in the requirement but may cause unexpected behavior.

Test Cases

IDENTIFIER: TEST-FUN-TEXT-DISPLAY-FEATURES
TEST CASE: Test if a graphical user interface is displayed, and within the three textboxes, labeled Program Area, Stack and Output, are displayed upon startup.
PRECONDITIONS: The JBefunge program is ready to be run.
EXECUTION STEPS: Start the program. Look for the features in the requirements.
POSTCONDITIONS: A graphical user interface should be displayed. Within, there should be three textboxes. One labeled Program area, and the other two Stack and Output.

IDENTIFIER: TEST-FUN-TEXT-DISPLAY-EDITABLE
TEST CASE: Test if the program area textbox is user-editable, and the other textboxes are never user-editable, when no code is running.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: When no code is running, check(click to see) if the program area text box is user-editable, and check(click to see) if the stack and output are not user-editable.
POSTCONDITIONS: The program area textbox shall be user-editable, while the stack and output are not user-editable.

IDENTIFIER: TEST-FUN-TEXT-DISPLAY-EDITABLE-RUNNING
TEST CASE: Test if the program area textbox is user-editable, and the other textboxes are never user-editable, when code is running.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: Run any code(ex. FizzBuzz), while running, check(click to see) if the program area text box is user-editable, and check(click to see) if the stack and output are not user-editable.
POSTCONDITIONS: The program area textbox shall be user-editable, while the stack and output are not user-editable.

IDENTIFIER: TEST-FUN-MENUS-MENU-GROUPS
TEST CASE: Test if three menu groups File, Color, and Options are displayed upon start up.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: Look for the menu group, check if File, Color and Options are present.
POSTCONDITIONS: File, Color and Options are present in the menu group.

IDENTIFIER: TEST-FUN-MENUS-MENU-ITEMS
TEST CASE: Test if Open File, Save File, Save As, and Quit are under File menu.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: Click on File menu, look if Open File, Save File, Save As, and Quit are present.
POSTCONDITIONS: Open File, Save File, Save As, and Quit are present under File.

IDENTIFIER: TEST-FUN-MENUS-MENU-COLOR
TEST CASE: Test if Red, Yellow, Blue, Pink, Green, and Orange are under Color.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: Click on Color in the menu, look if Red, Yellow, Blue, Pink, Green, and Orange are present.
POSTCONDITIONS: Red, Yellow, Blue, Pink, Green, and Orange are present under Color.

IDENTIFIER: TEST-FUN-MENUS-MENU-OPTIONS
TEST CASE: Test if Time Program, and Check for End Opcode are under Options.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS: Click on Options in the menu, look if Time Program, and Check for End Opcode are present.
POSTCONDITIONS: Time Program, and Check for End Opcode are present under Options.

IDENTIFIER: TEST-FUN-BEFUNGE-RUN
TEST CASE: Test if the system is able to run any valid Befunge-93 program. (In this case, only certain amount of sample programs)
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Click on each of the Run, Walk, Mousey, Step to execute the program.
3. Check if the program runs without unexpected errors or crashes.

POSTCONDITIONS: The system shall be able to run any valid Befunge-93 program without unexpected behaviors.

IDENTIFIER: TEST-FUN-BEFUNGE-VALID-OUTPUT
TEST CASE: Test if the system is able to run any valid Befunge-93 program, displaying valid stack data and output according to the Befunge-93 specification here: http://catseye.tc/view/befunge-93/doc/Befunge-93.markdown. (In this case, only certain amount of sample programs)
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Click on each of the Run, Walk, Mousey, Step to execute the program.
3. Open up http://catseye.tc/view/befunge-93/doc/Befunge-93.markdown
4. Check if the stack data is appropriate according to specifications in the website
5. Check if the output is appropriate according to specifications in the website

POSTCONDITIONS: The stack data and output after the program stops running is valid according to the Befunge-93 specification in http://catseye.tc/view/befunge-93/doc/Befunge-93.markdown

IDENTIFIER: TEST-FUN-RUN-SPEED-RUN
TEST CASE:

1. Test if the system contains Run as one of the execution speeds.
2. Test if the Run option has no pauses in execution.

PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Check if the Run option is present.
3. Click on Run to execute the program.
4. Observe the execution.

POSTCONDITIONS: The Run option is present, and there is no pause in the execution(not observable by human eyes).

IDENTIFIER: TEST-FUN-RUN-SPEED-WALK
TEST CASE:

1. Test if the system contains Walk as one of the execution speeds.
2. Test if the Walk option has 50 ms pause after each opcode.

PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Check if the Walk option is present.
3. Click on Run to execute the program. Either use Time Program or other ways to record the runtime.
4. Click on Walk to execute the program. Either use Time Program or other ways to record the runtime.
5. Assuming that Run and Time Program work fine. Use the runtime in Walk minus the runtime in Run, then divide by the number of opcode.
6. If the steps above do not work, try timing the gap between just one opcode.

