**JBefunge**

Tammy Ma and Connie Suh

CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

**Introduction**

While testing this program, we first reviewed each of the requirements of the program. For each requirement, we identified two or three “sub-requirements” that could each be encompassed by one test case that could sufficiently test that this requirement is met or not met. For certain requirements, such as FUN-RUN-SPEED, we initially had trouble developing a test case for validating the appropriate time between execution steps for “Mosey” and “Walk” speeds. We attempted to measure the time using the “Time Program” option in the program, but the results cannot be easily replicated or measured. As a result, we tested whether the buttons for these speeds function properly (can be clicked by the user and execute code). For FUN-MENUS, we also had to figure out a way to divide this requirement so that the test case chosen did not test for too many subrequirements at once (in the case that only one subrequirement was not met). While at first we chose to test whether all three menu groups included all menu items, we decided to test only if all three menu groups were present, and if the Options group items were checkable.

Another challenge in this assignment was coming up with edge and corner cases to test. This required trying several different inputs/tests to see how the program would respond to potentially “unexpected” edge/corner cases such as editing the code while the code is being executed. Another edge case was discovered accidentally while trying to edit code after the program had been paused in the middle of executing, which led to the Program Counter cursor (indicated by the yellow highlight) to also highlight the subsequent added code. We discovered that coming up with edge/test cases is often a process of trying multiple inputs that may potentially be unexpected.

**Test Cases**

IDENTIFIER: 1

TEST CASE: Determine if all three textboxes are labeled with the correct names, Program Area, Stack, and Output

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Take note that the labels above the textboxes are correct

POSTCONDITIONS: Missing “Program Area” label above the first textbox

IDENTIFIER: 2

TEST CASE: Determine if only the Program Area textbox is user-editable while the other two textboxes are not

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Within the GUI, click in each textbox to check if it is editable

POSTCONDITIONS: Program Area textbox is editable while other textboxes are not

IDENTIFIER: 3

TEST CASE: Determine if there are three menu groups: File, Color, and Options.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Within the GUI, check to see if the three menu groups (File, Color, and Options) are listed in the top left corner of the window

POSTCONDITIONS: The three menu groups (File, Color, and Options) are listed

IDENTIFIER: 4

TEST CASE: Determine if the Time Program and Check for End Opcode menu items under the Options menu group are checkable

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Within the GUI, click under the Options file group and select the Time Program option. See that it is checked.

3. Select the Check for End Opcode option. See that it is checked along with the Time Program option.

POSTCONDITIONS: The Time Program and Check for End Opcode options under the Options menu group are checkable and both can be checked at once.

IDENTIFIER: 5

TEST CASE: Determine if the system behaves appropriately when executing a loop written using Befunge-93 program specifications.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Type the code as shown below:
3. >v>v  
    >^  
   ^ <
4. Click the Step button and follow the loop until the PC passes the starting position

POSTCONDITIONS: PC executes the commands as specified in the Befunge-93 program specifications by following the directions (>, <, v, ^) and treats space as a null command that does nothing.

IDENTIFIER: 6

TEST CASE: Determine if the system appropriately pops a value off the stack when written using Befunge-93 program specifications.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Type the code as shown below:
3. 123.$.@
4. Press “Step” and step through each opcode until the program finishes executing. Read stack and output display.

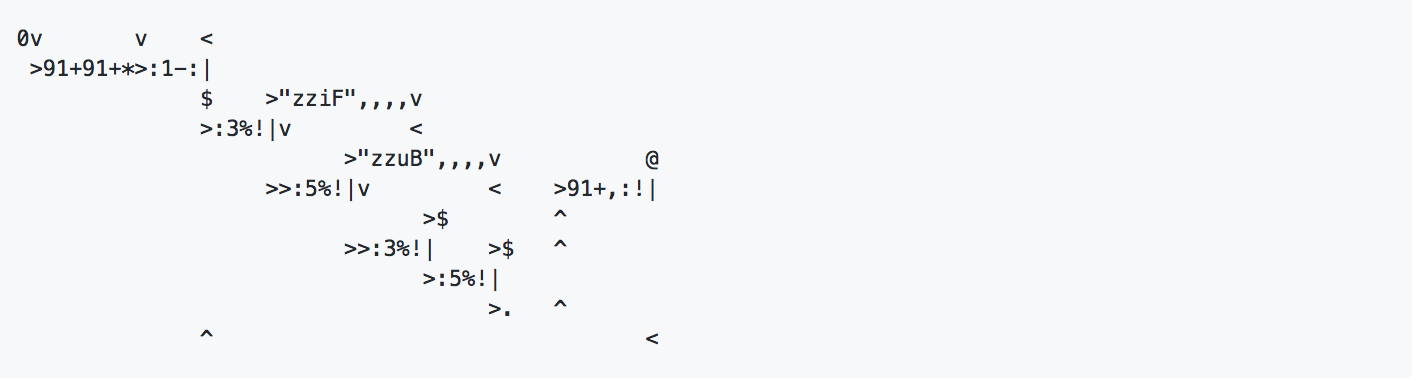
POSTCONDITIONS: Output appears as follows after executing: 31

