CS 1632

Software Quality Assurance

Spring 2017

Deliverable 1:

Test Plan and Traceability Matrix

Julian Monticelli

JJ Naughton

Xinyue Guo

Julian:

Personally, a problem I encountered is determining whether something is a defect and deciding on what to focus on (i.e., is reading in “0” if I enter “lulz” for a number of iterations a bad thing). However, I think the part that took the longest was understanding, myself, how this actually worked – not the testing – but the probability part of the Monty Hall Program.

JJ:

Xinyue:

Test Plan

(for convenience, <https://github.com/laboon/CS1632_Spring2017/blob/master/deliverables/1/requirements.md>

is the link to the requirements)

<example> (I’m not sure that I’m right on this – some sort of confirmation would be nice)

(1.)

IDENTIFIER: TEST-BASIC-CALC

TEST CASE: This test case will verify that the calculation of the Monty-Hall program is accurately being tested for. The values should approach one-third and two-thirds probability for staying and switching, respectively, for sufficiently large values.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: Run the program with args “Car Goat 10000 4”.

POSTCONDITIONS: The program displays the chances of getting both the good and bad options had you switched or had you stayed, and the values are correctly displaying one-third probability for staying and two-thirds probability for switching.

(2.)

IDENTIFIER: TEST-BASIC-CALC

TEST CASE: This test will verify whether or not the program dispatches even amounts of work (or off-by-one if there is an unevenly divisible number of threads – so long as they are within reason of being split evenly)

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: Run the program with args “Car Goat 10000 <num>” where <num> is an integer that is randomly selected – should be above 0 and less than the integer limit. This process should be repeated many times with different numbers (both even and uneven) until we are convinced that there is no error.

POSTCONDITIONS: The program dispatches work to threads proportionally, allowing only a pigeon-hole principle difference of one where allowable.

(6.)

IDENTIFIER: TEST-SMALL-NUM-WITH-EDGE-CASE

TEST CASE: This test case aims to confirm that the program should not issue any warning when the number of iterations the user choose is no less than the recommended minimum, which is 100. Here we test with the edge value 100, which is the first integer that is no less than 100.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: Run the program with command line argument “java -jar GoatGoatCar.jar car goat 100 5”

POSTCONDITIONS: The program displays the chances of getting both the good and bad options had the user switched or had the user stayed in 100 iterations.

(7.)

IDENTIFIER: TEST-SMALL-NUM

TEST CASE: This test case aims to test if the program will issue proper warning when the number of iterations the user choose is less than the recommended minimum, which is 100.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: Run the program with command line argument “java -jar GoatGoatCar.jar car goat 80 5”

POSTCONDITIONS: The program issues a warning saying “Recommended minimum number of times is 100. Continue?[y/n]” and waits for the feedback from the user.

(8.)

IDENTIFIER: TEST-SMALL-NUM-CONT-CASE-INSENSITIVE-YES

TEST CASE: This test case aims to test that, when the program is taking further instruction from the user to deal with the warning on minimum number of iterations, as long as the response entered is ‘y’, regardless of its cases, the program should give the calculated output.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: 1. Run the program with the argument “java -jar GoatGoatCar.jar car goat 80 5” and wait until the warning for minimum iterations shows up.

2. Enter “Y” and press “Enter”.

3. Run the program with the argument “java -jar GoatGoatCar.jar car goat 80 5” and wait until the warning for minimum iterations shows up.

4. Enter “y” and press “Enter”

POSTCONDITIONS: The program displays the chances of getting both the good and bad options had the user switched or had the user stayed in 80 iterations after both Step 2 and 4.

(9.)

IDENTIFIER: TEST-SMALL-NUM-CONT-CASE-INSENSITIVE-NO

TEST CASE: This test case aims to test that, when the program is taking further instruction from the user to deal with the warning on minimum number of iterations, as long as the response entered is ‘n’, regardless of its cases, the program should exit immediately.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: 1. Run the program with the argument “java -jar GoatGoatCar.jar car goat 80 5” and wait until the warning for minimum iterations shows up.

2. Enter “N” and press “Enter”.

3. Run the program with the argument “java -jar GoatGoatCar.jar car goat 80 5” and wait until the warning for minimum iterations shows up.

4. Enter “n” and press “Enter”

POSTCONDITIONS: The program exits with words “Please retry with a higher number of times” after both Step 2 and 4.

(10.)

IDENTIFIER: TEST-SMALL-NUM-CONT-INVALID-RESPONSE

TEST CASE: This test case aims to test that, when the program is taking further instruction from the user to deal with the warning on minimum number of iterations, as long as the response entered is not ‘y’ or ‘n’, the program will keep asking the user for a valid selection.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: 1. Run the program with the argument “java -jar GoatGoatCar.jar car goat 80 5” and wait until the warning for minimum iterations shows up.