POSTCONDITIONS: The Walk option is present, and there is 50 ms pause after each opcode. It should be slower than Run but faster than Mosey.

IDENTIFIER: TEST-FUN-RUN-SPEED-MOSEY
TEST CASE:

1. Test if the system contains Mosey as one of the execution speeds.
2. Test if the Mosey option has 500 ms pause after each opcode.

PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Check if the Mosey option is present.
3. Click on Run to execute the program. Either use Time Program or other ways to record the runtime.
4. Click on Mosey to execute the program. Either use Time Program or other ways to record the runtime.
5. Assuming that Run and Time Program work fine. Use the runtime in Mosey minus the runtime in Run, then divide by the number of opcode.
6. If the steps above do not work, try timing the gap between just one opcode.

POSTCONDITIONS: The Mosey option is present, and there is 500 ms pause after each opcode. It should be slower than both Run and Walk.

IDENTIFIER: TEST-FUN-STEP
TEST CASE: Test if

1. The Step mode is present.
2. The Step mode allows user to step one opcode at a time.
3. The stack and output are properly updated.

PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Check if the Step option is present.

3. Click on the Step option from start to the end of the program(@)(assuming that the program ends in a reasonable amount of time).
4. Observe cursor, as well as the stack and output.

POSTCONDITIONS: Until the end of the code, user is able to step one opcode at a time, and the stack and output are properly updated.

IDENTIFIER: TEST-FUN-STOP-ENABLED
TEST CASE: Test if:

1. When program is running, the stop button is enabled.
2. After clicking the stop button, the current running program stops.

PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Execute the program with any mode.
3. Before the program finishes running:
    1. See if the stop button is enabled.
    2. If the stop button is enabled, click to stop the running program.

POSTCONDITIONS: The stop button shall be enabled during running, and shall stop the funning program, as well as become disabled after clicking.

IDENTIFIER: TEST-FUN-STOP-DISABLED
TEST CASE: Test if:When program is not running, the stop button is disabled.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. See if the stop button is disabled.
2. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
3. Execute the program with any mode.
4. Assuming that the stop button stops the running program, click on the stop button.
5. If the running program stops, see if the stop button is also disabled.

POSTCONDITIONS: The stop button is disabled when the program is not running any code.

IDENTIFIER: TEST-FUN-TIME-INFORMED
TEST CASE: Test if, after program execution has completed, and the "Options..Time program" option is checked, the users are informed with the runtime took to execute, in microseconds.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Click on Options in the menu, check the Time program option.
3. Execute the program with any mode.
4. Assuming that the program terminates in reasonable amount of time, when it does, observe the result.

POSTCONDITIONS: A warning should pop up, telling the user the execution time of the program, in microseconds.

IDENTIFIER: TEST-FUN-TIME-UNINFORMED

TEST CASE: Test if, after program execution has completed, and the "Options..Time program" option is not checked, the users are not informed with the runtime took to execute.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Click on Options in the menu, uncheck the Time program option.
3. Execute the program with any mode.
4. Assuming that the program terminates in reasonable amount of time, when it does, observe the result.

POSTCONDITIONS: A warning should not pop up, telling the user the execution time of the program, in microseconds.

IDENTIFIER: TEST-FUN-TRACE-DISPLAYED
TEST CASE: Test if a program is being executed, the system displays a cursor on the current opcode indicating that is being executed, immediately after starting execution, using any mode possible.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Run the following steps with Run, Walk, Mosey, and Step:
    1. Start executing the program
    2. Observe the cursor, as well as the stack and output

POSTCONDITIONS: The cursor shall display one opcode at the time from the very start until the end of the execution.

IDENTIFIER: TEST-FUN-TRACE-NON-DISPLAYED
TEST CASE: Test if a program is not being executed, the system does not display a cursor on the current opcode indicating that is being executed, in any mode possible.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. See if the cursor displays before any code is executed.
2. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
3. Run the following steps with Run, Walk, Mosey, and Step:
    1. Start executing the program
    2. Stop the program/Wait for the termination of the program
    3. See if the cursor disappears

POSTCONDITIONS: The cursor shall not be displayed at the very beginning, or when no code is being executed.

IDENTIFIER: TEST-FUN-CHECK-END-OPCODE-DISPLAYED
TEST CASE:Test if the "Options...Check For End Opcode" option is checked, then the system would warn the user if no end opcode ("@" character) exists in the Program Area.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox, remove the end code(@) in it.
2. Click on Options in the menu, check the Check For End Opcode option.
3. Execute the program with any mode. Observe if warning pops up.

POSTCONDITIONS: The program gives a warning about missing end code.

IDENTIFIER: TEST-FUN-CHECK-END-OPCODE-NON-DISPLAYED
TEST CASE:Test if the "Options...Check For End Opcode" option is checked, then the system would warn the user if no end opcode ("@" character) exists in the Program Area.
PRECONDITIONS: The JBefunge program has started up.
EXECUTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox, with an end code(@) in it.
2. Click on Options in the menu, check the Check For End Opcode option.
3. Execute the program with any mode.
4. Assuming that the program reaches the end opcode in a reasonable amount of time, when it does, observe the result.

POSTCONDITIONS: The program does not give a warning about missing end code.

IDENTIFIER: TEST-PERF-EXECUTION-TIME
TEST CASE: Test if on a user's computer, the system is be able to complete execution of a reference FizzBuzz implementation (i.e., the one listed in the README.md file of the JBefunge repository), in less than 30 seconds (30,000,000 microseconds).
PRECONDITIONS: The JBefunge program has started up. The FizzBuzz program is available.
EXECUTION STEPS:

1. Copy code in FizzBuzz into the Program Area.
2. Check the Time Program option.
3. Execute the program with run mode.
4. When the program ends, observe the time result.

POSTCONDITIONS: The time shall be less than 30,000,000 microseconds.

Traceability Matrix

FUN-TEXT-DISPLAY:
- TEST-FUN-TEXT-DISPLAY-FEATURES
- TEST-FUN-TEXT-DISPLAY-EDITABLE
- TEST-FUN-TEXT-DISPLAY-EDITABLE-RUNNING

FUN-MENUS:
- TEST-FUN-MENUS-MENU-GROUPS
- TEST-FUN-MENUS-MENU-ITEMS
- TEST-FUN-MENUS-MENU-COLOR
- TEST-FUN-MENUS-MENU-OPTIONS

FUN-BEFUNGE:
- TEST-FUN-BEFUNGE-RUN,
- TEST-FUN-BEFUNGE-VALID-OUTPUT

FUN-RUN-SPEED:
- TEST-FUN-RUN-SPEED-RUN
- TEST-FUN-RUN-SPEED-WALK
- TEST-FUN-RUN-SPEED-MOSEY

FUN-STEP:
- TEST-FUN-STEP

FUN-STOP:
- TEST-FUN-STOP-ENABLED
- TEST-FUN-STOP-DISABLED

FUN-TIME:
- TEST-FUN-TIME-INFORMED
- TEST-FUN-TIME-UNINFORMED

FUN-TRACE:
- TEST-FUN-TRACE-DISPLAYED
- TEST-FUN-TRACE-NON-DISPLAYED

FUN-CHECK-END-OPCODE:
- TEST-FUN-CHECK-END-OPCODE-DISPLAYED
- TEST-FUN-CHECK-END-OPCODE-NON-DISPLAYED

PERF-EXECUTION-TIME:
- TEST-PERF-EXECUTION-TIME

Defects Found and Described

SUMMARY: Program Area label not displayed
DESCRIPTION: TEST-FUN-TEXT-DISPLAY-FEATURES test case specifies that upon startup, a textbox labeled Program Area shall be displayed. However, the label “Program Area” is unseen.
REPRODUCTION STEPS: Start Befunge program. Look for the textboxes and their associated labels.
EXPECTED BEHAVIOR: There shall be three textboxes, one labeled “Program Area”, one labeled “Stack”, one labeled “Output”.
OBSERVED BEHAVIOR: The label “Program Area” is missing. Other requirements are met.
SEVERITY: TRIVIAL
IMPACT: Any user who does not know that JBefunge is a text-based code programming software may not be aware that the textbox not labeled is the program area.

SUMMARY: Cursor display is one step behind
DESCRIPTION: TEST-FUN-TRACE-DISPLAYED test case specifies that the system displays a cursor on the current opcode indicating that is being executed, immediately after starting execution. But in Step mode, the cursor starts to display the second time the user presses Step.
REPRODUCTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Click on Step to execute the program, observe the cursor.
3. Click on Step again, observe the cursor.

EXPECTED BEHAVIOR: The cursor displays upon the first time clicking Step.
OBSERVED BEHAVIOR: The cursor displays starting from the second time Step is clicked.
SEVERITY: MINOR
IMPACT: The user may not be able to trace the first step by the cursor, and the problem goes away from the second opcode.

SUMMARY: Cursor still shows when program stops
DESCRIPTION: TEST-FUN-TRACE-NON-DISPLAYED test case specifies that f a program is not being executed, the system does not display a cursor on the current opcode indicating that is being executed. However, when the program stops running, the cursor is still shown at the end opcode(@).
REPRODUCTION STEPS:

1. Open a .bf file(ex. FizzBuzz.bf) or copy code from a valid Befunge-93 program into the Program Area textbox.
2. Execute the program with any mode, run until it stops.
3. Observe the cursor.

EXPECTED BEHAVIOR: The cursor disappears after the program terminates.
OBSERVED BEHAVIOR: The cursor still shows at the end opcode.
SEVERITY: TRIVAL
IMPACT: No real impact except that it violates the requirement.

Enhancement

The program can have a compile option which check if any invalid opcode is present in the Program Area.