IDENTIFIER: 7

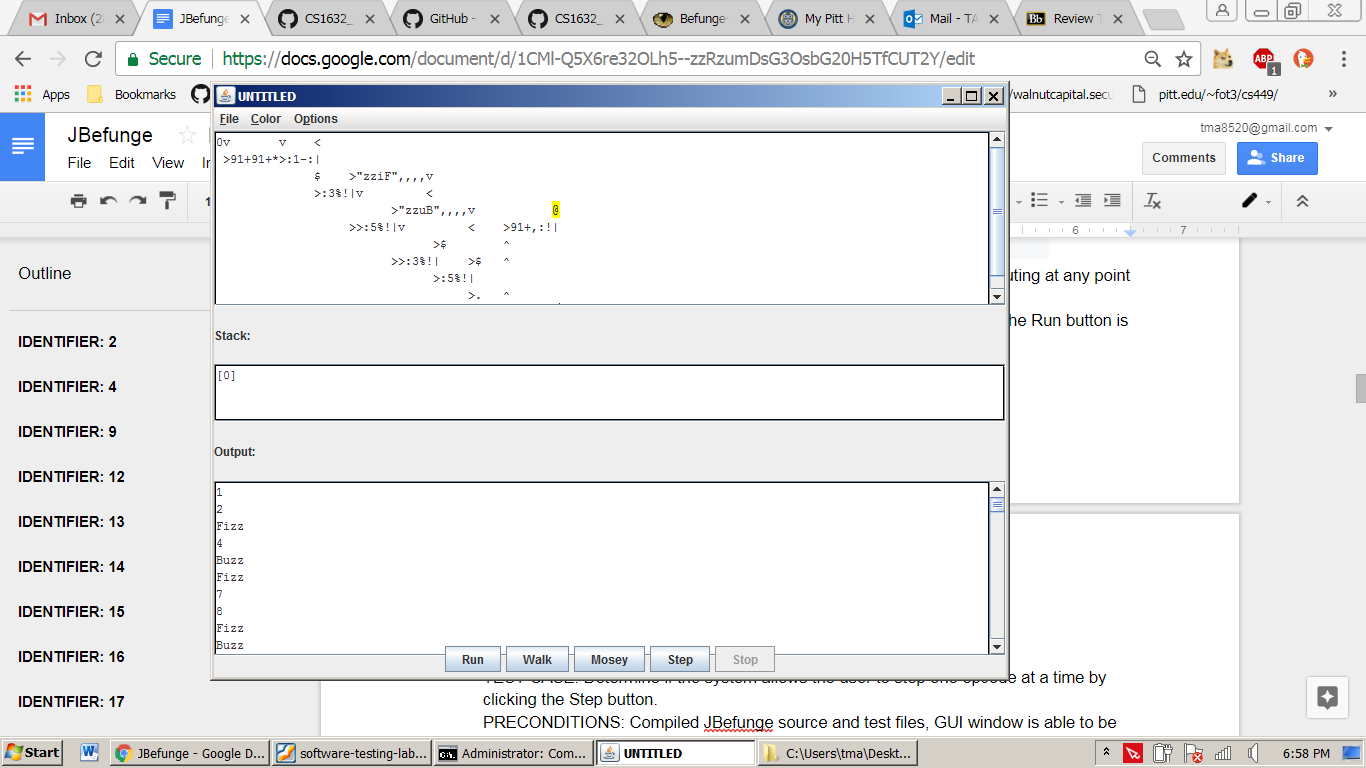
TEST CASE: Determine if all execution speeds (“Run”, “Walk”, and “Mosey”) execute code when pressing a button of the appropriate name.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Copy and paste Fizzbuzz code as shown below:
3. 
4. Click the “Run” button and check that the program is executing. Allow the program to execute completely.
5. Stop the code execution by pressing “Stop”
6. Click the “Walk” button and check that the program is executing. Allow the program to execute completely.
7. Stop the code execution by pressing “Stop”
8. Click the “Mosey” button and check that the program is executing. Allow the program to execute completely.

POSTCONDITIONS: All three speed buttons (“Run”, “Walk”, and “Mosey”) execute the Fizzbuzz code when clicked. Stack and output appear as shown below when finished:



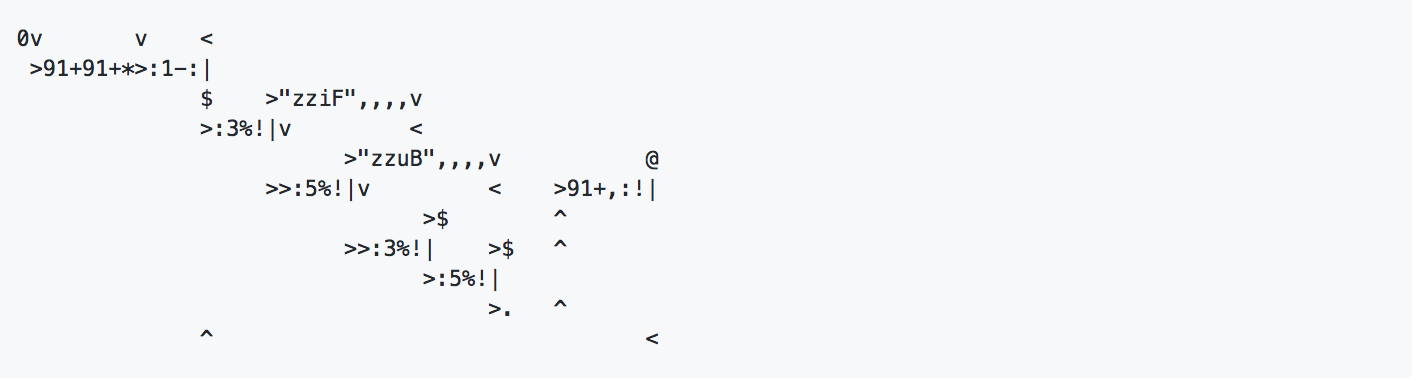
IDENTIFIER: 8

TEST CASE: Determine if the Run execution speed executes the code without any pauses when the “Run” button is clicked

