Ollscoil Mhá Nuad
Maynooth University

SEMESTER 2
2024-2025

CS608
Software Testing

Asst. Prof. G. Di Liberto, Dr. A. Mooney, Dr. S. Brown

Time allowed: 3 hours

Answer at least ***three*** questions

Your mark will be based on your best ***three*** answers

All questions carry equal marks

Instructions

	Yes	No	N/A
Formulae and Tables book allowed *(i.e. available on request)*		X	
Formulae and Tables book required *(i.e. distributed prior to exam commencing)*		X	
Statistics Tables and Formulae allowed *(i.e. available on request)*		X	
Statistics Tables and Formulae required *(i.e. distributed prior to exam commencing)*		X	
Dictionary allowed *(supplied by the student)*		X	
Non-programmable calculator allowed		X	
Students required to write in and return the exam question paper		X	

Use the test techniques as demonstrated during the module in answering this exam paper.

[25 marks]

1 (a) Given the method [5 marks]

```
long greater(long a, long b)
```

which returns the larger of the two values a and b, explain in detail why *exhaustive testing* is infeasible. Note: if a is equal to b, it returns that value.

(b) For the method `Climate.determine()` as defined below, identify the natural and specification-based ranges for `temp` and `humidity` using value lines. [5 marks]

Provide a table in your answer defining all the input and output partitions, including errors.

```
public static Climate.required determine(int temp,
                                         int humidity)

   Determine whether heating and/or air-conditioning is
   required for a building climate control system. Heating is
   required if the temperature is below 16°C. Air conditioning
   is required if the humidity is above 60%.

Parameters:
   temp - the current temperature (°C)
   humidity - the current humidity (%)
Returns:
   ERROR if the temperature is not within -130..60
      degrees or the humidity is not within 0..100 precent
   Otherwise, return:
      NONE if no heating or AC is required
      HEAT_ONLY if only heating is required
      AC_ONLY if only air conditioning is required
      HEAT_AND_AC if both are required

Climate.Required is defined as:
   enum Required {NONE,HEAT_ONLY,AC_ONLY,HEAT_AND_AC,ERROR}
```

(c) Develop Boundary Value Analysis (BVA) tests for the method `determine()` as defined above. Do not include error values. You answer should include tables for: the test coverage items (TCIs), selected equivalence values, and the test cases. Clearly identify test cases required to cover the expected output TCIs. Complete the TCI table, showing that each test case is covered. Make sure you have no duplicate coverage in your test cases. [15 marks]

[25 marks]

2 (a) The method **Filestore.decideWrite()** decides whether to write a file to a cloud-based filestore as defined below (Javadoc). [15 marks]

```
public static Boolean decideWrite(boolean enabled,
                                  boolean exists,
                                  boolean overwrite)

   Decide whether to write a file to the file store. Only do so
   if the function is enabled, and either the file does not
   exist already and overwrite is false, or the file does exist
   already and overwrite is true.

Parameters:
   enabled - whether write is enabled for this storage location
   exists - whether a file with the same name already exists
   overwrite - whether to overwrite an existing file
Returns:
   true – write (or overwrite) the file to the filestore
   false – do not write the file to the filestore
```

The method has already been tested using EP Black-Box test techniques. The White-Box coverage (JaCoCo) achieved by these tests is shown below. Lines 28 and 32 are yellow. Line 33 is red. All the other lines are green. Hovering on the diamonds on lines 28 and 32 shows "1 of 2 branches missed".

```
24.   public static boolean decideWrite(boolean enabled, boolean exists, boolean overwrite) {
25.       boolean temp=false;
26.       if (enabled) {
27.           if (exists) {
28.               if (overwrite)
29.                   temp=true;
30.           }
31.           else
32.               if (overwrite)
33.                   temp=false;
34.               else
35.                   temp=true;
36.       }
37.       else
38.           temp=false;
39.       return temp;
40.   }
```

Develop the **additional** tests required to provide full branch coverage (BC) for this method. In your answer make sure to: (a) clearly identify the untaken branches, including any 'null else' statements; (b) identify and explain the conditions for the inputs required to take each untaken branch, (c) provide a completed Test Coverage Items table, and (d) provide a Test Cases table.

