OLLSCOIL NA hÉIREANN MÁ NUAD

THE NATIONAL UNIVERSITY OF IRELAND
MAYNOOTH

SUMMER 2022 EXAMINATION

CS608

Software Testing

Prof. O. Conlan, Dr. J. Timoney, Dr. S. Brown

Time allowed: 3 hours

Answer at least ***three*** questions

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

All questions carry equal marks

[25 marks]

Q1 (a) Explain, with reference to the method sum() shown below, why *exhaustive testing* is infeasible. Consider the test design time and the test execution time in your answer. [5 marks]

The method long sum(int x, int y, int z) returns the sum of the three inputs.

(b) A heating system includes a method heatLevel() as defined below to decide what level to set the heating to. Analyse the method heatLevel() in preparation for Boundary Value Analysis testing: [5 marks]

- Identify the natural and specification-based ranges using value lines, treating the values of temp below -100 and above +100 as two separate partitions.
- Provide a table defining all the input and output boundary values.

Level is defined as follows:

```
public enum Level {NONE, LOW, HIGH, ERROR};
```

The method heatLevel() is defined as follows:

heatLevel

```
public static Level heatLevel(boolean disabled, int temp)
```

Determine the heating level required.

Parameters:
`disabled` – whether the heating system is disabled
`temp` – the current temperature (in degrees C)

Returns:
ERROR if the temperature is not within -100 to 100 degrees
If disabled always return NONE
Otherwise, return:

- HIGH if the temperature is below 16 degrees
- LOW if the temperature is between 16 and 24 degrees
- NONE if the temperature above 24 degrees

(c) Develop Boundary Value Analysis (BVA) tests for the method `heatLevel`() as defined above. You answer should include tables for the test coverage items (TCIs) and the test cases. Review your design: complete the TCI table, show that each test case is covered, show that there is no duplicate coverage in your test cases. [15 marks]

[25 marks]

Q2 (a) Compare black-box and white-box testing as follows: what types of errors they are likely to find, what is the impact on tests of changing the source code, must the code be written first, whether each ensures all the source code components have been exercised, and the difficulty of automatically measuring test coverage. [5 marks]

(b) The method Waves.categorise() describes waves by their height. [10 marks]

categorise

`public static Waves.Description categorise(int h)`

Categorise the wave conditions based on the height.

Parameters:

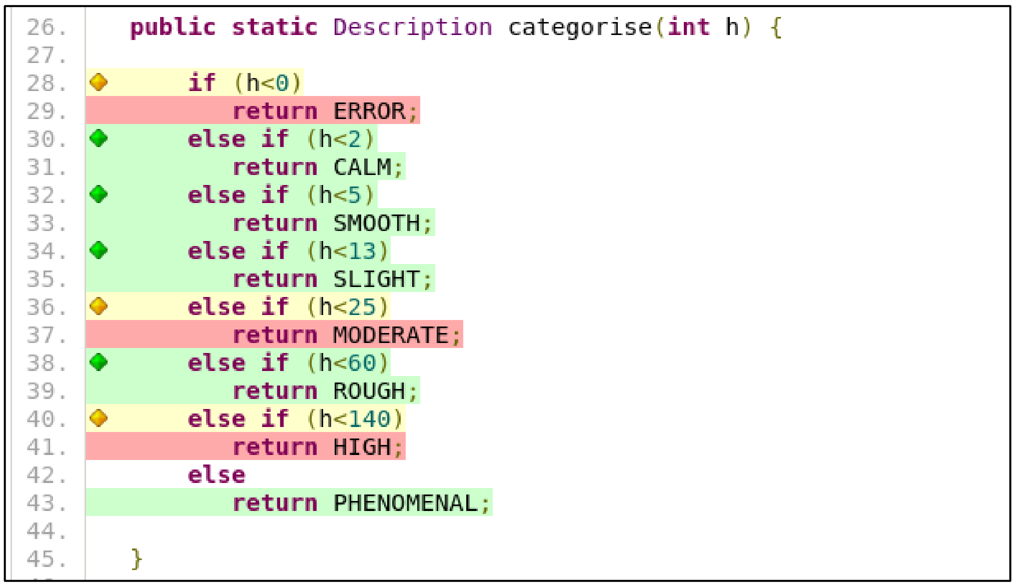
h – the wave height in decimetres (tenths of a metre)