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

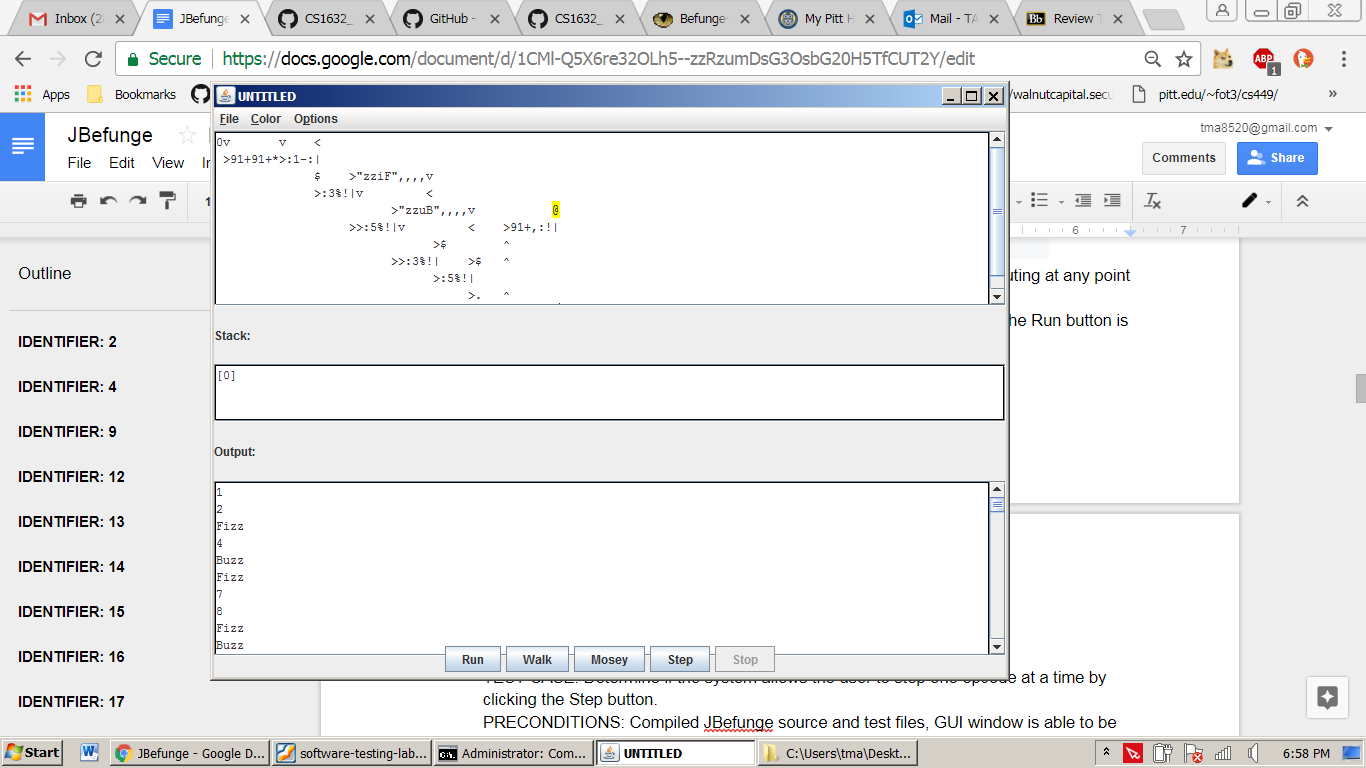
EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Copy and paste the FizzBuzz code shown below



1. Click the Run button and check to see if the program stops executing at any point before the @ (exit) symbol is reached.

POSTCONDITIONS: Program executes the code without pausing after the Run button is pressed. Stack and Output fields appear as shown below after running:



IDENTIFIER: 9

TEST CASE: Determine if the system allows the user to step one opcode at a time by clicking the Step button.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Copy and paste the FizzBuzz code shown below
3. 123.$.@
4. Click the Step button and note location of Program Counter cursor.

POSTCONDITIONS: After 1 step, cursor moves one opcode to the right. Stack/output appears as shown: Stack: [1] No Output

IDENTIFIER: 10

TEST CASE: Determine if the system correctly updates the stack and output when the user clicks the Step button.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program
2. Copy and paste the FizzBuzz code shown below
3. 123.$.@
4. Click the Step button and read output until program finishes executing.

POSTCONDITIONS: After 1 step, stack/output has following output: Stack: [1] No Output. After 2 steps: Stack: [1, 2] No Output. After 3 steps: Stack:[1, 2, 3] No Output. After 4 steps: Stack: [1, 2] No Output. After 5 steps: Stack:[1] Output: 3. After 6 steps: Stack: [] Output: 31

IDENTIFIER: 11

TEST CASE: Determine if the stop button is disabled when no program is being executed.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS: At command line, run “java com.laboon.JBefunge”

POSTCONDITIONS: Program opens and has no inputs and has not executed any code. Stop button is disabled.

