CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

Project Under Test: GoatGoatCar

Authors: Raj Patel and Christopher Good

INTRODUCTION

The purpose of this test plan is to provide the test cases used to identify the defects of the GoatGoatCar.jar. The GoatGoatCar program simulates the Monty Hall problem, and accepts four arguments in this order: good option, bad option, number of iterations, and number of threads. The displayed results include the number of iterations run per thread and the chances of getting the good and bad options in percentages if you switch doors or stay.

When formulating test cases, we first came up with base cases that directly correspond with the requirements. In other words, we just made cases that simply check the program accepted normal inputs, ran without issue, and displayed the expected results. In coming up with edge cases, we tested the upper and lower bound limits of the num_threads and num_times arguments, the latter of which was found using MAXINT, as well as testing varying numbers of arguments. To decide corner cases, we mainly employed atypical data types and special characters in each argument.

We encountered a few challenges while documenting the test plan. A glaring difficulty was simply coming up with multiple test cases to test each requirement. While this was easy for some requirements, it was difficult for others. Initially, finding a way to measure the thread performance was a challenge. We found a solution in running the program on a Linux system and using the “time” command to record the execution with each iterating thread count. Formulation of edge and corner cases was also a concern, as it required to rally think outside the box on what a user could possibly enter as argument that would cause the system to encounter an error.

TEST CASES

IDENTIFIER: TEST-ARGS-INVALID-CHOICES
TEST CASE: A test to make sure the program provides the proper response to a set of unusual and assumed invalid choice arguments.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: With all other arguments being valid, run the program with the choice arguments consisting of positive integers, negative integers, zeros, special characters (e.g. ‘^’,’%’,’*’,’!’,’`’,etc.) in different orders.
POSTCONDITIONS: The program will run normally.

IDENTIFIER: TEST-ARGS-INVALID-CHOICES-SAME
TEST CASE: A test to make sure the program provides the assumed proper response when the choice arguments are not different.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: With all other arguments being valid, run the program with the good and bad choice being the same string.
POSTCONDITIONS: The program will display a message explaining why it cannot run and will then shut down without any java exception or stack trace being shown.

IDENTIFIER: TEST-ARGS-INVALID-ITERATIONS
TEST CASE: A test to make sure the program provides the proper response to a set of invalid num_times arguments.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: With all other arguments being valid, run the program with the num_times argument being a non-positive integer, floating point value, and string.
POSTCONDITIONS: The program will display a message explaining why it cannot run and will then shut down without any java exception or stack trace being shown.

IDENTIFIER: TEST-ARGS-INVALID-THREADS
TEST CASE: A test to make sure the program provides the proper response to a set of invalid num_threads arguments.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: With all other arguments being valid, run the program with the num_threads argument being a non-positive integer, floating point value, and string.
POSTCONDITIONS: The program will display a message explaining why it cannot run and will then shut down without any java exception or stack trace being shown.

IDENTIFIER: TEST-ARGS-ITERATIONS-MAX

TEST CASE: A test to check whether the num_threads has a maximum limit before either not accepting the argument or displaying an error.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: With all other arguments valid, run the program with the num_times argument being above 2,147,483,647 (MAXINT).
POSTCONDITIONS: The program will display a message explaining why the number of iterations passed into the num_times argument was too high, an end the program.

IDENTIFIER: TEST-ARGS-NUMBER
TEST CASE: This test case tests the boundary values around the proper number of arguments, along with the proper number of arguments, to ensure the program responds as required in FUN-ARGS-NUMBER.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Using valid argument values, run the program with all arguments except one (removed argument is left to the tester's discretion). Run the program again with all arguments. Finally, run the program with an extra argument (value of argument does not matter).
POSTCONDITIONS: The program will only run when four arguments are passed. When three and five arguments are passed, the program shows an explanation of why it cannot run and then will shut down.

IDENTIFIER: TEST-ARGS-NUMBER-EMPTY
TEST CASE: This test case tests how the program responds to zero and empty arguments.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Using valid argument values, run the program with empty strings as the good/bad choice values. Then, run the program with no arguments whatsoever.
POSTCONDITIONS: The program will still run with the empty strings. The run with zero arguments will present a usage message and shut down.

