George Mason University

SWE 437 Software Testing and Maintenance J. Offutt

**Final Exam, take-home portion part 1**

Due 5 May, 130pm

Submit via dropbox request as specified below

**Rules**:

1. You must answer these questions individually, without any help
2. You may submit anytime between when the exam is distributed and the day and time it is due
3. You may submit answers as a text file, a PDF file, or a word document
4. You must include your name in your answers
5. Make sure that your responses are clearly labeled as to which question they answer
6. You may not share this exam with anyone but yourself—doing so will be considered an honor code violation and, as a maximum, dismissal from the program

Name:    

**I have not discussed this exam with anyone except the instructor, I will not share the exam with anyone else, and I will destroy all copies, both paper and electronic.**

Signature:    

Submit via the following dropbox request link: **XXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

Answer ten questions from the below list. Choose one sub-question from each of the 10 major questions. You can add your answers inside this word document, or create a separate document. If you create a separate document, label your answers with the same numbers that appear on the exam. That is, label an answer to the “*Web* and *evolutionary design”* question as “*2.2*” Each question is worth 1 point.

1. Answer one of three (1 point)
   1. Answer in **25 words or less**: What is *maintenance debt*?
   2. Answer in **25 words or less**: Why did we say that *software evolution* is a more appropriate term than the traditional *software maintenance*?
   3. Name one advantage and one disadvantage, in terms of maintenance, of more documentation.
2. Answer one of two (1 point)
   1. Answer in **25 words or less**: What is the relationship between *evolutionary design* and *perfect out of the box*?
   2. Answer in **25 words or less**: What is the relationship between the Web and *evolutionary design*?
3. Answer one of two (1 point)
   1. Name **two** things that should be documented to support coding for change.
   2. Answer in **25 words or less**: Explain why programmers should ***not*** optimize their code by hand.
4. Answer one of three (1 point)
   1. A Java method returned the wrong value, because of missing parentheses, causing the program to print the wrong number. From that description, what was the fault and what was the failure?
   2. Suppose a Java program has the expression “*A + B \* C*”, when it should have been “*(A + B) \* C*”. Give two sets of test values for the variables *A*, *B*, and *C*; one that would cause the expressions to have a different answer and one that would cause the expressions to have the same answer.
   3. Answer in **25 words or less**: Explain the difference between a *fault* and a *failure*.
5. Answer one of three (1 point)
   1. Answer in **25 words or less**: Define the terms *validation* and *verification*. Make sure your definitions clarify how they are different.
   2. Answer in **25 words or less**: Define the terms *controllability* and *observability*. Make sure your definitions clarify how they are different.
   3. Answer in **25 words or less**: Define the terms *test requirement* and *coverage requirement*. Make sure your definitions clarify how they are different.
6. Answer one of three questions for the following JUnit test class (1 point)

|  |  |
| --- | --- |
| public class someTest {  private Set mySet = new HashSet();  @After public void tearDown() {  mySet = null;  }  @Test public void test1() {  mySet.add("microsoft");  assertEquals("[microsoft]", mySet.toString());  } | @Test public void test2() {  mySet.add("google");  mySet.add("facebook");  mySet.add("google");  assertEquals("[google, facebook, google]", mySet.toString());  }  @Test public void test3() {  assertEquals(**????**, mySet.toString());  }  } |

* 1. How many times will the *tearDown*() method be run?
  2. Will *test2*() work correctly? Why or why not?
  3. The oracle for *test3*() does not have a value. What is the correct value?

1. Use one of the following terms to answer one of the three questions (1 point)

{ *happy path tests*, *invalid input tests*, *minimum viable product*, *refactoring*, *spike*, *test suite*, *user story* }

* 1. What does it mean to go through a period of intense programming to “get ahead” of the TDD tests?
  2. What is a type of tests that often are **not** designed when designing TDD tests?
  3. What is changing the code to improve a **non-functional** quality, without changing the code’s behavior?

