**CS 1632 – DELIVERABLE 1**: Test Plan and Traceability Matrix

**Project Under Test**: JBefunge

**By**: John Kelly & Nathan Davidson

**Introduction**

Beginning the project my main concern was black box testing with a graphical user interface. Especially because I have not had much experience programming Java GUIs, therefore I am not aware of the many intricacies that errors would hide. I also assumed it would be difficult to determine and exploit edge cases of the GUI because each widget is defined, mostly, by Java libraries – making me think the best place to exploit defects would be the interconnectivity between the widgets.

One of the largest stumbling blocks for me was the size of the defects. Previously, when my projects had defects, they were largely dramatic issues that caused program crashes or obviously incorrect output. Here, this is largely not the case. None of the defects we located have any implication for the primary function of the program – the emulation of Befunge appears to be fully correct.

Instead, the defects we encountered were largely minor. The cursor misbehaves at the start and end of execution. One of the text boxes isn’t labeled. These are definitely defects, but they are much smaller in scope than the ones I’ve previously dealt with.

**Test Cases**

IDENTIFIER: CORRECT-TEXT-DISPLAY (1.1)

TEST CASE: Ensure the program starts a normal graphical user interface.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”

POSTCONDITIONS: The program displays a graphical user interface with three text boxes, labeled Program Area, Stack, and Output.

IDENTIFIER: ALLOW-VALID-EDIT (1.2)

TEST CASE: Ensure the Program Area is editable by the user.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”, click inside the Program Area text box and type “ABCD”

POSTCONDITIONS: Program Area text box displays “ABCD”.

IDENTIFIER: DISALLOW-INVALID-EDIT (1.3)

TEST CASE: Ensure the Sack and Output areas are not editable by the user.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”, click inside the text box labeled Stack and type “ABCD”, then click inside the text box labeled Output and type “ABCD”

POSTCONDITIONS: Neither Stack text box nor Output text box displays any text.

IDENTIFIER: VALID-MENU-GROUPS (2.1)

TEST CASE: Ensure the graphical user interface has three valid menu groups: File, Color, and Options.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”

POSTCONDITIONS: The program displays a graphical user interface with three menu groups: File, Color and Options.

IDENTIFIER: MENU-ITEMS-CHECKABLE (2.2)

TEST CASE: Ensure that relevant submenu items (“Options… Time Program” and “Options… Check for End Opcode”) can be checked.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script. User has opened the Options menu.

EXECUTION STEPS: Click on “Options…. Time Program”. Click on “Options… Check for End Opcode”.

POSTCONDITIONS: Check marks appear in boxes next to “Options… Time Program” and “Options… Check for End Opcode”.

IDENTIFIER: VALID-MENU-ITEMS-FILE (2.3)

TEST CASE: Ensure that the File menu contains the correct options: Open File, Save File, Save As, and Quit.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Click the File menu.

POSTCONDITIONS: Under the File menu, the system shall correctly display four items: Open File, Save File, Save As, and Quit.

IDENTIFIER: VALID-MENU-ITEMS-COLOR (2.4)

TEST CASE: Ensure that the File menu contains the correct options.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Click the Color menu.

POSTCONDITIONS: Under the Color menu, there are six menu items: Red, Yellow, Blue, Pink, Green, and Orange.

IDENTIFIER: VALID-MENU-ITEMS-OPTIONS (2.5)

TEST CASE: Ensure that the Options menu contains the correct elements.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Click the Options menu.

POSTCONDITIONS: Under the Options menu, there are two items Time Program and Check for End Opcode.

IDENTIFIER: VALID-BEFUNGE (3.1)

TEST CASE: Ensure the IDE can run a valid Befunge program.

PRECONDITIONS: User is currently running the JBefunge IDE.

EXECUTION STEPS: Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Run.

POSTCONDITIONS: The program displays the correct output for the FizzBuzz program.

IDENTIFIER: RUN-NO-EXECUTION-PAUSE (4.1)

TEST CASE: Ensure the program does not pause after opcodes in run mode.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press “Run”.