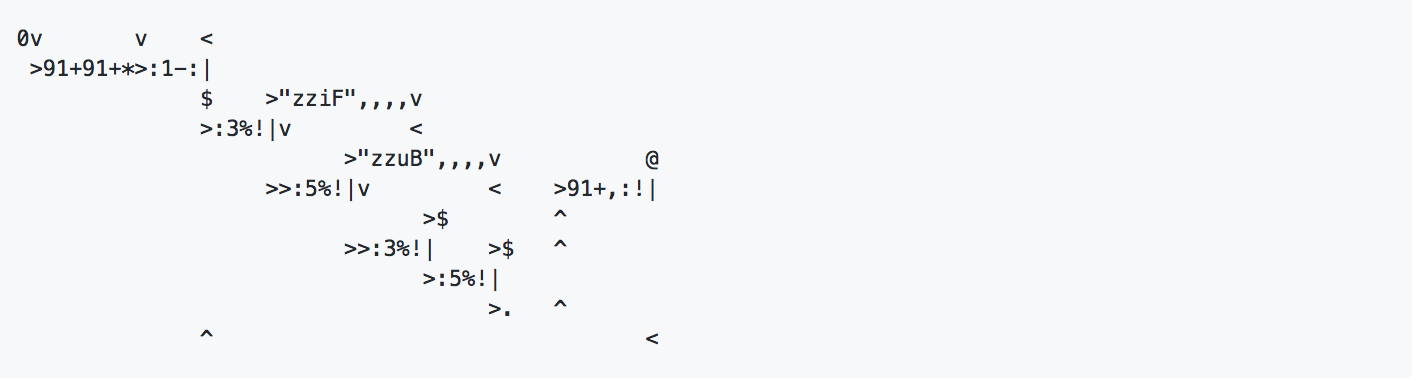
IDENTIFIER: 12

TEST CASE: Determine if when a program is executing code, the Stop button is enabled.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At command line, run “java com.laboon.JBefunge”
2. Enter the following code in the Program Area:



3. Press the “Run” button.

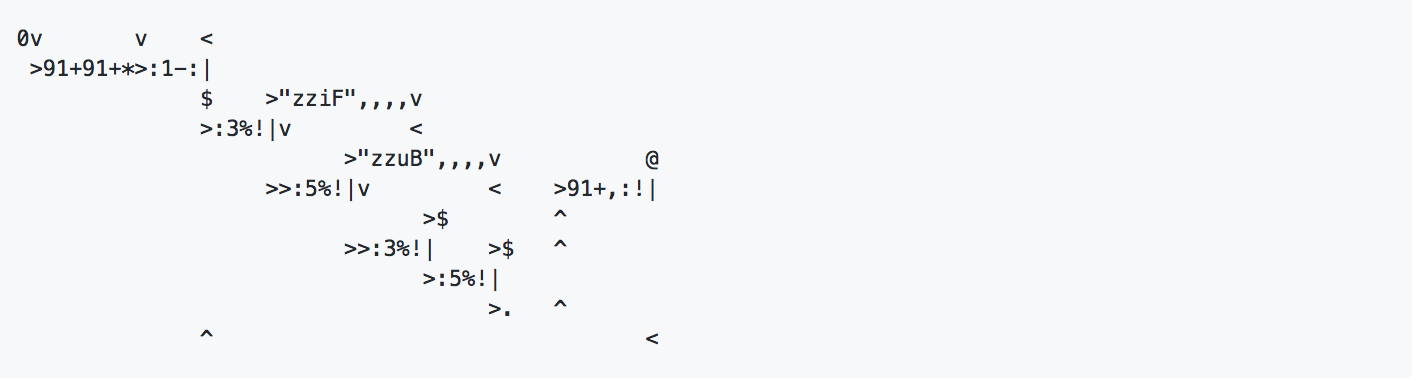
POSTCONDITIONS: When the code is being executed, the “Stop” button is enabled.

IDENTIFIER: 13

TEST CASE: Determine if when the “Options...Time program is checked, after the program execution is finished, user is informed of time in microseconds on how long the program took to execute.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Click on “Options”, located in the Menu group.
3. Select “Time Program”
4. Enter the following code in the Program Area:
5. Click the “Run” button. Allow for program to execute completely.

POSTCONDITIONS: Program displays a message dialogue box showing the amount of time it took to execute, “Time to execute: 6036425 microseconds”.

