U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

M.EIC045 and MESW0004
Software Testing, Verification and Validation
1st Exam

January 19, 2024

Student name: _____
Student number: _____

- ☐ **M.EIC**
- ☐ **MESW**

This *exam* is composed by two main groups:

- Group I is worth 25% (i.e., 5 points out of 20) and it is composed of 10 multiple choice questions.

(a) Note: There is only one correct answer per question!

(b) Note: Each correct answer is worth +0.5 points (out of 20) and each incorrect answer has a -0.25 penalty. In case you get a negative overall score in this Group, your score will be truncated to 0.

- Group II is worth 75% (i.e., 15 points out of 20) and it is composed of three open questions each with one or more sub questions.

Few additional details:

- You have 90 minutes to complete this exam.
  - In the first 30 minutes, you are not allowed to leave the room.
  - Once the first 30 minutes have passed, you are allowed to hand over your exam and leave the room.
- You should answer every single question to the best of your knowledge and you must write every single answer in the provided pieces of paper. No other pieces of paper will be considered/evaluated.
  - Group I must be answered in this document.
  - Group II must be answered in the provided/separated sheet.
- You are NOT allowed to use or consult any additional material (books, slides, etc) during the exam.
- The use of mobile phones or any other electronic devices is strictly prohibited during the exam and must be turned off.
- You may answer the open questions in Portuguese or English.

GROUP I (5 points)

1. Select the **CORRECT** statement:

- ☐ A defect is detected during software execution.
- ☐ A fault can be located during debugging.
- ☐ A stressed programmer introduces failures in the code.
- ☐ A failure is detected by static analysis.

2. Which of the following statements regarding static testing is **FALSE**?

- ☐ Static testing includes techniques such as reviews and inspections.
- ☐ Defects found by reviews are cheaper to fix than those found dynamically.
- ☐ Control flow analysis finds unused variables while Data Flow finds unreachable code.
- ☐ Metrics, such as cyclomatic complexity, may be calculated by static analysis.

3. Which expression best matches the following characteristics of review processes:

S) inspection	1. led by author
T) peer review	2. undocumented
U) informal review	3. no management participation
V) walkthrough	4. led by a trained moderator or leader
	5. uses entry and exit criteria

- ☐ S = 4, T = 3, U = 2 and 5, V = 1
- ☐ S = 4 and 5, T = 3, U = 2, V = 1
- ☐ S = 1 and 5, T = 3, U = 2, V = 4
- ☐ S = 5, T = 4, U = 3, V = 1 and 2
- ☐ S = 4 and 5, T = 1, U = 2, V = 3

4. Which of the following statements regarding Model Based Testing (MBT) is **TRUE**?

- ☐ MBT is white-box testing.
- ☐ In MBT, the oracle is the model being used.
- ☐ MBT is static testing because it is based on models.
- ☐ MBT is used for functionality testing, such as, usability.

5. Select the **CORRECT** statement:

- ☐ There is only one Test Plan for each project.
- ☐ The Test Case Specification is written during test case execution.
- ☐ The Test Policy is defined at the organization level.
- ☐ The Test Incident Report is written during test planning activities

6. According to the requirements, you get 2 equivalent classes for variable A, 2 equivalent classes for variable B, and 2 equivalent classes for variable C. What is the number of test cases that you get when combining the input values according to pairwise testing?

- ☐ 4 tests
- ☐ 5 tests
- ☐ 6 tests
- ☐ 7 tests

7. According to requirement R1, A can belong to [0,55[and [55-99]. Regarding the graphical user interface which does not allow inserting negative numbers nor 3-digit numbers, which of the following test cases achieve 100% coverage according to boundary value analysis for variable A?

- ☐ 0, 54, 55, 56, 99
- ☐ 0, 1, 2, 54, 55, 56, 98, 99
- ☐ 0, 1, 54, 55, 56, 98, 99
- ☐ 0, 1, 54, 55, 56, 97, 98, 99

8. Consider the following program:

```
1. read(B); read(A);
2. if (A > 2) {
3.     A = A - 1;
4.     while (B > 0) {
5.         B = B - 5;
6.     }
7. }
8. print(B);
9. print(A);
```

What is the minimum number of test cases needed to achieve 100% statement coverage and 100% decision coverage?

- ☐ 1 test
- ☐ 2 tests
- ☐ 3 tests
- ☐ 4 tests

9. Consider the following Boolean expression:

```
(~A or B) and (A or ~C)
```

What is the minimum number of test cases needed to achieve 100% MC/DC (Modified Condition Decision Coverage)?

- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6

10. Consider the following program:

```
1. read(A); read(B);
2. while (A > 0) {
3.     if (B < 5) {
4.         A = A - 5;
5.         B = B + 1;
6.     }
7. }
8. print(A);
```

What of the following are examples of Def-clear paths for variable A?

- ☐ 1-2-3 and 1-2-3-4 and 1-2-3-4-5 and 4-5-6-7-2-8
- ☐ 1-2 and 1-2-3-4 and 4-5-6-7-2 and 4-5-6-7-2-8
- ☐ 1-2-3-4-5 and 4-5
- ☐ 3-4 and 4-5-6-7-2-3-4

GROUP II (15 points)

1. Model-based Testing (7 points)

Consider there is a microwave oven with the following requirements:

- Its initial state is OFF.
- When the user turns it on, the machine goes to an ON state.
- If the user selects the warm program, then, the WARMING process starts. It goes back to ON when the defined time is reached. A user may cancel it at any time, taking the microwave back to the ON state.
- If the user selects the defrost program, then, the DEFROSTING process starts. It goes back to ON when the defined time is reached. A user may cancel it at any time, taking the microwave back to the ON state.
- The user can turn off the microwave (after which it is OFF), but only if the microwave is not warming up or defrosting food.

1.1 Draw a minimal state machine to represent the requirements. Tip: if a transition is not specified in the requirements, it simply does not exist, and thus, should not be represented in the state machine. (+2/7 points)

1.2 Derive a state transition tree for the microwave state machine. (+2/7 points)

1.3 Derive the transition table of the state machine. How many sneaky paths can be tested based on this transition table? (+3/7 points)

2. Structural and Logical Coverage (4 points)

Consider the following method named `isItLeapYear`.

```java
/**
 * Checks whether the year is divisible by 4 but not by 100, or it is
 * divisible by 400, to determine if it is a leap year. It returns true
 * if it is a leap year, false otherwise.
 */
1. public static boolean isItLeapYear(int year) {
2.     if ((year % 4 == 0 & year % 100 != 0) | (year % 400 == 0))
3.         return true;
4.     return false;
5. }
```

2.1 Define a minimal set of test cases (i.e., inputs) that fulfills MC/DC. Justify your answer. (+4 points)

3. Mutation Testing (4 points)

Consider the following class `MyMath` with the method `distanceToInt()`.

```java
public class MyMath {
   /**
    * Computes the difference between the given number and the nearest
    * whole integer.
    */
1.  public static double distanceToInt(double a) {
2.      if (a < 0) {
3.          a *= -1;
4.      }
5.      if (a % 1 <= 0.5) {
6.          return a % 1;
7.      } else {
8.          double fractional = a % 1;
9.          return 1 - fractional;
10.     }
11. }
12. }
```

3.1 Consider the following mutated version of the `distanceToInt()` method, which mutated line 3 from `a *= -1;` to `a /= -1;` . (+1/4 points)

```java
1.  public static double distanceToInt(double a) {
2.      if (a < 0) {
3.          a /= -1;
4.      }
5.      if (a % 1 <= 0.5) {
6.          return a % 1;
7.      } else {
8.          double fractional = a % 1;
9.          return 1 - fractional;
10.     }
11. }
```

Which of the following statements is **NOT** correct?

- ☐ The mutant is impossible to kill.
- ☐ Adding this mutant to the set of all mutants when testing will not affect the mutation score.
- ☐ The mutant behaves differently from the original program.
- ☐ The mutant was created by applying syntactic changes.

3.2 Describe a non-equivalent mutant for the original `distanceToInt()` method which the following test suite would not kill. Justify your answer. (+3/4 points)

```java
@Test
public void testDistanceToInt() {
    assertEquals(0.3, MyMath.distanceToInt(7.3));
}

@Test
public void testDistanceToIntNegative() {
    assertEquals(0.1, MyMath.distanceToInt(-4.1));
}

@Test
public void testDistanceToIntHalf() {
    assertEquals(0.5, MyMath.distanceToInt(2.5));
}

@Test
public void testDistanceToIntWhole() {
    assertEquals(0, MyMath.distanceToInt(12));
}

@Test
public void testDistanceToIntNegativeWhole() {
    assertEquals(0, MyMath.distanceToInt(-8));
}
```

```java
// Asserts that two double values are equal.
assertEquals(double expected, double actual)
```