Returns:

ERROR if the height is less than zero
CALM if wave height is greater than or equal to zero and less than 2
SMOOTH if wave height is greater than or equal to 2 and less than 5
SLIGHT if wave height is greater or equal to 5 and less than 13
MODERATE if wave height is greater or equal to 13 and less than 25
ROUGH if wave height is greater or equal to 25 and less than 60
HIGH if wave height is greater or equal to 60 and less than 140
PHENOMENAL if wave height is greater or equal to 140

The following figure (next page) shows the measured coverage of existing black-box tests for the method categorise(). Note that lines 29, 37 and 41 are highlighted in red, and lines 28, 36 and 40 in yellow.

Develop the **additional** tests for the method required to provide full statement coverage (SC). In your answer, include the conditions required to execute each unexecuted line in your analysis, the completed test coverage items table, and the test cases.



```
26.   public static Description categorise(int h) {
27.
28.       if (h<0)
29.           return ERROR;
30.       else if (h<2)
31.           return CALM;
32.       else if (h<5)
33.           return SMOOTH;
34.       else if (h<13)
35.           return SLIGHT;
36.       else if (h<25)
37.           return MODERATE;
38.       else if (h<60)
39.           return ROUGH;
40.       else if (h<140)
41.           return HIGH;
42.       else
43.           return PHENOMENAL;
44.
45.   }
```

(Continued on next page.)

(c) Work out the required input parameter values for `x` and `enable` that ensure execution of lines 7 and 9 in the method inRange() shown below. Clearly identify the conditions required to execute each of the lines, and then show how you determine what input parameter values are required to meet these conditions. [10 marks]

```
 1 public boolean inRange(int x, boolean enable) {
 2    int lower=100; int upper=1000;
 3    if (enable) upper = 2000;
 4    x = x - lower;
 5    upper = upper - lower;
 6    if (v>=0 && v<=upper)
 7       return true;
 8    else
 9       return false;
10 }
```

[25 marks]

Q3 (a) Explain the difference between conventional testing and testing 'in class context'. Include a simple example of each in your answer, clearly showing the differences. [5 marks]

(b) Use the Decision Table provided below to develop DT tests 'in class context' for the method Lighting.decide(). Do not consider error cases. The lighting system can be set to three power settings: 0 is off, 1 is low power lighting, and 2 is full power lighting. The current brightness of the room is measured against two levels: below level1 the room is dark, above level2 the room is bright, and between the two levels (inclusive) the room is dim. [20 marks]

If override is true, then the power setting is always set to full power. Otherwise, if the brightness is below level1, full power is required. If the brightness is above level2, no lighting is required. And between the two levels low power lighting is required.

Use the data values of 100 for limit2, 50 for limit1, and only use valid values >=0 for brightness.

In your answer, identify the accessor methods, provide the DT test coverage items, and show your selected data values for the brightness parameter to decide(). Include a test cases table in your answer, and complete the TCI table to show that you have achieved full coverage of the test coverage items.

Decision Table for decide() where brightness>=0

	Rule					
	1	2	3	4	5	6
Causes						
brightness <= limit2	T	T	T	T	F	F
brightness < limit1	T	T	F	F	F	F
override	T	F	T	F	T	F
Effects						
setting==0	F	F	F	F	F	T
setting==1	F	F	F	T	F	F
setting==2	T	T	T	F	T	F

(Continued on next page.)