IDENTIFIER: TEST-BASIC-CALC
TEST CASE: A simple test to make sure the program runs and calculates the results of the monty hall problem.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Run the program with valid arguments.
POSTCONDITIONS: The program will show the calculated results of switching or staying for however many iterations were passed in via command line arguments.

IDENTIFIER: TEST-DISPLAY-RESULTS
TEST CASE: A test to ensure the results of the calculations are displayed as required.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Run the program with valid input (specifically with num_times > 100)
POSTCONDITIONS: The program, after ending, displays the calculated stay and switch

percentages with three decimal places of accuracy, all under the good_choice and bad_choice arguments as inputted.

IDENTIFIER: TEST-SMALL-NUM
TEST CASE: This test checks that the low iteration warning is displayed under the proper conditions.
PRECONDITIONS: Valid input is chosen for the good choice, bad choice, and number of threads.
EXECUTION STEPS: Run the program so that 99 iterations are calculated. Run the program again with 100 iterations, and then 101 iterations.
POSTCONDITIONS: During the first runthrough (with 99 iterations), a warning asking if the user wants to continue is displayed. The second two runthroughs will show no warning.

IDENTIFIER: TEST-SMALL-NUM-CONT-INVALID
TEST CASE: This test checks to make sure the continuity menu for <100 times works as required with invalid input.
PRECONDITIONS: The user runs the program with less than 100 iterations and the "continue" menu is present.
EXECUTION STEPS: When the menu asks whether to continue, enter a character that is not 'y','Y','n', or 'N'. Repeat this with an integer, a floating point value, and string starting with 'y', 'Y', 'n', and 'N'.
POSTCONDITIONS: The program will display a message saying "Sorry, I don't know what <input> means!", where <input> is the input value.

IDENTIFIER: TEST-SMALL-NUM-CONT-VALID
TEST CASE: This test checks to make sure the continuity menu for <100 times works as required with valid input.
PRECONDITIONS: The user runs the program with less than 100 iterations and the "continue" menu is present.
EXECUTION STEPS: When the menu asks whether to continue, press 'y' and then enter. Reach the menu again and repeat with 'n', then with 'Y', and finally with 'N'.
POSTCONDITIONS: The program will continue upon entering 'y' or 'Y', and exit upon pressing 'n' or 'N' with no messages stating the input is invalid.

IDENTIFIER: TEST-THREAD-BALANCE
TEST CASE: A test to confirm that the the iteration load is balanced between all threads.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS:

1. Run the program such that the arguments match this list:
    - 1.1. good_choice: Car
    - 1.2. bad_choice: Goat
    - 1.3. num_times: 997
    - 1.4. num_threads: 1
2. Record thread load balance.
3. Repeat steps 1 and 2, incrementing upward the value of num_threads until its value reaches 11.

POSTCONDITIONS: In every recorded load balance, the balance between any two threads should be within a zero to one iteration difference.

IDENTIFIER: TEST-THREAD-BALANCE-DISPLAYED
TEST CASE: Test to ensure that the number of iterations per thread is displayed at its designated location.
PRECONDITIONS: User is in the command prompt with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS:

1. Run the program such that the arguments adhere to this list:
    - 1.1. good_choice: Car
    - 1.2. bad_choice: Goat
    - 1.3. num_times: <any number greater than 100>
    - 1.4. num_threads: <any integer greater than one>

POSTCONDITIONS: The number of iterations allocated to each thread should be displayed before the switch percentages are calculated.

IDENTIFIER: TEST-THREAD-PERFORMANCE
TEST CASE: A test to ensure the program improves performance with a greater thread count.
PRECONDITIONS: User is in a Linux terminal window or Windows Powershell shell with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Using either the “time” command if in Linux, or “Measure-Command” command in Windows Powershell, run the program with 1,000,000 iterations and one thread. Repeat this process while incrementing the number of threads upward until it has been run with 10 threads.
POSTCONDITIONS: The execution time as recorded will decrease each time the thread count increases.