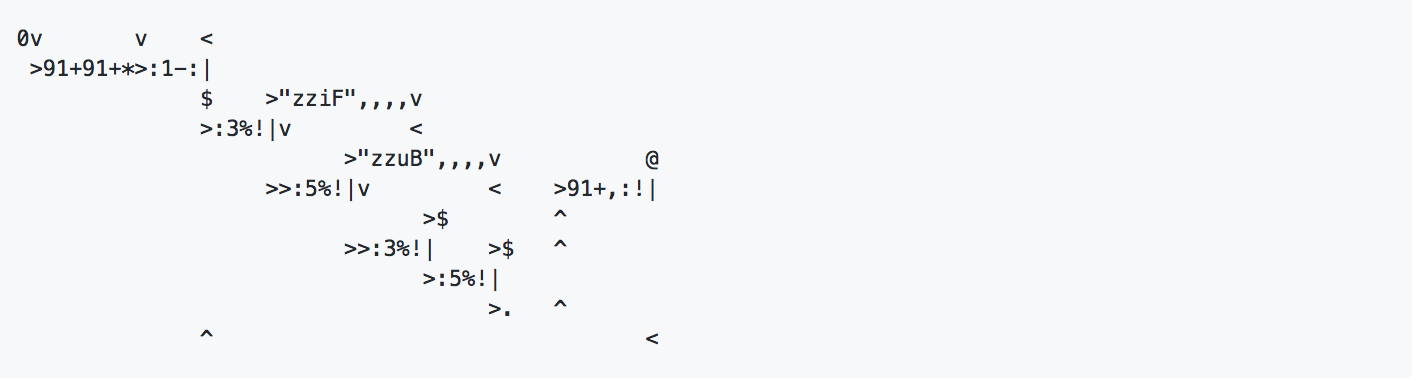
IDENTIFIER: 14

TEST CASE: Determine if the “Option…Time Program” is unchecked and the user is not informed of the time it takes to execute the program.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At command line, run “java com.laboon.JBefunge”
2. Enter the following code in the Program Area:



3. Press the “Run” button.

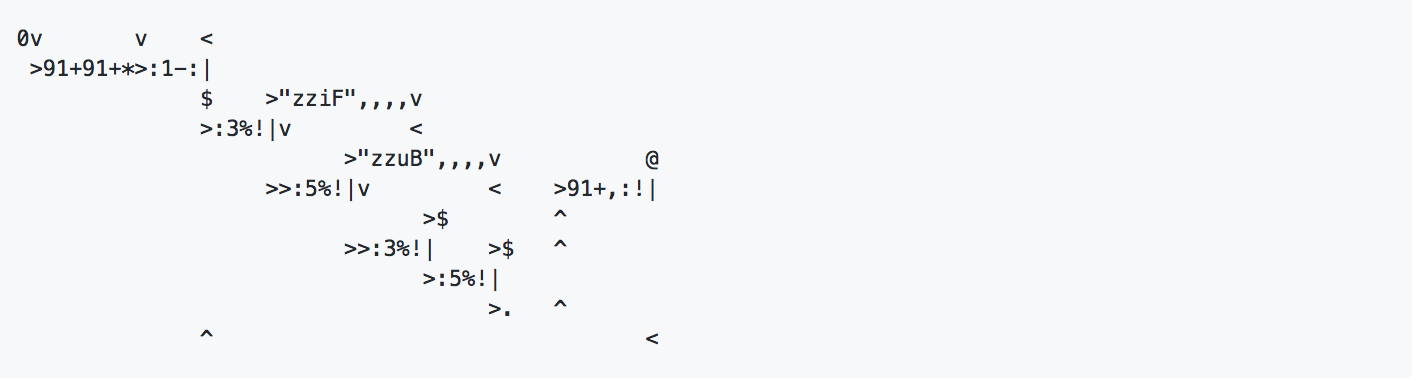
POSTCONDITIONS: After the program is finished executing, the user is not informed of the time to run the code. The “Stop” button is disabled and the buttons “Run”, “Walk”, “Mosey”, and “Step” are enabled.

IDENTIFIER: 15

TEST CASE: Determine if the system displays a cursor on the current opcode if and only if a program is being executed for the “Run” button.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Enter the following code in the Program Area:
3. Press the “Run” button to see yellow cursor appear.
4. Press “Stop” button.

POSTCONDITIONS: The yellow cursor appears in the Program Area on the current opcode that is being executed when the program has stopped executing.

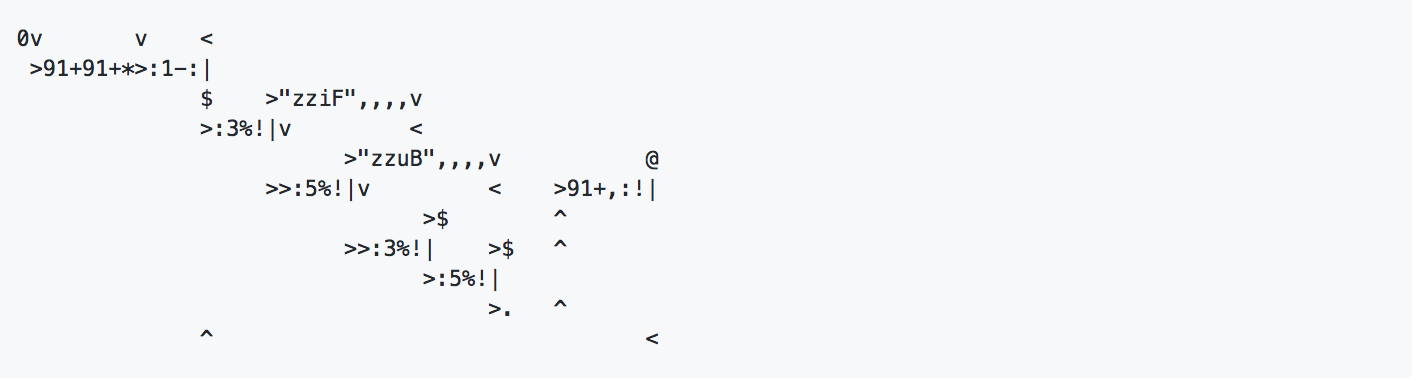
IDENTIFIER: 16

TEST CASE: Determine if the cursor will not appear when the program is not being executed.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly. The program has been executed once from pressing the “Walk” button and then stopped before the program finished executing.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Enter the following code in the Program Area:



1. Press the “Walk” button.
2. Press “Stop” button before program finishes.

POSTCONDITIONS: After stopping the program before it finishes, the cursor appears on the opcode it is currently executing.

