# JBefunge

## Ben Miller and Colin Spratt

### CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

# This test plan tests the JBefunge IDE for file I/O functionality, program execution time, variable run speed functionality, editor functionality and UI cleanliness by means of black-box testing. Without examining any program code or writing any test programs, we were able to write a test plan that offers fairly complete coverage of the given requirements and expected behavior.

# While writing and testing the JBefunge requirements, our group found it difficult to test correct run-speeds. This is due to the nature of black box testing, leading us to rely solely on averaging displayed runtimes. Gathering enough data points to determine proper behavior proved time-consuming. Enough to prohibit gathering a sufficient sample size to determine accurate averages.

# Our concerns with this test plan lie in the execution of our run-speed tests, specifically the tests associated with the execution speed and minimum run speed requirements. Executing such tests yields inconsistent results based on variables like machine performance and runtime environment. This test plan uses a virtual machine to provide a more reproducible environment, however this solution is time consuming in setup for its minimal amelioration of the problem.

# Despite our test plan's shortcomings, it still provides sufficient coverage of the requirements.

# Test Cases -Ben

IDENTIFIER: TEST-TEXT-DISPLAY  
DESCRIPTION:  
 This test verifies that the GUI displays three text boxes labeled 'Program Area', 'Stack', and 'Output'.  
PRECONDITIONS:  
 JBefunge is compiled as specified by its readme.  
EXECUTION STEPS:   
 1.The user runs JBefunge  
POSTCONDITIONS:   
 The JBefunge GUI displayed, showing three text boxes labeled 'Program Area', 'Stack', and 'Output'respectively.

IDENTIFIER: TEST-UNEDITABLE-PROGRAM-AREA-TEXT-DISPLAY  
DESCRIPTION:   
 This test verifies that the 'Stack' and 'Output' text cannot be edited by the user.   
PRECONDITIONS:   
 JBefunge is running.  
EXECUTION STEPS:   
 1. Click on the middle text box labeled 'Stack'.  
 2. Type the word 'test'.  
 3. Repeat steps 1 and 2 for the lower-most text box labeled 'Output'.  
POSTCONDITIONS:  
 Verify that the word 'test' has not appeared in neither the 'Stack' nor the 'Output' text boxes.

IDENTIFIER: TEST-SAVE-FILE-UPDATE  
DESCRIPTION:  
 This test verifies that saving an updated file applies any changes to an existing file.  
PRECONDITIONS:  
 JBefunge is running with a new, untitled program. The title of the window reads 'UNTITLED' and the program area is empty.  
EXECUTION STEPS:   
 1. Click on the program area text box, and type "test-text".  
 2. Click the file menu and click on "save file".   
 3. Choose a directory (Desktop for example), type a unique file name in the 'File  
 Name:' field (save-test for example), and press the "save" button.  
 4. Clear the "Program Area" text box, and type "new-stuff".  
 5. Repeat step 2.  
 6. Close and reopen JBefunge, the title should read "UNTITLED".  
 7. Click on "open file" under the file menu.  
 8. Navigate to directory where the test file was saved, and locate to saved file.  
 9. Double-click on the file, opening it.  
POSTCONDITIONS:   
 The file saved with a unique file name creates a new file in the specified directory. Opening this file displays "new-stuff" in the program area.

IDENTIFIER: TEST-SAVE-AS-FILE-UPDATE  
DESCRIPTION:  
 This test case verifies that the save as function updates or overwrites a specified.   
PRECONDITIONS:   
 JBefunge is running with a new, untitled program. The title of the window reads 'UNTITLED' and the program area is empty.  
EXECUTION STEPS:   
 1. Click on the program area text box, and type "test-text".  
 2. Click the file menu and click on "save file".   
 3. Choose a directory (Desktop for example), type a unique file name in the 'File  
 Name:' field (save-as-test for example), and press the "save" button.  
 4. Clear the "Program Area" text box, and type "new-stuff".  
 5. Click on "Save as" under the file menu.  
 6. Repeat step 3.  
 7. Close and reopen JBefunge, the title should read "UNTITLED".  
 8. Click on "open file" under the file menu.  
 9. Navigate to directory where the file was saved, and locate the saved file.  
 10. Double-click on the file, opening it.  
POSTCONDITIONS:   
 Opening this file displays "new-stuff" in the program area, and the file name in the title of the window.