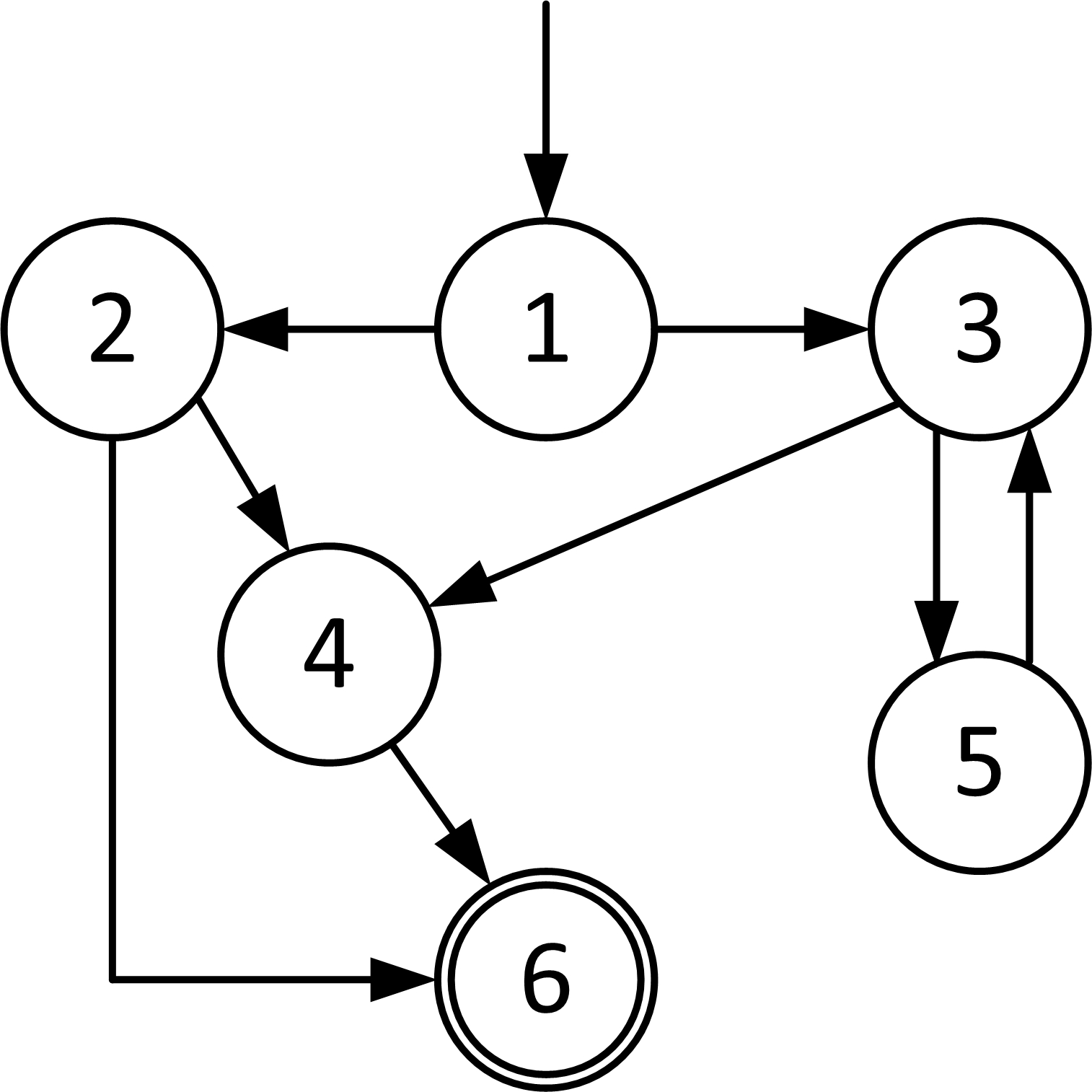
1. Answer one of three questions for ISP for the following test requirements (1 point)

The following table specifies three types of users for a home thermostat and the functions that each is allowed to perform.

|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Owners** | **Guests** | **Admins** |
| A. Set desired temp | **X** | **X** | **X** |
| B. Switch heat to cool | **X** | **X** | **X** |
| C. Enable remote monitoring (using wifi) | **X** |  |  |
| D. Change date & time | **X** |  | **X** |
| E. Connect to wifi | **X** |  |  |
| F. Run diagnostics |  |  | **X** |

* 1. List all test requirements for the test criterion C1: **test each function once**.
  2. List all test requirements for the test criterion C2: **test one function for each user**.
  3. If test criterion C3 says to **test every function for every user**, what coverage level can we achieve if we do not have the administrative password?

1. Answer one of two questions for the following graph (1 point)



* 1. List all the **edge-pairs** for this graph.
  2. List all the **prime paths** for this graph.

1. Answer one of three (1 point)
   1. Give truth assignments to satisfy CACC on the predicate: *(a & b & !c) | (b & c)*
   2. Give truth assignments to satisfy CACC on the predicate: *(a | !b | c) & (b | c)*
   3. Give truth assignments to satisfy CACC on the predicate: *(a & b) | (a & c) | (b & c)*