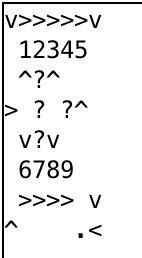
IDENTIFIER: 17

TEST CASE: Determine if when the “Options...Check For End Code” option is checked, the system warns the user if no end opcode exists in the program.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Click on “Options”, located in the Menu group.
3. Select “Check for end code” option.
4. Enter the following code in the Program Area:



1. Press the “Run” button.

POSTCONDITIONS: The program displays a Warning dialogue box that warns the user that no end opcode (@) was found. The message includes, “No end code opcode (@) found! This may result in a non-terminating program. Run anyway?” Give the user the option to run the program or to not.

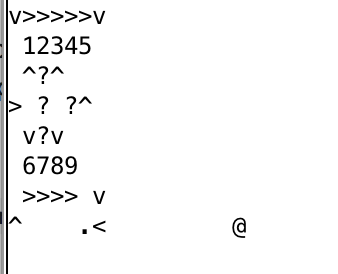
IDENTIFIER: 18

TEST CASE: Determine if when the “Options...Check For End Code” option is checked, the system will not show a warning to the user when an end opcode exists.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Click on “Options”, located in the Menu group.
3. Select “Check for end code” option.
4. Enter the following code in the Program Area:



1. Press the “Run” button.

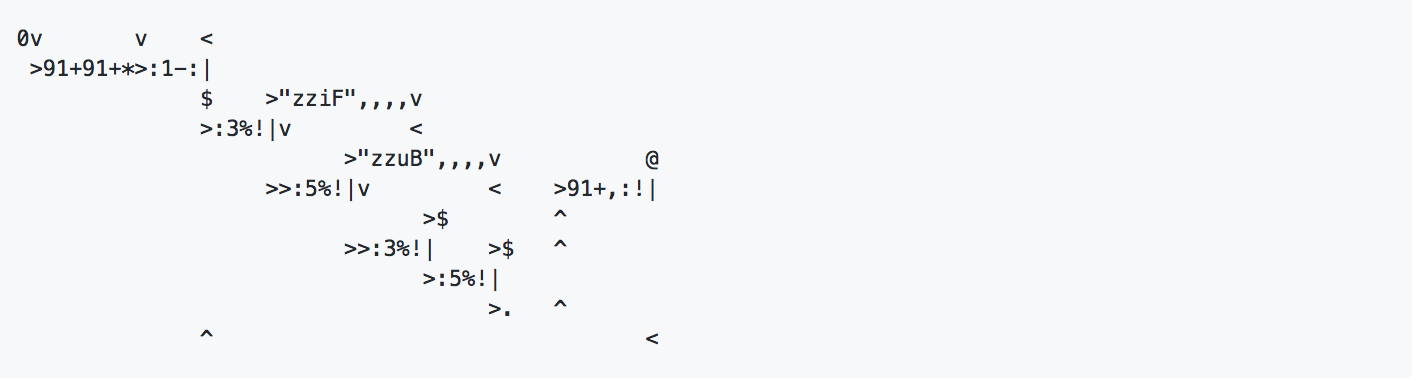
POSTCONDITIONS:The program does not display any warning to the user and executes the code.

IDENTIFIER: 19

TEST CASE: Determine if the system is able to complete execution of a reference FizzBuzz implementation in less than 30,000,000 microseconds on a MAC computer.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly. A 13-inch MacBook Pro laptop with a 2.3GHz dual-core Intel Core i5 processor was used for FizzBuzz. While other models exist for MacBooks that came before and after this model, this is the latest available model to the tester.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Click on “Options”, located in the Menu group.
3. Select “Time Program”
4. Enter the following code in the Program Area:
5. Click the “Run” button. Allow for program to execute completely.

POSTCONDITIONS: Program is able to run on computer and displays a message dialogue box showing the amount of time it took to execute, “Time to execute: 6326160 microseconds”.

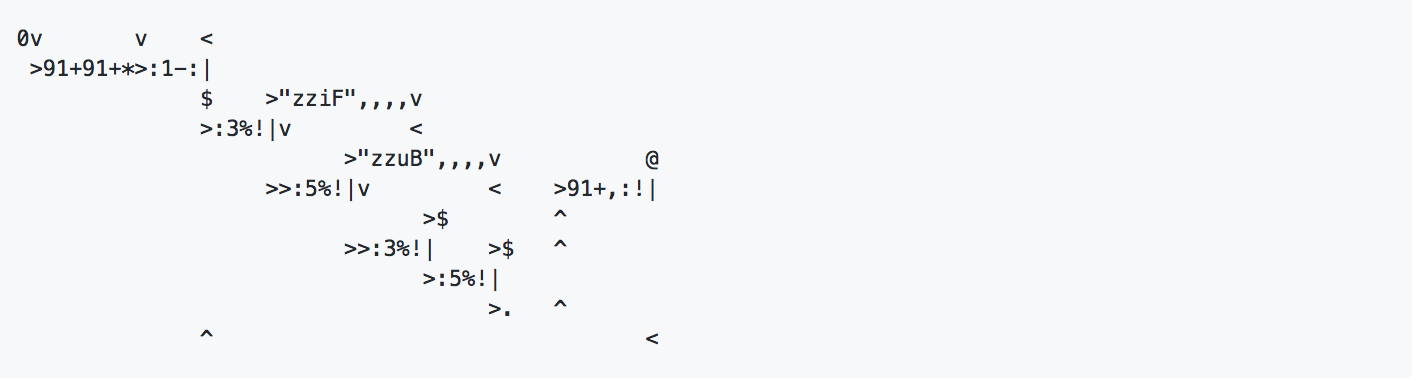
PERF-EXECUTION-TIME - On any given computer, the system shall be able to complete execution of a reference FizzBuzz implementation (i.e., the one listed in the README.md file of the JBefunge repository), in less than 30 seconds (30,000,000 microseconds).