2. Enter “m” and press “Enter”.

3. Enter “d” and press “Enter”.

4. Enter “q” and press “Enter”.

5. Enter “n” and press “Enter”.

POSTCONDITIONS: The program displays “Sorry, I don’t know what m means!” after Step 2. The program displays “Sorry, I don’t know what d means!” after Step 3. The program displays “Sorry, I don’t know what q means!” after Step 4. The program exits with words “Please retry with a higher number of times” after Step 5.

(11.)

IDENTIFIER: TEST-NF-PERFORMANCE

TEST CASE: This test case aims to confirm that the program runs faster with a higher number of num\_threads given.

PRECONDITIONS: Command prompt window is opened and the current directory is the folder where GoatGoatCar.jar is saved.

EXECUTION STEPS: 1. Run the program with command line argument “java -jar GoatGoatCar.jar car goat 4000 5”

2. Benchmark the time used as t1.

3. Run the program with command line argument “java -jar GoatGoatCar.jar car goat 4000 4000”

4. Benchmark the time used as t2

POSTCONDITIONS: t2 is smaller than t1

Traceability Matrix

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req/Tst | (1) TEST- BASIC-CALC | (2) TEST-THREADS | (3) TEST-ARGS-NUMBER | (4) TEST-DISPLAY-RESULTS | (5) TEST-ARGS-INVALID | (6,7) TEST-SMALL-NUM(with or without edge case) | (8,9,10) TEST-SMALL-NUM-CONT (YES & NO) | (11) TEST-NF-PERFORMANCE |
| (1) FUN-BASIC-CALC | x |  |  |  |  |  |  |  |
| (2) FUN-THREADS |  | x |  |  |  |  |  |  |
| (3) FUN-DISPLAY-RESULTS |  |  | x |  |  |  |  |  |
| (4) FUN-ARGS-NUMBER |  |  |  | x |  |  |  |  |
| (5) FUN-ARGS-INVALID |  |  |  |  | x |  |  |  |
| (6) FUN-SMALL-NUM |  |  |  |  |  | x |  |  |
| (7) FUN-SMALL-NUM-CONT |  |  |  |  |  |  | x |  |
| (8) NF-PERFORMANCE |  |  |  |  |  |  |  | x |

Defects

(1)

SUMMARY: High num\_times error

DESCRIPTION: When you enter above (231 – 1), it does not read the value properly.

REPRODUCTION STEPS: Exceed signed integer positive limit (231 – 1) in num\_times. (For instance, use 231 as num\_times argument.)

EXPECTED BEHAVIOR: Display the results of 231 (or more) iterations of CarCarGoat.

OBSERVED BEHAVIOR: Displays that num\_times must be greater than 0.

(2)

SUMMARY: Threads don’t always make program faster

DESCRIPTION: The more threads the program runs with, the more time it takes to print out the number of iterations per thread, which will cause a delay in the run time of the program, and that is unexpected. Also, threads take extra time without the print statements due to thread switch overhead costs, and only offer improvement up to a certain point, (i.e. available amount of CPUs – this program does use multicore threading).

REPRODUCTION STEPS: 1. Run the program with command line argument “java -jar GoatGoatCar.jar car goat 4000 5”

2. Benchmark the time used as t1.

3. Run the program with command line argument “java -jar GoatGoatCar.jar car goat 4000 4000”

4. Benchmark the time used as t2

EXPECTED BEHAVIOR: t1 is less than t2

OBSERVED BEHAVIOR: t2 is noticeably longer than t1

(3)

SUMMARY: Program accepts the same value as success and failure flags

DESCRIPTION: The program will accept car for both “success” option and “failure” option, which is counterintuitive to the expectations of the program – how can one both win *and* lose for winning a car? This will make the problem itself nonsensical.

REPRODUCTION STEPS: 1. Run the program with java -jar GoatGoatCar.jar Car Car 1000 4

EXPECTED BEHAVIOR: Although not explicitly stated, it is implicit that it would throw some sort of error because the problem wouldn’t make sense.

OBSERVED BEHAVIOR: Runs as normal without warning the user that it makes no sense.

(4)

SUMMARY: small\_num border value is wrong

DESCRIPTION: The recommended minimum number for iterations is 100, so passing 100 as the iteration number should not arouse any warning, but the testing fact is the opposite.

REPRODUCTION STEPS: 1. Open the command prompt window and change the directory to the folder where GoatGoatCar.jar is saved.

2. Run the program with command line argument “java -jar GoatGoatCar.jar car goat 100 5”

EXPECTED BEHAVIOR: The program displays the chances of getting both the good and bad options had the user switched or had the user stayed in 100 iterations.

OBSERVED BEHAVIOR: The program issues a warning saying “Recommended minimum number of times is 100. Continue?[y/n]” and waits for the feedback from the user.

(You guys can add more if you want :)