**Texas Tech University**

**Computer Science Department - Software Engineering II**

**Due: March 25, 2020 (Midnight)**

**Team-based Project: Release # 2**

**5 marks of the total course mark**

**Submission through the Blackboard System ONLY**

**No Other Forms of Submissions Will be Accepted**

**Goal**: Creating software testing artifacts for the team-based project implementation

**Description**:

In the first release of your project, your team created a simple application to practice “agile methodology.” In this release, you will create test harness and artifacts for testing your application.

**Expectations**:

Your team will create the software testing artifacts and upload them to the GitHub repository, your team created for the team-based project. The software testing artifacts should be uploaded on the GitHub in an organized style. For instance, the functional testing artifacts (i.e., the specification and JUnit implementation) should be uploaded on folders called:

1. ../test/functional/specification
2. ../test/functional/JUnit

Similarly, the structural testing artifacts (i.e., the specification and JUnit implementation) should be uploaded on folders called:

1. ../test/structural/specification
2. ../test/structural/JUnit

The requirements for functional tests:

1. The equivalence partitioning of input variables for your program
2. The test case specification for testing each partition
3. The JUnit implementation for your test case specification (some of the test case specification might not be directly implementable using JUnit. In these cases you ignore their JUnit implementation

The requirements for Structural tests:

1. Capture the statement adequacy of the functional test cases developed using coverage tools (e.g., Cobertura for Java)
2. Add additional test cases to your test pool to achieve 100% statement coverage
3. Capture the screenshot showing that your test pool achieves 100% statement adequacy.

**Deliverable**:

1. Functional Test (**Total: 3 marks**)
   1. The specification of equivalence partitioning of input variables of your code for your project (a tabular format similar to your assignment 2, Table 1) (1 mark)

|  |  |  |  |
| --- | --- | --- | --- |
| Partition ID | Input variable | Valid partition | Invalid partition |
| 1. | number | 4<=id | 4>id |
| 2. | number | 0>id | 0<id |
| 3. | alphabet | numbers | char |
| 4. | Fractions | Real numbers | id != x/y |
| 5 | integers | id>0 | id<0 |
| 6 | Symbols | Real numbers |  |
| 7 | decimals | 0<hours | Hours>24 |
| 8 | Alpha numeric | Real numbers |  |
| 9 | Multiple inputs | Single input real number |  |
| 10 | Number too high | 0<hours | Hours>24 |

Template for Table 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Test ID | Test inputs | Expected output | Partition ID covered |
| 1 | 1 | cont | 1 |
| 2 | 1000 | error | 10 |
| 3 | 25 | 10.0 | 2 |
| 4 | A | error | 5 |
| 5 | 0 | quit | 7 |
| 6 | 2/3 | error | 6 |
| 7 | -3 | null | 7 |
| 8 | / | error | 8 |
| 9 | 10A | error | 9 |

* 1. The JUnit implementation of the test cases you devised (1 mark)

1. Structural Test (**Total: 2 marks**)
   1. The initial screen shot of the statement adequacy achieved by your functional tests (0.5 mark)
   2. The JUnit implementation of the additional tests to increase the statement coverage to 100% (1 mark)
   3. The final screen shot of the statement adequacy achieved by your functional and structural tests (0.5 mark)

**Note**. If your initial screen shot already shows achieving 100% statement coverage, you can skip 2.b. and 2.c)

The contribution of each teammate **should be submitted through the blackboard system separately along with the link for the GitHub repository**.

1. Contribution of each teammate. Who created which diagrams? (Mandatory: without this submission the team will not receive any credit for this release).