IDENTIFIER: TEST-OPEN-FILE-CHANGES-GUI-TITLE  
DESCRIPTION:   
 JBefunge is running with no program selected, the title reads "UNTITLED", and a JBEFUNGE file named "open-test" with a body of "test" already exists.  
PRECONDITIONS:   
 JBefunge is running with no program selected, the title reads "UNTITLED", and a JBEFUNGE file named "open-test" with a body of "test" already exists.  
EXECUTION STEPS:   
 1. Click on "open file" under the file menu.  
 2. Navigate to directory where "open-test" exists, and locate "open-test".  
 3. Double-click on the file, opening it.  
POSTCONDITIONS:   
 The title of the window is changed from "UNTITLED" to "open-test".

IDENTIFIER: TEST-OPEN-FILE-UPDATES-WINDOW-TITLE-FROM-OPENED-FILE  
DESCRIPTION:  
 This test verifies that opening a file while a file already is opened updates the JBefunge window properly.  
PRECONDITIONS:  
 JBefunge is running with some file "open-test-fail" opened. The title reads "open-test-fail", and a JBefunge file named "open-test-success" already exists.  
EXECUTION STEPS:   
 1. Click on "open file" under the file menu.  
 2. Navigate to directory where "open-test-success" exists, and open "open-test-success".  
POSTCONDITIONS:  
 The title of the JBefunge window reads "open-test-success".7

IDENTIFIER: TEST-STEP-STACK-UPDATE  
DESCRIPTION:   
 Verifies that Step updates the Stack appropriately  
PRECONDITIONS:  
 JBefunge is running without a file opened.  
EXECUTION STEPS:   
 1. Enter "1." into the Program Area.  
 2. Click "Step" once.  
POSTCONDITIONS:   
 The Stack area displays "[1]".

IDENTIFIER: TEST-STOP-PROGRAM-END-DISABLED  
DESCRIPTION:   
 Verifies that the Stop button is disabled upon the termination of a program.  
PRECONDITIONS:   
 JBefunge is running and the HelloWorld.bf program exists in the JBefunge directory.  
EXECUTION STEPS:   
 1. Open "HelloWorld.bf".  
 2. Press Run.  
 3. Allow the program to finish executing  
POSTCONDITIONS:   
 After executing the end-of-program symbol (@), the Stop button is disabled.

IDENTIFIER: TEST-STOP-ENABLED-STEP  
DESCRIPTION:   
 Verifies that executing a program by pressing Step enables the Stop button.  
PRECONDITIONS:   
 JBefunge is running and the FizzBuzz.bf program exists in the JBefunge directory.  
EXECUTION STEPS:   
 1. Open "FizzBuzz.bf".  
 2. Press the Step button.  
POSTCONDITIONS:   
 The Stop button is enabled, indicated by the button becoming blue.

IDENTIFIER: TEST-STOP-ENABLED-WALK  
DESCRIPTION:   
 Verifies that executing a program by pressing Walk enables the Stop button.  
PRECONDITIONS:   
 JBefunge is running and the FizzBuzz.bf program exists in the JBefunge directory.  
EXECUTION STEPS:   
 1. Open "FizzBuzz.bf".  
 2. Press the Walk button.  
POSTCONDITIONS:   
 The Stop button is enabled while FizzBuzz is running, indicated by the button becoming blue.

IDENTIFIER: TEST-TRACE-STEP  
DESCRIPTION:   
 Verifies that Step updates the cursor indicating the current program location moves once each Step.  
PRECONDITIONS:  
 JBefunge is running without a file opened.  
EXECUTION STEPS:   
 1. Enter "1." into the Program Area.  
 2. Click Step.  
POSTCONDITIONS:   
 The yellow cursor should appear, highlighting "1" in the Program Area.

IDENTIFIER: TEST-TRACE-WALK  
DESCRIPTION:   
 Verifies that the Cursor appears when the Walk button is pressed.  
PRECONDITIONS:   
 JBefunge is running and the FizzBuzz.bf program exists in the JBefunge directory.  
EXECUTION STEPS:   
 1. Open "FizzBuzz.bf".  
 2. Press the Walk button.  
POSTCONDITIONS:   
 The yellow cursor appears, and moves as the program executes.