IDENTIFIER: 20

TEST CASE: Determine if the system is able to complete execution of a reference FizzBuzz implementation in less than 30,000,000 microseconds on a Windows 7 Dell Desktop.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly. Use a Windows 7 Dell Desktop. While other models exist for desktop computers, outside of Dell, that came before and after this model, this is the latest available model to the tester.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Click on “Options”, located in the Menu group.
3. Select “Time Program”
4. Enter the following code in the Program Area:
5. Click the “Run” button. Allow for program to execute completely.

POSTCONDITIONS: Program is able to run on computer and displays a message dialogue box showing the amount of time it took to execute, “Time to execute: 6326160 microseconds”.

IDENTIFIER: 21

TEST CASE: Under File, determine if there are four menu items: Open File, Save File, Save As, and Quit.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly

EXECUTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Within the GUI, click under the File menu group and check that the four menu items (Open File, Save File, Save As, and Quit) are listed

POSTCONDITIONS: The four menu items (Open File, Save File, Save As, and Quit) are listed.

IDENTIFIER: 22 (Edge Case)

TEST CASE: Determine if the “Run” button is enabled when no code is in the Program Area.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”.
2. Press the “Run” button.

POSTCONDITIONS: After clicking the “Run” button, no output is given by program. The run button remains enabled after it has been clicked.

IDENTIFIER: 23 (Edge Case)

TEST CASE: Determine if the user can modify code in the Program Area while code is already executing.

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”.
2. Enter the following code in the Program Area: “>123456789.........<@”
3. Press “Mosey” button.
4. As code is executing, do not press the “Stop” button and modify the code in the Program Area to “>123456789......<@” by deleting three periods before the “<” symbol.
5. Press “Stop” button after 3 loops.

POSTCONDITIONS: After making changes to the code, the output did not reflect the changes. It continued to output values from the original code such as, “987654321000000000987654321123456789987654321”. This output is expected for code executed in step 2 but not in step 4.

IDENTIFIER: 24 (Corner Case)

TEST CASE: Determine if program will continue to execute code if no exit symbol is included and warning dialogue is ignored

PRECONDITIONS: Compiled JBefunge source and test files, GUI window is able to be loaded properly.

EXECUTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”.
2. Enter the following code in the Program Area: “>123456789…” and press “Walk”.
3. When the warning dialogue box, “No end opcode (@) found! This may result in a non-terminating program. Run anyway?”, appears, press “Yes”.
4. Allow for 3 iterations and then press “Stop” button
5. Read output

POSTCONDITIONS: The output appears as “987654321987654321987654321”.

**Defects**

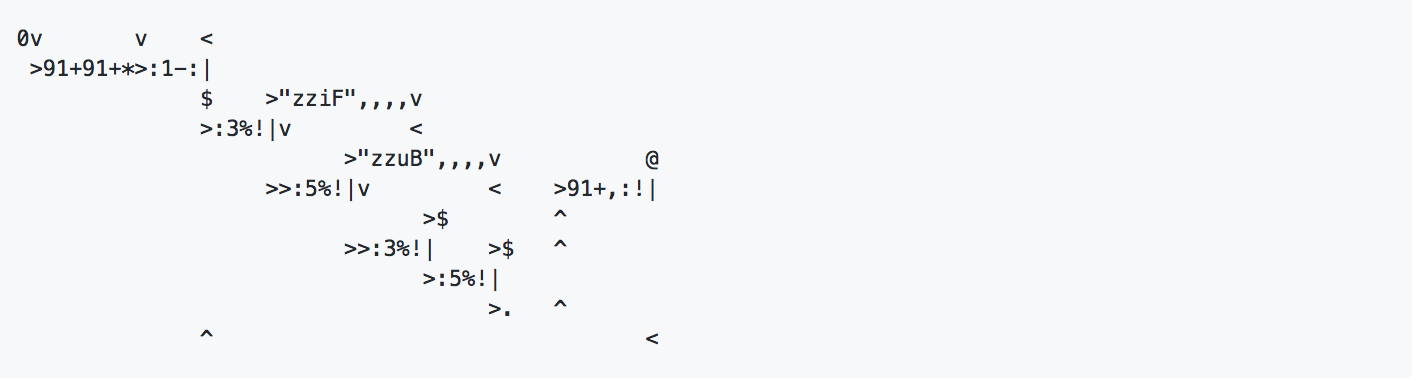
**Defect 1**

SUMMARY: Cursor does not disappear after “stop” button is pressed for “Walk” button.

DESCRIPTION: A defect was found when running test case 16. When the program has stopped executing, the yellow cursor should disappear from the program area.

REPRODUCTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”
2. Enter the following code in the Program Area:



1. Press the “Walk” button.
2. Press “Stop” button before program finishes.

EXPECTED BEHAVIOR: Cursor (highlighted) should disappear from program area when program is not executing or has stopped executing.

OBSERVED BEHAVIOR: The cursor remained in the program area, highlighting the opcode that it was on when stopping the program.

SEVERITY: The issue does not meet the requirements required by client. It is not a major issue in terms of functionality but none the less, it will need to be corrected.

IMPACT: Minor - this will not affect the user experience and can easily be fixed. However, since it is in the requirements, it will need to be corrected.

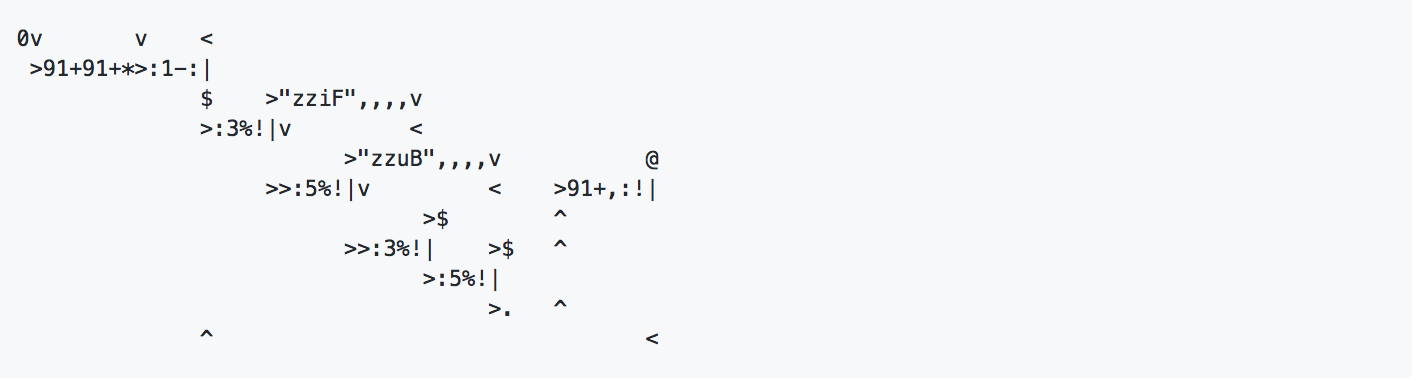
**Defect 2**

SUMMARY: Cursor highlighting subsequent code from location after stopping program.

DESCRIPTION: A further defect was found for test case 16. When code is executing under “Walk” and stopped in the middle of executing, editing the code where the Program Counter is highlighted on the current opcode will cause the subsequent typed code to be highlighted.

REPRODUCTION STEPS:

1. At the command line, run “java com.laboon.JBefunge”.
2. Enter the following code in the Program Area:



1. Press “Walk” button.
2. Press the “Stop” button when the cursor is highlighting the first addition symbol (‘+’) in the code in the program area.
3. Get cursor to be on the right side of the highlighted opcode and click to see blinking cursor.
4. Type in the Program Area “abc”.

EXPECTED BEHAVIOR: After executing the code, yellow cursor appears and highlights the current opcode that is being executed. When the “stop” button is pressed and the code is not executing the yellow cursor should disappear off the screen.

OBSERVED BEHAVIOR: After the “stop” button was pressed, the yellow cursor remained on the display. Subsequent code written after the highlighted opcode were also highlighted.

SEVERITY: Minor - The user may or may not run into this issue if similar or exact execution steps are replicated. However, since the program is still able to function properly and extra highlighting disappears after another button is pressed, it does not cause much of an issue to the user.

IMPACT: Minor -User will see modified code to be highlighted but when either the “Run”, “Walk”, “Mosey”, or “Step” buttons are pressed, the highlighting will disappear and the first opcode being executed in the program will be highlighted to mark the beginning of the execution.

**Defect 3**

SUMMARY: Missing textbox label

DESCRIPTION: No Program Area” label is displayed above the Program Area textbox (TEST CASE 1)

REPRODUCTION STEPS:

1. Type “java com.laboon.JBefunge” in command line to run program

2. Take note that the labels above the textboxes are correct

EXPECTED BEHAVIOR: A “Program Area” label should be displayed above the Program Area textbox

OBSERVED BEHAVIOR: No label is displayed above the Program Area textbox

SEVERITY: Minor - The Program Area textbox functions appropriately, but its function may not be immediately apparent to the user. User can infer the textbox is meant for inputting code.

IMPACT: Normal -This is a defect that inconveniences the user but does not impact the overall functionality of the program.

**ENHANCEMENTS**: System should not display a Java exception or stack trace directly to the user. Although logging errors and exception in the command line can be helpful for developing and managing the program, it should not be made visible to the user. This could lead to security or functionality issues if clients try to resolve problems on their own or unknown programmers attempt to change the Java code. Furthermore, it is not beneficial to the client who may not understand what is occurring in the command line.

**Traceability Matrix**

FUN-TEXT-DISPLAY: 1, 2

FUN-MENUS: 3, 4, 21

FUN-BEFUNGE: 5, 6

FUN-RUN-SPEED: 7, 8

FUN-STEP: 9, 10

FUN-STOP: 11, 12, 22

FUN-TIME: 13, 14

FUN-TRACE: 15, 16

FUN-CHECK-END-OPCODE: 17, 18

PERF-EXECUTION-TIME: 19, 20