POSTCONDITIONS: The program runs without pausing.

IDENTIFIER: WALK-EXECUTION-PAUSE (4.2)

TEST CASE: Ensure the program pauses after each opcode if Walk is pressed.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Run.

POSTCONDITIONS: The program runs, but pauses after each opcode.

IDENTIFIER: WALK-PAUSE-TEST (4.3)

TEST CASE: Ensure that the program pauses correctly when in Walk mode.

PRECONDITIONS: User has the JBefunge program running.

EXECUTION STEPS: User inputs the following into the Program Area:

@

User selects “Options… Time Program”, and presses Run. Once program terminates, user records time, then presses Walk, waits for the program to terminate, then records the time.

POSTCONDITIONS: Because there is only one opcode, the program should pause exactly once. As a result, the time for the program on Run should be approximately 50ms (50000 microseconds) shorter than the Walk time.

IDENTIFIER: MOSEY-PAUSE-TEST (4.4)

TEST CASE: Ensure that the program pauses correctly when in Mosey mode.

PRECONDITIONS: User has the JBefunge program running.

EXECUTION STEPS: User inputs the following into the Program Area:

@

User selects “Options… Time Program”, and presses Run. Once program terminates, user records time, then presses Mosey, waits for the program to terminate, then records the time.

POSTCONDITIONS: Because there is only one opcode, the program should pause exactly once. As a result, the time for the program on Run should be approximately 500ms (500000 microseconds) shorter than the Mosey time.

IDENTIFIER: ALLOW-STEP (5.1)

TEST CASE: Ensure the IDE iterates through a program one step at a time.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Step until the program is finished executing.

POSTCONDITIONS: The program advances one opcode each time Step is pressed.

IDENTIFIER: STEP-STACK-UPDATE (5.2)

TEST CASE: Ensure IDE updates the stack in Step mode.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Input the following into the Program Area:

4.

Press Step once.

POSTCONDITIONS: The stack displays 4.

IDENTIFIER: STEP-OUTPUT-UPDATE (5.3)

TEST CASE: Ensure IDE updates the stack in Step mode.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Input the following into the Program Area:

4.

Press Step twice.

POSTCONDITIONS: The output field displays 4.

IDENTIFIER: VALID-ACTIVE-STOP (6.1)

TEST CASE: Ensure the Stop button becomes active once a program begins execution.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Mosey. Press Stop once the program begins running.

POSTCONDITIONS: Program is no longer running.

IDENTIFIER: INVALID-INACTIVE-STOP (6.2)

TEST CASE: Ensure the Stop button is no longer active when program terminates (edge case).

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Run. Press Stop once the program terminates.

POSTCONDITIONS: There should be no change in the state of the program and the Stop button shall be inactive.

IDENTIFIER: DISPLAYS-TIME-CHECKED (7.1)

TEST CASE: Ensure that when “Options… Time Program” is checked the system shall display the execution time at program termination.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Ensure “Options… Time Program” is checked. Press Run.

POSTCONDITIONS: The system shall inform the user about the program’s execution time when the program terminates.

IDENTIFIER: NO-DISPLAY-TIME-UNCHECKED (7.2)

TEST CASE: Ensure that when “Options… Time Program” is not checked the system shall not display the execution time at program termination.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Ensure “Options… Time Program” is checked. Press Run.

POSTCONDITIONS: The system shall not inform the user about the program’s execution time when the program terminates.

IDENTIFIER: CHECKED-AT-START-DISPLAY-TIME (7.3)

TEST CASE: Ensure that if “Options… Time Program” is checked at the beginning, then unchecked during execution, execution time information is displayed at program termination.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Ensure “Options… Time Program” is checked. Press Run. Then, as the program is running, ensure “Options… Time Program” is not checked.

POSTCONDITIONS: The system shall inform the user about the program’s run time when the program terminates.