# Test Cases -Colin

**IDENTIFIER:** TEST-MENU-FILE

**DESCRIPTION:** This test determines if the File menu populates Open File, Save File, Save As, and Quit.

**PRECONDITIONS:** JBefunge is compiled as specified by its readme.

**EXECUTION STEPS:**

1. Run JBefunge.  
2. Select the File menu item  
3. Observe Open File, Save File, Save As, and Quit listed underneath.

**POSTCONDITIONS:** Open File, Save File, Save As, and Quit are options listed under the File menu.

**IDENTIFIER:** TEST-MENU-OPTIONS

**DESCRIPTION:** This test determines if the Options menu populates Time Program and Check for End Opcode.

**PRECONDITIONS:** JBefunge is running.

**EXECUTION STEPS:**

1. Select the Option menu item.  
2. Observe checkable items Time Program and Check for End Opcode.

**POSTCONDITIONS:** Checkable items Time Program and Check for End Opcode are listed under Options.

**IDENTIFIER:** TEST-WALK-SPEED

**DESCRIPTION:** This test determines if the Walk button executes with a 50 ms pause after each opcode.

**PRECONDITIONS:** JBefunge is running, Time Program has been checked, and HelloWorld.bf has been opened.

**EXECUTION STEPS:**

1. Press the Walk button.  
2. Record its Time to execute.

**POSTCONDITIONS:** The HelloWorld.bf program should Walk around 50x longer than its run speed.

**IDENTIFIER:** TEST-MOSEY-SPEED

**DESCRIPTION:** This test determines if the Mosey button executes with a 500 ms pause after each opcode.

**PRECONDITIONS:** JBefunge is running, Time Program has been checked, and HelloWorld.bf has been opened.

**EXECUTION STEPS:**

1. Press the Mosey button.  
2. Record its Time to execute.

**POSTCONDITIONS:** The HelloWorld.bf program should Mosey around 10x longer than its walk speed.

**IDENTIFIER:** TEST-TIME-ON

**DESCRIPTION:** This test determines if, when Time program is checked, the total time to execute is displayed after running a program.

**PRECONDITIONS:** JBefunge is running.

**EXECUTION STEPS:**

1. Open the Options menu, check the "Time program" checkbox.  
2. Run the included FizzBuzz program.  
3. Ensure the time to execute in microseconds is displayed.

**POSTCONDITIONS:** The correct time to execute is displayed after running FizzBuzz.

**IDENTIFIER:** TEST-TIME-OFF

**DESCRIPTION:** This test determines if, when Time program is not checked, the total time to execute is not displayed after running a program.

**PRECONDITIONS:** JBefunge is running.

**EXECUTION STEPS:**

1. Open the Options menu, ensure the "Time program" checkbox is empty.  
2. Run the included FizzBuzz program.  
3. Ensure the time to execute in microseconds is not displayed.

**POSTCONDITIONS:** No time to execute in microseconds is displayed.

**IDENTIFIER:** TEST-TIME-SWITCH

**DESCRIPTION:** This test is a **Edge Case** that checks for correct execution time if Time Program is selected mid-execution.

**PRECONDITIONS:** JBefunge is running.

**EXECUTION STEPS:**

1. Open the Options menu, ensure the "Time program" checkbox is empty.  
2. Run the included FizzBuzz program.  
3. Before FizzBuzz finishes execution, open the Options menu and press "Time program"  
3. Observe the time to execute.

**POSTCONDITIONS:** Correct execution time is displayed.

**IDENTIFIER:** TEST-BEFUNGE-VALID

**DESCRIPTION:** This test determines if JBefunge can successfully execute a JBefunge-93 program.

**PRECONDITIONS:** JBefunge is running and "HelloWorld.bf" has been opened in the IDE.

**EXECUTION STEPS:**

1. Press the "Run" button.  
2. Wait for execution to finish.

**POSTCONDITIONS:** The stack is empty and "Hello, World!" is printed in the Output field.

**IDENTIFIER:** TEST-BEFUNGE-INVALID

**DESCRIPTION:** This test is an **Edge Case** to determine if JBefunge will run an invalid Java program.

**PRECONDITIONS:** JBefunge is running and a basic Java "Hello, World!" program is entered in the Program Area. (WARNING: Test will continue to execute indefinitely, will need to be interrupted.)

**EXECUTION STEPS:**

1. Press the "Run" button.  
2. Observe the Stack and Output textboxes.

**POSTCONDITIONS:** Program continues to execute indefinitely, with nothing displayed in the Stack or Output textboxes.

**IDENTIFIER:** TEST-100%-CPU-PERF-EXECUTION-TIME

**DESCRIPTION:** This test determines if a computer with a clock speed over 1.3 GHz can run FizzBuzz.bf in under 30 seconds on Run.

**PRECONDITIONS:** JBefunge is running. FizzBuzz.bf has been opened and Time Program has been checked. Computer's clock speed is over 1.3 GHz.

**EXECUTION STEPS:**

1. Press the Run button.  
2. Wait for execution to finish.

**POSTCONDITIONS:** Time to execute is under 30 seconds (or 30,000,000 microseconds)

**IDENTIFIER:** TEST-50%-CPU-PERF-EXECUTION-TIME

**DESCRIPTION:** This test determines if a computer with a clock speed under 1.3 GHz can run FizzBuzz.bf in under 30 seconds on Run.

