EECS 4315 3.0 Mission Critical Systems

A solution to Quiz 2

9:00–10:00 on January 21, 2019

1 (4 marks)

Consider the API of the class `PowerOfTwo` that can be found at the end of this quiz.

(a) How many "inputs" does the method `compareTo` have?

Answer: Two: the `PowerOfTwo` object on which the method is invoked and the `PowerOfTwo` object provided as an argument.

Marking scheme: 1 mark for the correct answer.

(b) Approximate the number of different combinations of "inputs." Explain your answer (marks are only assigned for your explanation). Recall that there are 2^{32} different integers.

Answer: Recall that there are 2^{32} different integers. Of these, $2^{31} - 1$ are positive. Hence, there are 31 different `PowerOfTwo` objects. Therefore, there are $31 \times 31 = 961 \approx 1000$ different combinations.

Marking scheme: 1 mark for a reasonable approximation of the number of `PowerOfTwo` objects. 1 mark for a reasonable approximation of the number of combinations.

(c) Can we check all different combinations of "inputs" in a reasonable amount of time (for example, one minute). Explain your answer (marks are only assigned for your explanation).

Answer: Yes. One can do at least 10^8 operations within one minute, hence, one can easily check all combinations.

Marking scheme: 1 mark for observing how many operations can be performed in one minute.

2 (2 marks)

A JUnit class usually contains the statement

```
import org.junit.jupiter.api.Test;
```

(a) What type of entity is `Test`?

Answer: An annotation.

Marking scheme: 1 mark for annotation or meta data.

(b) How is this entity used in the class?

Answer: To annotation those methods that are tests.

Marking scheme: 1 mark for making the link between methods and tests.

3 (3 marks)

Consider the following test case.

```
private Color c1;
private Color c2;

public void test1() {
  for (int r = -128; r <= 127; r++) {
  for (int g = -128; g <= 127; g++) {
  for (int b = -128; b <= 127; b++) {
  c1 = new Color((byte) r, (byte) g, (byte) b);
  c2 = new Color((byte) (r + 1), (byte) (g + 1), (byte) (b + 1));
  Assertions.assertFalse(c1.equals(c2));
  }
  }
  }
}
```

Although the above test case is correct, it can be improved. Mention three things, related to style, that can be improved.

1.

2.

3.

Answer:

- Make the attributes `c1` and `c2` local variables.
- Use descriptive names instead of `c1` and `c2`.
- Use a descriptive name instead of `test1`.
- Introduce constants for the magic numbers `-127` and `128`.
- Add a string as an argument of the invocation of `assertFalse` to describe the error.
- Indent the code properly.

Marking scheme: 1 marks for each correct answer.

4 (3 marks)

(a) Give an example of a bug that gave rise to loss of life.

Answer: For example, Therac-25 or Tesla.

Marking scheme: 1 mark for a correct example.

(b) Give an example of a bug that gave rise to a significant financial loss.

Answer: For example, Pentium bug or Ariane 5.

Marking scheme: 1 mark for a correct example.

(c) Give an estimate of the (yearly world-wide) cost of debugging.

Answer: US $312 billion.

Marking scheme: 1 mark for something in the order of 100's of billions of US $.

5 (2 marks)

Explain the difference between verification and validation.

Answer: In verification, we address the question "have you made what you were trying to make?" That is, does the code satisfy all the properties of its specification. In validation, we address the question "have you made the right thing?" That is, is the specification capturing the system correctly.

Marking scheme: 1 mark for a correct characterization of verification and 1 mark for a correct characterization of validation.

6 (2 marks)

Nondeterministic code is code that, even for the same input, can exhibit different behaviors on different runs, as opposed to deterministic code. Mention two phenomena that give rise to nondeterministic code.

1.

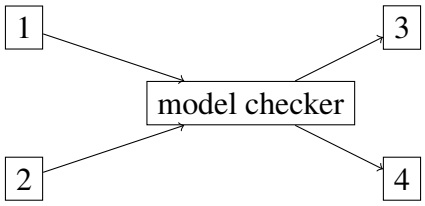
2.

Answer: Randomization and concurrency.

Marking scheme: 1 mark for randomization and 1 mark for concurrency.

7 (3 marks)

The following diagram summarizes model checking.



Fill in the missing labels.

1.

2.

3.

4.

Answer:

1. code
2. property
3. yes
4. no

Marking scheme: 1 mark for code (or model), 1 mark for property (or specification) and 1 mark for yes and no (or true and false).

8 (4 marks)

(a) Mention two advantages of model checking (in comparison with theorem proving).

1.

2.

Answer:

- The approach is automatic.
- The approach is fast.

- The approach provides counterexamples if a property is not satisfied.
- The approach can dealt with partially defined systems and, hence, can be used in the design phase.
- Temporal logic, which is used by many model checkers, can easily express many properties of interest.

Marking scheme: 1 mark for each correct advantage.

(b) Mention two disadvantages of model checking (in comparison with theorem proving).

1.
2.

Answer:

- The state space explosion problem.
- Proving a program correct helps improve the understanding of the program.
- Temporal logic specifications are ugly.
- Writing specifications is hard.

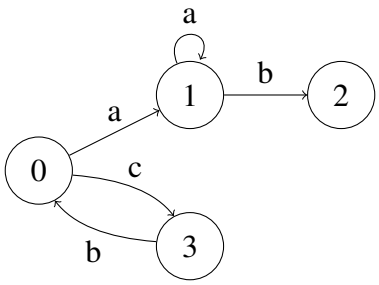
Marking scheme: 1 mark for each correct disadvantage.

9 (4 marks)

A labelled transition system is a tuple $\langle S, A, \rightarrow, s_0 \rangle$ consisting of

- a set S of states,
- a set A of actions,
- a transition relation $\rightarrow \subseteq S \times A \times S$, and
- a start state $s_0 \in S$.

Consider the following model, where 0 is the initial state.



Give the corresponding labelled transition system.

Answer: $\langle\{0,1,2,3\},\{a,b,c\},\{(0,a,1),(1,a,1),(1,b,2),(0,c,3),(3,b,0)\},0\rangle$

Marking scheme: 1 marks for the correct set of states, 1 mark for the correct set of actions, 1 mark for the correct set of transitions, and 1 mark for the correct initial state.