IDENTIFIER: SHOW-CURSOR (8.1)

TEST CASE: Ensure that the system displays a cursor on the opcode being executed immediately upon the start of the program (edge case).

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Step repeatedly.

POSTCONDITIONS: The system shall display a cursor to indicate the opcode being executed.

IDENTIFIER: NO-CURSOR-STOP (8.2)

TEST CASE: Ensure the system displays a cursor if and only if a program is being executed.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Run. Press Stop once the Output text box displays “Fizz”.

POSTCONDITIONS: The system shall not display a cursor.

IDENTIFIER: NO-CURSOR-EXECUTION-FINISHED (8.3)

TEST CASE: Ensure that once execution is finished, the cursor is no longer displayed.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Press Run. Wait for the program to terminate.

POSTCONDITIONS: No cursor is displayed in the program text box.

IDENTIFIER: NO-END-OPCODE-WARN (9.1)

TEST CASE: Ensure the system warns the user that no end opcode exists in the Program Area.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Ensure “Options… Check For End Opcode” is checked, input the following code into the Program Area text box:

64+"!dlroW ,olleH">:#,\_

Click Run.

POSTCONDITIONS: The program shall warn the user that no end opcode is present in the Program Area.

IDENTIFIER: UNREACHABLE-END-OPCODE-NO-WARN (9.2)

TEST CASE: Ensure the system does not warn the user for lack of an end opcode, regardless if it is reachable (corner case).

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Ensure “Options… Check For End Opcode” is checked, input the following code into the Program Area text box:

64+"!dlroW ,olleH">:#,\_vvvv @

Click Run.

POSTCONDITIONS: The program shall not warn the user of a missing end opcode.

IDENTIFIER: END-OPCODE-NO-WARN (9.3)

TEST CASE: Ensure the system dos not warn the user about a non-existent end opcode if there is an end opcode.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Ensure “Options… Check For End Opcode” is checked, and input the following code into the Program Area text box:

64+"!dlroW ,olleH">:#,\_@

Click Run.

POSTCONDITIONS: The program shall not warn the user that no end opcode is present in the Program Area.

IDENTIFIER: DISABLED-NO-WARN (9.4)

TEST CASE: Ensure the system does not warn the user that no end opcode exists in the Program Area if “Options… Check for End Opcode” is unchecked.

PRECONDITIONS: User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Ensure “Options… Check For End Opcode” is unchecked, and input the following code into the Program Area text box:

64+"!dlroW ,olleH">:#,\_

Click Run.

POSTCONDITIONS: The program shall not warn the user that no end opcode is present in the Program Area.

IDENTIFIER: VALID-PERF-EXECUTION-TIME-MAC-SIERRA (10.1)

TEST CASE: Ensure the program executes a reference FizzBuzz implementation in less than 30 seconds.

PRECONDITIONS: Use machine MacBook Pro (MacOS: Sierra, Processor: 3.1 GHz i7, 8 GB RAM) User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Ensure that “Options… Time Program” is checked. Press Run.

POSTCONDITIONS: The program shall display a run time for FizzBuzz not greater than 30,000,000 microseconds (30 seconds).

IDENTIFIER: VALID-PERF-EXECUTION-TIME-DESKTOP (10.2)

TEST CASE: Ensure the program executes a reference FizzBuzz implementation in less than 30 seconds.

PRECONDITIONS: User is on a desktop (i3-6100 3.7GHz GPU, 16 GB RAM). User has a valid JBefunge program, has compiled it by running the “compile.sh” script, and has a Bash terminal open to the directory containing the run.sh script.

EXECUTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Ensure that “Options… Time Program” is checked. Press Run.

POSTCONDITIONS: The program shall display a run time for FizzBuzz not greater than 30,000,000 microseconds (30 seconds).