**PRECONDITIONS:**

1. A Windows 10 Virtual Machine set to 50% Execution Cap is running (simulating a 1.2 GHz Machine).  
2. JBefunge is compiled and Running.  
3. FizzBuzz.bf has been opened in JBefunge.

**EXECUTION STEPS:**

1. Press the Run button.  
2. Wait for execution to finish.

**POSTCONDITIONS:** Time to execute is under 30 seconds (or 30,000,000 microseconds)

# Traceability Matrix

* **FUN-TEXT-DISPLAY**:
  + TEST-TEXT-DISPLAY
  + TEST-EDITABLE-PROGRAM-AREA-TEXT-DISPLAY
  + TEST-UNEDITABLE-PROGRAM-AREA-TEXT-DISPLAY
* **FUN-MENUS**:
  + TEST-MENU-FILE
  + TEST-MENU-COLOR
  + TEST-MENU-OPTIONS
* **FUN-FILE-LOADING**:
  + TEST-SAVE-FILE-CREATION
  + TEST-SAVE-FILE-UPDATE
  + TEST-SAVE-AS-FILE-CREATION
  + TEST-SAVE-AS-FILE-UPDATE
  + TEST-OPEN-FILE-TEXT-LOADS-INTO-PROGRAM-AREA
  + TEST-OPEN-FILE-CHANGES-GUI-TITLE
  + TEST-OPEN-FILE-LOADS-INTO-PROGRAM-AREA-FROM-OPENED-FILE
  + TEST-OPEN-FILE-UPDATES-WINDOW-TITLE-FROM-OPENED-FILE
* **FUN-BEFUNGE**:
  + TEST-BEFUNGE-VALID
  + TEST-BEFUNGE-INVALID
* **FUN-RUN-SPEED**:
  + TEST-RUN-SPEED
  + TEST-WALK-SPEED
  + TEST-MOSEY-SPEED
* **FUN-STEP**:
  + TEST-STEP-OPERATION-MOVEMENT
  + TEST-STEP-STACK-UPDATE
  + TEST-STEP-OUTPUT-UPDATE
* **FUN-STOP**:
  + TEST-STOP-PROGRAM-END-DISABLED
  + TEST-STOP-STOP-DISABLED
  + TEST-STOP-ENABLED-STEP
  + TEST-STOP-ENABLED-MOSEY
  + TEST-STOP-ENABLED-WALK
  + TEST-STOP-ENABLED-RUN
* **FUN-TIME**:
  + TEST-TIME-ON
  + TEST-TIME-OFF
  + TEST-TIME-SWITCH
* **FUN-TRACE**:
  + TEST-TRACE-STEP
  + TEST-TRACE-MOSEY
  + TEST-TRACE-WALK
  + TEST-TRACE-RUN
* **PERF-EXECUTION-TIME**:
  + TEST-100%-CPU-PERF-EXECUTION-TIME
  + TEST-50%-CPU-PERF-EXECUTION-TIME

# Defects

SUMMARY: FUN-OPEN-FILE-DOES-NOT-UPDATE-PROGRAM-AREA  
DESCRIPTION:   
 Running command 'Open File' does not update the text inside of the program-area text box.  
REPRODUCTION STEPS:  
 1. Run JBefunge.  
 2. Type text 'hello world' into the program area.  
 3. Save the file as 'test-1'.  
 4. Clear the program area and use save as 'test-2-no-text'.  
 5. Close and re-launch JBefunge.  
 6. Open 'test-1', which should display 'hello world' in the program area.  
 7. Open 'test-2-no-text'.  
EXPECTED BEHAVIOR: Program area should be empty.  
OBSERVED BEHAVIOR: Program still displays 'hello world'.  
SEVERITY: BLOCKER  
IMPACT: Users will not be able to load files properly.

SUMMARY: PERF-EXECUTUTION-TIME-OVER-30-SECONDS  
DESCRIPTION:   
 UNDER-1.3-PERF-EXECUTION-TIME fails to run FizzBuzz.bf in under 30 seconds  
REPRODUCTION STEPS:  
 1. Initialize a Windows 10 Virtual Machine with a 50% Execution Cap.  
 2. Install JDK and compile JBefunge, then run JBefunge.  
 3. Open File, select FizzBuzz.bf.  
 4. Check the Time Program checkbox.  
 5. Press Run, wait for execution time pop-up.  
EXPECTED BEHAVIOR: Program finishes execution in under 30,000,000 microseconds.  
OBSERVED BEHAVIOR: Program finished execution in 78,968,914 microseconds on the Virtual Machine.  
SEVERITY: Trivial  
IMPACT: Users with under-powered machines will experience longer wait times for executions of programs.