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)

IDENTIFIER: TEST-BASIC-CALC

TEST CASE: Run the program with args “Car Goat 10000 8” or another number of tries and threads and make sure that it calculates the results of switching and staying in the Monty Hall problem.

PRECONDITIONS: None (the program is run from program arguments)

EXECUTION STEPS: Run the program with args “Car Goat 10000 8” (or a reasonable amount of iterations and threads)

POSTCONDITIONS: The program displays the chances of getting both the good and bad options had you switched or had you stayed.

Traceability Matrix

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Req/Tst | TEST- BASIC-CALC | TEST-THREADS | TEST-ARGS-NUMBER | TEST-DISPLAY-RESULTS | TEST-ARGS-INVALID | TEST-SMALL-NUM | TEST-SMALL-NUM-CONT | TEST-NF-PERFORMANCE |
| FUN-BASIC-CALC | x |  |  |  |  |  |  |  |
| FUN-THREADS |  | x |  |  |  |  |  |  |
| FUN-DISPLAY-RESULTS |  |  | x |  |  |  |  |  |
| FUN-ARGS-NUMBER |  |  |  | x |  |  |  |  |
| FUN-ARGS-INVALID |  |  |  |  | x |  |  |  |
| FUN-SMALL-NUM |  |  |  |  |  | x |  |  |
| FUN-SMALL-NUM-CONT |  |  |  |  |  |  | x |  |
| NF-PERFORMANCE |  |  |  |  |  |  |  | x |

Defects

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.

SUMMARY: Threads don’t always make program faster

DESCRIPTION: Threads only make the program faster up to a certain point, depending on amount of available CPU cores that the current computer has, and after that tend to show diminishing returns from I/O and context switching.

REPRODUCTION STEPS: Run program with 1 Billion iterations at 4 threads, and benchmark speed. Then, run program with 1 Billion iterations at 4,000 threads, and benchmark speed.

EXPECTED BEHAVIOR: Program runs any rate faster with 4,000 cores than it does 4 cores.

OBSERVED BEHAVIOR: Program runs significantly slower with 4,000 cores as compared to running with 4 cores.

SUMMARY: Threads don’t always make program faster

DESCRIPTION: Threads only make the program faster up to a certain point, depending on amount of available CPU cores that the current computer has, and after that tend to show diminishing returns from I/O and context switching.

REPRODUCTION STEPS: Run program with 1 Billion iterations at 4 threads, and benchmark speed. Then, run program with 1 Billion iterations at 4,000 threads, and benchmark speed.

EXPECTED BEHAVIOR: Program runs any rate faster with 4,000 cores than it does 4 cores.

OBSERVED BEHAVIOR: Program runs significantly slower with 4,000 cores as compared to running with 4 cores.

(You guys can add more if you want :)