**Traceability Matrix**

**FUN-TEXT-DISPLAY (3 Tests):** CORRECT-TEXT-DISPLAY, ALLOW-VALID-EDIT, DISALLOW-INVALID-EDIT

**FUN-MENUS (5 Tests):** VALID-MENU-GROUPS, MENU-ITEMS-CHECKABLE, VALID-MENU-ITEMS-FILE, VALID-MENU-ITEMS-COLOR, VALID-MENU-ITEMS-OPTIONS

**FUN-BEFUNGE (1 Test):** VALID-BEFUNGE

**FUN-RUN-SPEED (4 Tests):** RUN-NO-EXECUTION-PAUSE, WALK-EXECUTION-PAUSE, WALK-PAUSE-TEST, MOSEY-PAUSE-TEST

**FUN-STEP (3 Tests):** VALID-STEP, STEP-STACK-UPDATE, STEP-OUTPUT-UPDATE

**FUN-STOP (2 Tests):** VALID-ACTIVE-STOP, INVALID-INACTIVE-STOP

**FUN-TIME (3 Tests):** DISPLAYS-TIME-CHECKED, NO-DISPLAY-TIME-UNCHECKED, CHECKED-AT-START-DISPLAY-TIME

**FUN-TRACE (3 Tests):** SHOW-CURSOR, NO-CURSOR-STOP, NO-CURSOR-EXECUTION-FINISHED

**FUN-CHECK-END-OPCODE (4 Tests):** NO-END-OPCODE-WARN, VALID-UNREACHABLE-END-OPCODE, END-OPCODE-NO-WARN, DISABLED-NO-WARN

**PERF-EXECUTION-TIME (2 Tests):** VALID-PERF-EXECUTION-TIME-MAC-SIERRA, VALID-PERF-EXECUTION-TIME-DESKTOP

**30 Total Tests**

**Defects**

SUMMARY: The system shall display a cursor on the opcode being executed.

DESCRIPTION: The system shall display a cursor on the opcode being executed, but the system does not display a cursor on the first item in the Program Area, although the opcode is added to the stack (VALID-CURSOR-TRACE (8.1)).

REPRODUCTION STEPS Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Then click the Step button to iterate through the opcodes.

EXPECTED BEHAVIOR: The system shall display a cursor on the first opcode of the program during execution.

OBSERVED BEHAVIOR: The system first displays the cursor on the second opcode of the program- not the first.

SEVERITY: Trivial

IMPACT: User does not see a cursor on the first opcode, but the program execution is unaffected.

SUMMARY: The system shall only display the cursor during execution of a program.

DESCRIPTION: The system displays the cursor once execution of a program is stopped, but the cursor remains during the Walk execution (STOP-VALID-CURSOR-TRACE (8.2)).

REPRODUCTION STEPS: Run “./run.sh”. Select “File… Open file”. Navigate to the JBefunge folder and select the file labeled “FizzBuzz.bf”. Then click Walk. Click Stop once the output box displays the “Fizz”.

EXPECTED BEHAVIOR: The system should not display a cursor.

OBSERVED BEHAVIOR: The system displayed a cursor on the opcode “!” of line >>:3%!| >$ ^

SEVERITY: Trivial

IMPACT: User will continue to see cursor, but program execution is unaffected.

SUMMARY: System does not display the label for all text boxes.

DESCRIPTION: The labels for Stack and Output are correct, but the Program Area box is unlabeled (VALID-TEXT-DISPLAY (1.1)).

REPRODUCTION STEPS: Start the program by running “./run.sh” in the appropriate folder after compilation.

EXPECTED BEHAVIOR: All three boxes will be labeled: Program Area, Stack and Output.

OBSERVED BEHAVIOR: There is no label on the top box.

SEVERITY: Minor

IMPACT: User might be confused about where to put the program, but this would be resolved after minimal experimentation, or if they opened a program from a file. Program runs normally.

**Enhancement**

A better set of debugging tools for Befunge would be helpful as scale increases. Specifically, the ability to pause and resume execution (currently, once stopped execution must return to the beginning of the program), and the ability to step backwards through the code. The ability to implement breakpoints in a program is incredibly useful, if not necessary, in any modern IDE. Stepping backwards also allows better examination of the program’s behavior around a failure point.