COMP 587
Spring 2020

Midterm Review

The following review, **in conjunction with your assignments and the in-class handouts**, are intended to be comprehensive for the midterm exam. The questions below are representative of the sort of questions you'l see on the exam.

Test Planning

1.) Name one advantage of Attributes, Components, Capabilities (ACC) over traditional test planning.

2.) Name one advantage of traditional test planning over ACC.

3.) A startup developing a mobile app for rapid frozen yogurt delivery is interested in formulating a test plan for their software. Should traditional test planning or ACC-based planning be preferred? Why or why not?

4.) A space agency is developing a real-time operating system for a lunar rover. Should traditional test planning or ACC-based planning be preferred? Why or why not?

5.) Consider the following partial ACC plan for an app to find someone to walk a dog immediately:

Attributes:
Intuitive: the interface is easy to use
Fast: walkers can be found quickly
Robust: the app does not crash and isn't prone to glitches

Components:
Map: users can see a map of available dog walkers nearby
Reservation Creator: users can reserve a walker
Reservation Manager: users can see past, present, and future walker reservations
Configuration: users can configure payment information and basic app behaviors

Capabilities:
Reservation Creator is Fast: walkers can be found within 20 seconds
Reservation Creator is Intuitive: users can reserve a walker with one press
Map is Intuitive: the map's color scheme is easy to interpret

There are some potential issues with this test plan. Provide some feedback regarding how realistic this plan is.

Writing Unit Tests

The questions below use Python. You may not be familiar with Python; that's ok. For our purposes, I will not take off for syntactic things. You can even write pseudocode, as long as it's detailed enough.

6.) Consider the following Python code:

```
def cap(value):
  if value < 0:
    return -1
  elif value == 0:
    return 0
  else:
    assert value > 0
    return 1
```

Using `assert`, write some tests which will collectively get 100% line coverage of `cap`.

7.) Consider the following Python code and corresponding tests:

```
def decToZero(value, limit):
  while value > 0 and limit > 0:
    value -= 1
    limit -= 1
  return value

assert decToZero(10, 100) == 0
assert decToZero(0, 10) == 0
```

7.a.) While the above tests get 100% line coverage (and maybe even 100% branch coverage depending on what we consider a branch), they arguably don't cover everything. What's missing?

7.b.) Write additional tests to cover the missing behavior identified above.

8.) How can tests be used as a specification of a system?

9.) It's possible to get 100% coverage without ever doing any meaningful testing. Explain a scenario under which this can happen.

10.) You have a 100 line program with 1,000 unique tests. All these tests pass with 100% coverage on every metric. Additionally, you use mutation testing/analysis, and are able to show that all mutants are killed. Is the program correct? Explain.

11.) Mutation testing/analysis is generally considered very expensive. Why?

Mocking and Testability

12.) Consider the following Java code, taken from a larger system:

```
public class Something {
  public int x = 0;

  public void doOperation(int input) {
    x += input;
    System.out.println("Output: " + input);
    OtherThing.value = x;
  }
}
```

12.a.) Is `doOperation` testable? Why or why not?

12.b.) What might you do to make `doOperation` more testable?

Automated Testing

13.) You are developing a system which is to undergo verification later. That is, it will later be *proven* correct. With this in mind, is it worthwhile to test the system while under development? Why or why not?

14.) What's the difference between random and bounded-exhaustive testing?

15.) What is the small scope hypothesis? Why is it only a hypothesis?

16.) When is random testing most appropriate?

17.) When is bounded-exhaustive testing most appropriate?

18.) You're developing a system which is to be tested using automated testing techniques. Should you bother with unit tests? Why or why not?

19.) You have a system which is composed of 1,000 lines of code. Using automated testing techniques, you've subjected the system to 1,000,000 tests, and it has passed all of them. How confident are you that the system is correct? Is there any additional information you'd need to assess this?

20.) Consider the Java code below:

```
public class Node {
  public static Random rand = new Random();
  public final int value;
  public final Node rest;

  public Node(final int value, final Node rest) {
    this.value = value;
    this.rest = rest;
  }

  public static Node genNode() {
    if (rand.nextBoolean()) {
      return null;
    } else {
      return new Node(rand.nextInt(), genNode());
    }
  }
}
```

Node represents a single node in a singly-linked list. null is intended to represent a leaf node. genNode is used for randomly generating nodes for testing purposes.

20.a.) Will `genNode` generate diverse tests? Why or why not?

20.b) What changes could you make to `genNode` to make it generate more diverse tests?