Class Lighting

```
public class Lighting {

   private int limit1, limit2, setting;

   // Set brightness level for lighting level 1
   public void setLimit1(int level) {
      limit1 = level;
   }

   // Set brightness limit for lighting level 2
   public void setLimit2(int level) {
      limit2 = level;
   }

   // Get the lighting setting set by decide
   public int getSetting() {
      return setting;
   }

   // Calculate setting
   public void decide(int brightness,
                      boolean override) {
      … code not required …
      … to answer the question …
   }

}
```

[25 marks]

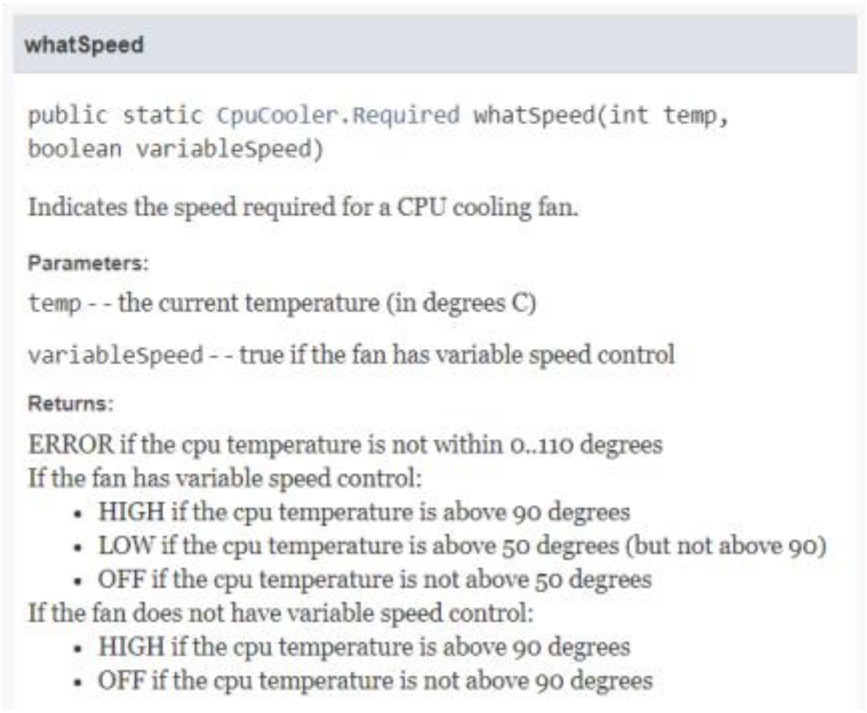
Q4 (a) Briefly describe the following key problems in fully automated random testing, making sure to explain why each problem is a barrier to fully automated random testing: [6 marks]

(i) the test data problem
(ii) the test oracle problem
(iii) and the test completion problem.

(b) Develop random tests for the method whatSpeed() based on the provided non-error EP Test Coverage Items and Test Cases. [19 marks]

In your answer:
(i) Identify the criteria for the input parameter values for each non-error EP.
(ii) Produce a table of the Random EP Test Cases.
(iii) Provide an outline what the automated test code.
(iii) Describe how this resolves two of the random test. problems: the test oracle and selecting test data.

Specification for whatSpeed():



whatSpeed

```
public static CpuCooler.Required whatSpeed(int temp,
boolean variableSpeed)
```

Indicates the speed required for a CPU cooling fan.

Parameters:

`temp` - - the current temperature (in degrees C)

`variableSpeed` - - true if the fan has variable speed control

Returns:

ERROR if the cpu temperature is not within 0..110 degrees
If the fan has variable speed control:
- HIGH if the cpu temperature is above 90 degrees
- LOW if the cpu temperature is above 50 degrees (but not above 90)
- OFF if the cpu temperature is not above 50 degrees

If the fan does not have variable speed control:
- HIGH if the cpu temperature is above 90 degrees
- OFF if the cpu temperature is not above 90 degrees

(Continued on next page.)

Equivalence Partition Test Coverage Items (non error only):

TCI	**Inputs**	**Expected Results**
EP2	temp	0..50
EP3		51..90
EP4		91..110
EP6	variableSpeed	true
EP7		false
EP9	Return Value	OFF
EP10		LOW
EP11		HIGH

Equivalence Partition Test Cases (non error only):

ID	TCI	Inputs		Expected
	Covered	temp	variableSpeed	Results
T1	EP2,6,9	25	true	OFF
T2	EP3,6,10	75	true	LOW
T3	EP4,7,11	100	false	HIGH