QUESTION 2 CONTINUES ON THE NEXT PAGE

QUESTION 2 CONTINUED

(b) An optimized version of decideWrite has been developed: `optimisedDecideWrite()`, with it's own set of tests. The test coverage is shown below – line 51 is yellow and line 52 is red. The other lines are green. [5 marks]

```
49.   public static boolean optimisedDecideWrite(boolean enabled, boolean exists, boolean overwrite) {
50.       if (enabled) {
51.           if (exists && !overwrite)
52.               return false;
53.           if (!exists && overwrite)
54.               return false;
55.           return true;
56.       }
57.       return false;
58.   }
```

Hovering on the diamond on line 51 brings up the popup message "1 of 4 branches missed."

```
49.   public static boolean optimisedDecideWrite(
50.       if (enabled) {
51.           if (exists && !overwrite)
52.               return false;
53.           1 of 4 branches missed.  && overwrite)
54.               return false;
```

Explain this message, making sure to clearly identify the 4 branches. Refer to short-circuit evaluation bytecode-level branch coverage in your answer.

(c) Summarise four key differences between black-box and white-box testing, and explain why is it usual to do black-box testing first followed by white-box testing. [5 marks]

[25 marks]

3 (a) Compare 'conventional testing' of the static method `isNeg()` with testing 'in class context' of the method `checkValue()` defined below. Show the order in which the methods are called, and where actual results are compared with expected results. [10 marks]

```
public class Numbers {
   // isNeg() returns true if x<0
   public static boolean isNeg(int x);
   int value; boolean result;
   // setValue() sets the attribute named value to x
   public void setValue(int x);
   // checkValue() determines whether the attribute value<0
   public void isNegative();
   // getResult() returns the result from checkValue()
   public Boolean getResult();
}
```

(b) Develop EP tests 'in class context' for the method `decide()` which determines whether emergency braking is required. [15 marks]

Emergency braking can be overridden by the driver:

- If override is true, `decide`() sets emergencyBraking to false
- Otherwise, `decide`() sets emergencyBraking to true if the danger level is greater than 0, otherwise to false

Develop EP tests 'in class context' for `Braking.decide()`

Analysis: identify the accessor methods for every attribute, develop a value line for dangerLevel, and identify Equivalence Partitions for override, dangerLevel, and emergencyBrake.

Test Design: develop tables for the Test Coverage Items (TCI), data values for dangerLevel, and the Test Cases (with method calls & expected values). Review your completed TCI table.

```
public class Braking {

   private boolean emergencyBrake;
   private boolean override;

   // Get the emergency brake setting set by decide
   public boolean getEmergencyBrake() {
      return emergencyBrake;
   }
   // Set the override flag
   public void setOverride(boolean value) {
      override = value;
   }
   // Decide if emergency braking is required
   public void decide(int dangerLevel) {
     // set emergencyBrake based on dangerLevel and override
   }
}
```

[25 marks]

4 (a) Develop tests for `PV.exportPower()` using random test data. [9 marks]

The method `PV.exportPower()` determines whether power from a PV (PhotoVoltaic) solar cell array is to be exported back into the grid. If the PV system is not enabled, then this always returns false. Otherwise, this returns true if and only if the netPower is greater than 0 (false otherwise)

```
public static boolean exportPower(
       boolean enabled
       int nettPower);
```

Base your random tests on the following Decision Table.

Decision Table for PV.exportPower()

	Rules			
	1	2	3	4
Causes				
enabled	T	T	F	F
nettPower>0	T	F	T	F
Effects				
Return true	T	F	F	F

In your answer:

i. Identify the random value generation criteria for nettPower (in order for the cause nettPower>0 to be true or false)

ii. Complete the provided Random DT Test Cases Table (setting enabled to true or false as in the Decision Table)

Random DT Test Cases Table (to be completed)

ID	**TCI Covered**	**Inputs**		**Expected Results**
		enabled	**nettPower**	
T1	Rule 1	true	rand(Integer.MIN_VALUE,0)	
T2	Rule 2			
T3	Rule 3			
T4	Rule 4			

(b) Provide an outline structure of the automated test code to implement the tests you have defined in part (a). This code need not by syntactically correct, but it must include all the important elements, especially the ranges. Also, show how you can safely generate a random integer between two integer limits. [10 marks]

(c) Identify and describe the three automated random test problems. For each of these problems, show how the random tests you developed in parts (a) and (b) resolves that problem. [6 marks]