IDENTIFIER: TEST-THREAD-PERFORMANCE-EXTREME
TEST CASE: A test to ensure the program performs as expected with an unreasonably high thread count.
PRECONDITIONS: User is in a Linux terminal window or Windows Powershell shell with the location of the GoatGoatCar.jar file being the active directory.
EXECUTION STEPS: Using either the “time” command if in Linux, or “Measure-Command” command in Windows Powershell, run the program with MAXINT (2,147,483,647) iterations and 10 threads. Then repeat process with MAXINT iterations and MAXINT threads.
POSTCONDITIONS: The execution time will be comparable or lower than that of 10 threads.

TRACEABILITY MATRIX

REQUIREMENT	TEST CASE
FUN-BASIC-CALC	TEST-BASIC-CALC
FUN-THREADS	TEST-THREAD-BALANCE-DISPLAY
	TEST-THREAD-BALANCE
FUN-DISPLAY-RESULTS	TEST-DISPLAY-RESULTS
FUN-ARGS-NUMBER	TEST-ARGS-NUMBER
	TEST-ARGS-NUMBER-EMPTY
FUN-ARGS-INVALID	TEST-ARGS-INVALID-ITERATIONS
	TEST-ARGS-INVALID-THREADS
	TEST-ARGS-INVALID-CHOICES
	TEST-ARGS-INVALID-CHOICES-SAME
	TEST-ARGS-ITERATIONS-MAX
FUN-SMALL-NUM	TEST-SMALL-NUM
FUN-SMALL-NUM-CONT	TEST-SMALL-NUM-CONT-VALID
	TEST-SMALL-NUM-CONT-INVALID
NF-PERFORMANCE	TEST-THREAD-PERFORMANCE
	TEST-THREAD-PERFORMANCE-EXTREME

FOUND DEFECTS

SUMMARY: FUN-SMALL-NUM-CONT defect
DESCRIPTION: On continue prompt, capital "N" are not accepted. This defect can be found by employing the test case TEST-SMALL-NUM-CONT-INVALID.
REPRODUCTION STEPS: Run the program appropriate argument types, however keep the number of iterations below 100. The program will then prompt the user "Continue? "[y/n]". Enter a capital "N".
EXPECTED BEHAVIOR: Program will accept the capitalized command as "no"
OBSERVED BEHAVIOR: Program does not accept the capitalized commands as "no", and prompts the user again for input.

SUMMARY: FUN-ARGS-INVALID defect
DESCRIPTION: On input of a floating point value for the number of threads, the system displays a Java exception. This can be found by employing test case TEST-ARGS-INVALID-THREADS.
REPRODUCTION STEPS: Run the program, but pass in a floating point value as the argument for number of threads.
EXPECTED BEHAVIOR: The program will display a message saying the input value must be a positive integer.
OBSERVED BEHAVIOR: The system displays a NumberFormatException and shuts down the program.

SUMMARY: FUN-SMALL-NUM defect
DESCRIPTION: Program issues recommendation when number of iterations is exactly 100. This defect can be found using the test case TEST-SMALL-NUM.
REPRODUCTION STEPS: Run the program with the argument for the number of iterations set at exactly 100
EXPECTED BEHAVIOR: The program will run and display its results.
OBSERVED BEHAVIOR: The program issues a recommendation, "Recommended minimum number of times is 100". The program then prompts the user, asking if the user wishes to continue.

SUMMARY: FUN-DISPLAY-RESULTS defect
DESCRIPTION: Both good and bad options as the same string does not bring up an error, which can be confusing the user. This can be found by employing test case TEST-ARGS-INVALID-CHOICES.
REPRODUCTION STEPS: Run the program with the same value in the arguments for the good option and bad option.
EXPECTED BEHAVIOR: Program will catch this error and either display a message requesting the inputs be different strings, or shut down the program.
OBSERVED BEHAVIOR: Program runs without error and displays the results regardless of similar names for the good and bad options.

SUMMARY: NF-PERFORMANCE defect
DESCRIPTION: When reviewing thread performance, it was revealed that a higher number of threads did not result in a better performance. This was found using the test case, TEST-THREAD-PERFORMANCE.
REPRODUCTION STEPS: Run the program multiple times, each time with an incremented thread count. Record the execution time of that run.
EXPECTED BEHAVIOR: With each run, the execution time decreases, as there are more threads running the process.
OBSERVED BEHAVIOR: Execution time did not decrease with each run. The execution times were varied, and so performance did not necessarily improve with a higher thread count but rather remained inconsistent.