|  |  |
| --- | --- |
| Correct Seal Colors | Northeastern University  CS 5500 – Managing Software Development  Spring, 2018 – Professors Annunziato and Weintraub |

Grading for Homework 4: Testing

|  |  |
| --- | --- |
| Student Name: | Samanjate Sood |
| Repo name: | student-137 |
| Repo commit date | Mar 16 |
| Graded by | Biswaraj |
| Total Score | 88.5 |

|  |
| --- |
| **1a: (10 points): +1** |
| +1 Any of the following inputs are provided: NULL , MAXINT, or ‑MAXINT, or any cases with bad data types (e.g. float, char, string). |
| **1b: (10 points): -7** |
| −3 Inputs {MIN\_INT, MAX\_INT, 11.0, 10.01} should be true  −2 Inputs {10, -9.99, 0, 9.99} should be false  −2 Summary incorrect, it’s not between -10 and 10, its ‘x<-10 or x>10’. |
| **1c: (10 points): +1** |
| +1 Test case that includes: NULL, a long string, a string without digits (ideally 10 characters), and a string with non-printable characters. |
| **2a For the method sqrt (5 pts)** |
|  |
| **2b For the method sqr (5 pts)** |
| **2c For the method factorial (5 pts): +1**  +1 Following inputs are provided: mis-typed input (char, string...) (negative case), negative int (negative) |
| **3 Evaluate a test specification (20 pts) : -3.5** |
| **-3.5** for missing the following cases:   1. If the answer doesn't suggest the pair {MAX\_INT, 1} and label it as Overflow. 2. If the answer doesn't suggest the pair {-1, -MAX\_INT} or {-1, MIN\_INT} and label it as Underflow.   Warning: the answer doesn't suggest NULL for one or both of the inputs (three cases). |
| **4 Create a suite of at least 20 tests.** |
|  |
| **5. Create two different implementations of the Sorter interface (20 pts): -4** |
| -2 there are any signs of sloppiness. (e.g. TODO comments, obsolete code that was commented out, duplicated code, missing comments etc.): Car and Person classes and its methods have no comments  -2 The solution does not use the required package or class names for the sorters and the tests. |
| **6. structural testing coverage** |
| 1. for each of your sorters, create a report listing statement coverage and branch coverage achieved by the test suites from Question 4. 2. Attached to this assignment is another test suite for sorters. For each of your sorters, create a report listing statement coverage and branch coverage achieved by this test suite. 3. Extend your test suites from Question 4 by adding tests needed to achieve 100% statement coverage. Place the resulting test suites in a package tests3c. 4. Extend the test suite that you downloaded by adding tests necessary to achieve 100% branch coverage. Place the resulting test suite in a package tests3d. |