Final Exam

CS 427 Software Engineering I

TIME LIMIT = 3 hours
COVER PAGE + 6 PAGES

Upon receiving your exam, print your name and netid neatly in the space provided above; print your netid in the upper right corner of every page.

This is a closed book, closed notes examination. You may not use calculators or any other electronic devices. Any sort of cheating on the examination will result in a zero grade. We cannot give any clarifications about the exam questions during the test. If you are unsure of the meaning of a specific question, write down your assumptions and proceed to answer the question on that basis.

Do all the problems in this booklet. Do your work inside this booklet, using the back of pages if needed. The problems are of varying degrees of difficulty so please pace yourself carefully, and answer the questions in the order which best suits you. Answers to essay-type questions should be as brief as possible. If the grader cannot understand your handwriting you may get 0 points.

Part One: ________/30

Part Two:
White Box Testing ________/15   Total Score ________/100
Black Box Testing ________/20

...
Name _____________________________

Part One  (2 points each)
The first section of the test is multiple choice or true/false. Once you determine an answer, draw an X or fill in the box on the answer sheet (on the first page). The five columns (a-e) for the first 11 questions represent the multiple-choice answers. For 12-15, check the T (true) or F (false) column.

Choose the word on the right for the definition/description.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Concerns about whether we are building the right product. (d) a. Include/inclusion
2. Denotes a multi-step subtask in a use case diagram. (a) b. Dependency
3. Denotes a sub-task that is a similar but more general way of accomplishing the subtask in a use case diagram. (c) c. Extend/Extension
   d. Validation
   e. Verification

4. HttpUnit is used to for _______ testing; JUnit is used for ____________ testing.
   a. white box, customer acceptance
   b. black box, customer acceptance
   c. **system, white box**
   d. system, integration
   e. white box, smoke
5. The adapter design pattern is used to:
   a. provide the ability for an object to change its behavior in response to internal state changes
   b. ensure that a class has only one instance and provide a global point of access to it
   c. **convert the interface of a class to an interface that clients expect**
   d. define a many-to-one dependency between objects so that when the one object changes state, all its dependents are notified and updated
   e. define an interface for creating an object, but let subclasses decide which class to instantiate

6. Validating requirements entails checking the requirements’ several different aspects, such as
   a. consistency, completeness, schedulability and feasibility
   b. correctness, consistency, maintainability and cohesion
   c. correctness, consistency, completeness and reliability
   d. **correctness, consistency, completeness and feasibility**

7. A functional requirement does **not** describe the following:
   a. outputs that the system should react to inputs
   b. actions that the system should carry out in particular situations
   c. **response-time constraints on services provided by the system**
   d. services that the system should provide

8. In order to unit test incomplete programs, programmers can build __________. This involves writing __________ (dummy programs that perform simple tasks such as minimal data manipulation or print messages) and __________ (code that passes test data to a component being test).
   a. Scaffolding, driver, stub
   b. