**JBefunge Software Quality Assurance Report**

Randyll Bearer and Branden Keck

CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

9/20/2017

**INTRODUCTION**

This report details the test plan for JBefunge, a Java application that allows the user to create and execute Befunge-93 programs and which includes a variety of features that assist the user in these tasks. The initial difficulty we faced while testing the JBefunge program was, naturally, learning enough Befunge-93 to be able to test the program. As mentioned in class, it is difficult to test code when you do not know what the code is meant to be doing. Fortunately Befunge-93 ended up being pretty simple to understand (at least we think we understand it well enough) so the domain barrier was not too difficult to overcome.

Another difficulty we faced while testing JBefunge was determining what defects were actually significant enough to warrant reports. By the end of our testing, the number of defects we found far exceeded the amount we were required to find, but the majority of them were small requirement contradictions or small errors which did not seem to qualify as “defects”. A significant portion of time was spent debating whether something should be labelled as a defect, even if it did not technically go against the requirements, primarily because the requirements were somewhat vague.

As for how we planned our test cases and executed the testing, we tried to take it one requirement at a time from a black-box testing perspective. For each requirement, we would first try and recognize what the ideal case would be, the interior value so to speak, so that we knew the program would at least work in optimal situations. These interior cases tended to make up most of our test plans since it was suggested to keep test cases relatively minimal. Our exploratory testing however was focused entirely on the edge cases and boundary values.

For example, many of the user interface requirements were tested for the “ideal” cases. These requirements include those that necessitate the presence of text boxes, menus, or buttons. For requirements that specified program performance, it was much easier to start with interior cases and work our way towards edge cases. This proved useful for testing the cursor position, identification of a missing end opcode, and general Benfunge-93 standards. It was easy to test that the JBefunge program satisfied these requirements for simple programs and normal operating conditions. However, it was also possible to test that these requirements were satisfied in the case of very extreme operating conditions.

We were able to postulate several test cases for each of the ten listed requirements. Information regarding the preconditions, execution steps, and postconditions for each test case is described in this report. We were able to identify three significant defects in the JBefunge software. The reproduction steps for these defects can also be found in this report. Finally, we have included a traceability matrix and suggested software enhancements.

**TEST CASES**

**IDENTIFIER:** FUN-TEXT-DISPLAY-AREAS

**TEST CASE:** Ensure that, upon startup, the program correctly displays a graphical user interface with three textboxes labeled Program Area, Stack, and Output.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 but not yet ran

**EXECUTION STEPS:**

1. Run the program by entering “java com.laboon.JBefunge” into the command line

2. Observe displayed graphical interface

**POSTCONDITIONS:** The program should continue to run and correctly display three labeled textbox areas: Program Area, Stack, and Output.

**IDENTIFIER:** FUN-TEXT-DISPLAY-EDITABLE

**TEST CASE:** Ensure that out of the three textboxes displayed to the user, only the Program Area textbox is editable by the user.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and has not yet been ran.

**EXECUTION STEPS:**

1. Run the program by entering “java com.laboon.JBefunge” into the command line

2. Left-click in the Program Area textbox, attempt to type “Hello World”

3. Left-click in the Stack textbox, attempt to type “Hello World”

4. Left-click in the Output textbox, attempt to type “Hello World”

**POSTCONDITIONS:** The program should continue to run and display “Hello World” in only the Program Area textbox. Both the Stack and Output textboxes should be empty.

**IDENTIFIER:** FUN-MENUS-SELECTABLE-DROPDOWN

**TEST CASE:** During normal operation, three selectable drop-down menus should be selectable by the user. They should be titled ‘File’, ‘Color’, and ‘Options’.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running.

**EXECUTION STEPS:**

1. Click on the “File” menu button, observe the results.

2. Click on the “Color” menu button, observe the results.

3. Click on the “Options” menu button, observe the results

**POSTCONDITIONS:**

* Upon clicking the the ‘File’ menu, four items should appear: ‘Open File’, ‘Save File’, ‘Save As’, and ‘Quit’
* Upon clicking the ‘Color’ menu six items should appear: ‘Red’, ‘Yellow’, ‘Blue’, ‘Pink’, ‘Green’, and ‘Orange’
* Upon clicking the ‘Options’ menu, two items should appear: ‘Time Program’ and ‘Check for End Opcode’

**IDENTIFIER:** FUN-MENUS-CHECKABLE-ITEMS

**TEST CASE:** During normal operation, the ‘Options’ menu should display two checkable items. If unchecked, clicking them should check the items. If checked, clicking them should uncheck the items.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running.

**EXECUTION STEPS:**

1. Click the ‘Options’ drop-down menu to reveal the ‘Time program’ and ‘Check for end opcode’ checkable items

2. Click the ‘Time program’ item

3. Open the ‘Options’ menu again, then click the ‘Check for end opcode’ item

4. Open the menu a third time and click ‘Time program’ again

5. Open the options menu a final time

**POSTCONDITIONS:** The first time the ‘Options’ menu is clicked neither item should be checked. The second time the menu is opened, only ‘Time program’ should be checked. The third time the menu is opened, both items should be checked. The final time the ‘Options’ menu is opened, only ‘Check for end opcode’ should be checked.

**IDENTIFIER:** FUN-BEFUNGE-VALID-STACK

**TEST CASE:** The program “JBefunge” shall correctly display the current Stack values in the Stack textbox at any given runtime.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter the “FizzBuzz” reference implementation into the Program Area textbox.

2. Press the “Step” button ten times, observe the resultant output in the Stack textbox.

**POSTCONDITIONS:** The Program Counter should be highlighting the “\*” opcode and the Stack textbox should be displaying “[0,100]”. “FizzBuzz” should have initially pushed a 0 onto the stack, then pushed both 10 and 10 to the stack and then multiplied them, resulting in 100.

**IDENTIFIER:** FUN-BEFUNGE-VALID-OUTPUT

**TEST CASE:** Ensure that the program “JBefunge” correctly displays valid output in the Output textbox.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox

2. Press the “Run” button, observe the resultant output in the Output textbox

**POSTCONDITIONS:** Program should continue to run and have the text “20A” in the output textbox.

**IDENTIFIER:** FUN-RUN-SPEED-RUN

**TEST CASE:** Ensure that Befunge-93 code in the Program Area textbox, when the “Run” button is hit, is executed without pauses.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.

2. Under the “Options” menu, ensure that the “Time program” option is checked.

3. Click the “Run” button to execute the code without any pauses.

4. Observe the Highlighted Program Counter as the code executes, notice if there are any obvious pauses between opcodes.

**POSTCONDITIONS:** The program should provide a pop-up message detailing the time to execute the code in microseconds, this time should be less than the recorded time for the “Walk” or “Mosey” options.

**IDENTIFIER:** FUN-RUN-SPEED-WALK

**TEST CASE:** Ensure that Befunge-93 code in the Program Area textbox, when the “Walk” button is hit, is executed with 50ms pauses after each opcode.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Under the “Options” menu, ensure that the “Time program” option is checked.  
3. Click the “Walk” button to execute the code with 50ms pauses.  
4. Observe the Highlighted Program Counter as the code executes, notice if there are any pauses obviously faster or slower than 50ms between opcodes.

**POSTCONDITIONS:** The program should provide a pop-up message detailing the time to execute the code in microseconds. There are 19 opcodes in this Befunge program, so the time to execute the code should be at least 950 milliseconds (950000 microseconds). The code should execute more slowly than “Run”, and more quickly than “Mosey”.

**IDENTIFIER:** FUN-RUN-SPEED-MOSEY

**TEST CASE:** Ensure that Befunge-93 code in the Program Area textbox, when the “Mosey” button is clicked, is executed with 500 ms pauses after each opcode.

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Under the “Options” menu, ensure that the “Time program” option is checked.  
3. Click the “Mosey” button to execute the code with 500ms pauses.  
4. Observe the Highlighted Program Counter as the code executes, notice if there are any pauses obviously faster or slower than 500ms between opcodes.

**POSTCONDITIONS:** The program should provide a pop-up message detailing the time to execute the code in microseconds. There are 19 opcodes in this Befunge program, so the time to execute the code should be at least 9500 milliseconds (9500000 microseconds). The code should execute more slowly than both “Run” and “Walk”.

**IDENTIFIER:** FUN-STEP-EMPTY

**TEST CASE:** Ensure that the program gracefully handles stepping one opcode at a time when there are no opcodes in the Program Area textbox. This is an edge case.

**PRECONDITIONS:** The program “JBefgunge” is compiled using jdk1.8.0\_144 and is currently running with no code in the Program Area textbox.

**EXECUTION STEPS:**

1. Select the “Step” button.

**POSTCONDITIONS:** The program should either disable the “Step” button or simply do nothing.

**IDENTIFIER:** FUN-STEP-NORMAL

**TEST CASE:** For a functional Befunge-93 program, the user should be able to traverse the program one opcode at a time by clicking the “Step” button.

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running. The following program is inserted in the Program Area: 64+"!dlroW ,olleH">:#,\_@

**EXECUTION STEPS:**

1. Press the step button exactly the number of times it takes for the program to reach the terminating opcode  
2. As this is done, observe the Stack and Output windows

**POSTCONDITIONS:** The cursor will move as the user presses the step button. The cursor will move one opcode at a time as the user presses “Step”. If the current opcode is a number (i.e. 6 or 4) or anything between quotes, the appropriate value or ASCII code should appear at the end of the stack. If the current opcode is the comma, the appropriate character should be added to the Output field and that item should be removed from the end of the stack. If the current opcode is the underscore, the appropriate item should be removed from the end of the stack. If the current opcode is the colon, the appropriate item should be duplicated and added to the end of the stack.

**IDENTIFIER:** FUN-STOP-DISABLED

**TEST CASE:** Ensure that the “Stop” button is disabled while no program is being executed.

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Attempt to select the “Stop” button.

**POSTCONDITIONS:** The program should not allow the “Stop” button to be selected.

**IDENTIFIER:** FUN-STOP-ENABLED

**TEST CASE:** Ensure that the “Stop” button is enabled while code is executing and that it properly stops execution of said code.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Select the “Mosey” button to mosey through the Befunge code.  
3. While the Befunge code is executing, attempt to select the “Stop” button.

**POSTCONDITIONS:** The program should have allowed “Stop” to be selected, and the program should have stopped the execution of any Befunge-93 code.

**IDENTIFIER:** FUN-TIME-CHECKED

**TEST CASE:** Ensure that the program displays a “Time to execute in microseconds” message after the execution of any Befunge-93 code when the “Time program” option is checked.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Time program” option is checked under “Options” menu.  
3. Select the “Walk” button.

**POSTCONDITIONS:** Immediately after the Befunge-93 code has completed execution, the program should display a message detailing how long the code took to execute in microseconds.

**IDENTIFIER:** FUN-TIME-UNCHECKED

**TEST CASE:** Ensure that the program doesn’t display a “Time to execute” message after the execution of any Befunge-93 code when the “Time program” option is unchecked.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Time Program” option is not checked under “Options” menu.  
3. Select the “Walk” button.

**POSTCONDITIONS:** The Befunge-93 code should finish executing and no “Time to execute” message should be displayed afterwards.

**IDENTIFIER:** FUN-TIME-UNCHECKED-THEN-CHECKED

**TEST CASE:** Ensure that the program properly displays an accurate “Time to execute” message even when the “Time program” option has not been checked until the code has already begun to execute. This is an edge case.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Time program” option is unchecked under “Options” menu.  
3. Select the “Mosey” button.  
4. While the Befunge-93 code is executing, check the “Time program” option.

**POSTCONDITIONS:** The program should display a “Time to execute” message after the Befunge-93 code has been executed.

**IDENTIFIER:** FUN-TIME-CHECKED-THEN-UNCHECKED

**TEST CASE:** Ensure that the program does not display a “Time to execute” message when the “Time program” option has only been unchecked once the code has already begun to execute. This is an edge case.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Time program” option is checked under “Options” menu.  
3. Select the “Mosey” button.  
4. While the Befunge-93 code is executing, uncheck the “Time program” option.

**POSTCONDITIONS:** The program should not display a “Time to execute” message after the Befunge-93 code has been executed.

**IDENTIFIER:** FUN-TRACE-RUN

**TEST CASE:** Initialize a program using the “Run” command to ensure that a cursor is correctly displayed at this speed (no pauses in execution)

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running. Ensure that the FizzBuzz reference implementation is inserted in the Program Area.

**EXECUTION STEPS:**

1. Press the “Run” button  
2. Observe the position of the cursor until the program reaches the end opcode

**POSTCONDITIONS:** The cursor should move very quickly through the program, but the relative position of the cursor should still be apparent. The program should stop execution at the “@” end opcode and the cursor should disappear.

**IDENTIFIER:** FUN-TRACE-WALK

**TEST CASE:** Initialize a program using the “Walk” command to ensure that a cursor is correctly displayed at this speed (50ms pauses between opcodes)

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running. The following program is inserted in the Program Area: 64+"!dlroW ,olleH">:#,\_@

**EXECUTION STEPS:**

1. Press the “Walk” button  
2. Observe the position of the cursor until the program reaches the end opcode

**POSTCONDITIONS:** The cursor should move through the program at a moderate pace and the position of the cursor should always be apparent. The program should stop execution at the “@” end opcode and the cursor should disappear.

**IDENTIFIER:** FUN-TRACE-MOSEY

**TEST CASE:** Initialize a program using the “Mosey” command to ensure that a cursor is correctly displayed at this speed (500ms pauses between opcodes)

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running. The following program is inserted in the Program Area: 64+"!dlroW ,olleH">:#,\_@

**EXECUTION STEPS:**

1. Press the “Walk” button  
2. Observe the position of the cursor until the program reaches the end opcode

**POSTCONDITIONS:** The cursor should move through the program at a very slow pace and the position of the cursor should always be apparent. The program should stop execution at the “@” end opcode and the cursor should disappear.

**IDENTIFIER:** FUN-TRACE-STEP

**TEST CASE:** Manually traverse a program using the “Step” command to ensure that a cursor is correctly displayed on each opcode in the program

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running. The following program is inserted in the Program Area: 64+"!dlroW ,olleH">:#,\_@

**EXECUTION STEPS:**

1. Continually press the “Step” button to manually advance the cursor once each time the button is pressed.  
2. Press the button until the end opcode is reached.  
3. Observe the position of the cursor the entire time.

**POSTCONDITIONS:** The cursor should appear with each click of the “Step” button beginning with the very first click of this button. The step button is clicked until the program reaches the end opcode. At this time, execution should be stopped and the cursor should disappear.

**IDENTIFIER:** FUN-CHECK-END-OPCODE-CHECKED-REACHABLE-INBOUNDS

**TEST CASE:** Ensure that the program will not display a “No end opcode” warning when an end opcode (@) is reachable and the “check end opcode” option is checked.

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Check for end opcode” option is checked under “Options” menu.  
3. Select “Mosey”.

**POSTCONDITIONS:** The Befunge-93 code should immediately begin executing without a “No end opcode” warning displayed.

**IDENTIFIER:** FUN-CHECK-END-OPCODE-CHECKED-UNREACHABLE-INBOUNDS

**TEST CASE:** Ensure that the program will not display a “No end opcode” warning when an end opcode (@) is unreachable (but still in program area textbox) and the “Check end opcode” option is checked.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Check for end opcode” option is checked under “Options” menu.  
3. Select “Mosey”.

**POSTCONDITIONS:** The Befunge-93 code should immediately begin executing without a “No end opcode found” warning displayed.

**IDENTIFIER:** FUN-CHECK-END-OPCODE-NO-OPCODE

**TEST CASE:** Ensure that the program will display a “No end opcode” warning when no end opcode (@) exists in the Program Area textbox and the “Check end opcode” option is checked.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.  
2. Ensure that the “Check for end opcode” option is checked under “Options menu.  
3. Select “Mosey”

**POSTCONDITIONS:** A “No end opcode found” warning should be displayed before the execution of any Befunge-93 code.

**IDENTIFIER:** FUN-CHECK-END-OPCODE-REACHABLE-OUTOFBOUNDS

**TEST CASE:** Ensure that the program will not display a “No end opcode” warning when the end opcode (@) exists in the Program Area but is not within the bounds of the Befunge-93 torus. This is an edge case.

**PRECONDITIONS:** The program “JBefunge” is compiled using jdk1.8.0\_144 and is currently running with no Befunge-93 code in the Program Area textbox.

**EXECUTION STEPS:**

1. Enter 80 “>” symbols followed by a “@” end opcode into the Program Area textbox  
2. Select “Run”

**POSTCONDITIONS:** The program should not display a “No end opcode” warning and continue to run.

**IDENTIFIER:** PERF-EXECUTION-TIME-MAC-OS10.11

**TEST CASE:** Test the program performance on a Macintosh machine operating with iOS7

**PRECONDITIONS:** The program ‘JBefunge” is compiled using jdk1.8.0\_144 on a Mac OS X 10.11 El Capitan machine with an Intel i7 2.80Ghz CPU with minimal other processes running. The Program Area textbox should be empty.

**EXECUTION STEPS:**

1. Enter the FizzBuzz reference implementation into the Program Area textbox.  
2. Ensure that the “Time program” option is checked under the “Options” menu  
3. Select “Run”.

**POSTCONDITIONS:** A “Time to execute in microseconds” message should be displayed immediately after the Befunge-93 code is finished executing. This time should be less than 30 seconds (30,000,000 microseconds).

**IDENTIFIER:** PERF-EXECUTION-TIME-WINDOWS8

**TEST CASE:** Ensure that the “JBefunge” program can complete execution of the FizzBuzz reference implementation on a Windows8 machine.

**PRECONDITIONS:** The “JBefunge” program is compiled using jdk1.8.0\_144 on a Windows8 machine with an Intel i7-4700MQ 2.40Ghz CPU with minimal other processes running. The Program Area textbox should be empty.

**EXECUTION STEPS:**

1. Enter the FizzBuzz reference implementation into the Program Area textbox.  
2. Ensure that the “Time program” option is checked under the “Options” menu  
3. Select “Run”.

**POSTCONDITIONS:** A “Time to execute in microseconds” message should be displayed immediately after the Befunge-93 code is finished executing. This time should be less than 30 seconds (30,000,000 microseconds).

**IDENTIFIER:** PERF-EXECUTION-TIME-WINDOWS10

**TEST CASE:** Ensure that the “JBefunge” program can complete execution of the FizzBuzz reference implementation on a Windows10 machine.

**PRECONDITIONS:** The “JBefunge” program is compiled using jdk1.8.0\_144 on a Windows10 machine with an Intel i7-4810MQ 2.80Ghz CPU with minimal other processes running. The Program Area textbox should be empty.

**EXECUTION STEPS:**

1. Enter the FizzBuzz reference implementation into the Program Area textbox.  
2. Ensure that the “Time program” option is checked under the “Options” menu  
3. Select “Run”.

**POSTCONDITIONS:** A “Time to execute in microseconds” message should be displayed immediately after the Befunge-93 code is finished executing. This time should be less than 30 seconds (30,000,000 microseconds).

**TRACEABILITY MATRIX**

|  |  |
| --- | --- |
| FUN-TEXT-DISPLAY | FUN-TEXT-DISPLAY-AREAS  FUN-TEXT-DISPLAY-EDITABLE |
| FUN-MENUS | FUN-MENUS-SELECTABLE-DROPDOWN  FUN-MENUS-CHECKABLE-ITEMS |
| FUN-BEFUNGE | FUN-BEFUNGE-VALID-STACK  FUN-BEFUNGE-VALID-OUTPUT |
| FUN-RUN-SPEED | FUN-RUN-SPEED-RUN  FUN-RUN-SPEED-WALK  FUN-RUN-SPEED-MOSEY |
| FUN-STEP | FUN-STEP-EMPTY  FUN-STEP-NORMAL |
| FUN-STOP | FUN-STOP-DISABLED  FUN-STOP-ENABLED |
| FUN-TIME | FUN-TIME-CHECKED  FUN-TIME-UNCHECKED  FUN-TIME-UNCHECKED-THEN-CHECKED  FUN-TIME-CHECKED-THEN-UNCHECKED |
| FUN-TRACE | FUN-TRACE-RUN  FUN-TRACE-WALK  FUN-TRACE-MOSEY  FUN-TRACE-STEP |
| FUN-CHECK-END-OPCODE | FUN-CHECK-END-OPCODE-CHECKED-REACHABLE-INBOUNDS  FUN-CHECK-END-OPCODE-CHECKED-UNREACHABLE-INBOUNDS  FUN-CHECK-END-OPCODE-NO-OPCODE  FUN-CHECK-END-OPCODE-CHECKED-REACHABLE-OUTOFBOUNDS |
| PERF-EXECUTION-TIME | PERF-EXECUTION-TIME-MAC-OS10.11  PERF-EXECUTION-TIME-WINDOWS8  PERF-EXECUTION-TIME-WINDOWS10 |

**DEFECTS**

**SUMMARY:** Program Area textbox is not labeled.

**DESCRIPTION:** As per the FUN-TEXT-DISPLAY requirement, the three textboxes should be labeled “Program Area”, “Stack”, and “Output”. However, the first textbox is not labeled. This defect was found by the FUN-TEXT-DISPLAY-AREAS test case.

**REPRODUCTION STEPS:**

1. Compile the “JBefunge” program on jdk1.8.0\_144 on a Windows10 machine using this command: “javac ./com/laboon/\*.java”

2. Run the program through the command line using: “java com.laboon.JBefunge”

3. Observe the lack of a label on the uppermost textbox.

**EXPECTED BEHAVIOR:** The uppermost textbox should be labeled “Program Area”

**OBSERVED BEHAVIOR:** The uppermost textbox is not labeled whatsoever.

**SEVERITY:** Major: not performance critical, but very unprofessional/unnecessarily confusing.

**IMPACT:** The user will not see a label for the Program Area textbox, may not understand where to place code.

**SUMMARY:** PC Highlighted Cursor does not display on first opcode when “Step” is selected.

**DESCRIPTION:** As per the FUN-TRACE requirement, the program should highlight the current executing opcode starting from the very first opcode. However, when the “Step” button is selected with code in the Program Area textbox the very first opcode is not highlighted. This defect was found by the FUN-TRACE-STEP test case.

**REPRODUCTION STEPS:**

1. Compile the “JBefunge” program on jdk1.8.0\_144 on a Windows10 machine using this command: “javac ./com/laboon/\*.java”

2. Run the program through the command line using: “java com.laboon.JBefunge”

3. Enter “091+091++.91+6\*5+,@” into the Program Area textbox.

4. Select the “Step” button.

5. Observe the unhighlighted first “0” opcode.

**EXPECTED BEHAVIOR:** The first “0” opcode should be highlighted the color of whichever color is selected under the “Color” menu.

**OBSERVED BEHAVIOR:** The first “0” opcode is not highlighted whatsoever.

**SEVERITY:** Minor: a simple test by the user should confirm that the first opcode is in fact being executed.

**IMPACT:** The user will not see a highlighted first opcode. The user may not be aware that the program has begun execution of the first opcode.

**SUMMARY:** “Check for end opcode” option does not check the entire Program Area textbox.

**DESCRIPTION:** As per the FUN-CHECK-END-OPCODE requirement, the program should not display a “No end opcode” warning when the “Check end opcode” option is checked and an end opcode (@) exists in the Program Area textbox. However, if the end opcode is placed outside the bounds of the Befunge-93 torus (80x25 instructions) but still within the Program Area textbox, the program throws a “No end opcode” warning. This defect was found by the FUN-CHECK-END-OPCODE-CHECKED-REACHABLE-OUTOFBOUNDS test case.

**REPRODUCTION STEPS:**

1. Compile the “JBefunge” program on jdk1.8.0\_144 on a Windows10 machine using this command: “javac ./com/laboon/\*.java”

2. Run the program through the command line using: “java com.laboon.JBefunge”

3. Ensure that the “Check end opcode” option is checked.

4. Enter 100 “>” characters followed by a “@” end opcode.

5. Select “Run”

**EXPECTED BEHAVIOR:** The program should not display a “No end opcode” found warning.

**OBSERVED BEHAVIOR:** The program does display a “No end opcode” found warning.

**SEVERITY:** Minor; edge case scenario

**IMPACT:** The user is not warned that their code exceeds the boundaries of the Befunge-93 torus, just that they do not have an opcode even when one exists in the Program Area textbox.

**RECOMMENDED ENHANCEMENTS**

Introduce character limits to the Program Area textbox to better simulate the 80x25 Befunge-93 torus. The “JBefunge” program is still valid without having this enhancement, but it leaves Befunge-93 developers open to accidentally making their programs a little too large and thus invalid. An alternative enhancement would be to display the current grid-location of any opcode so that the developer could verify that the opcode is inbounds.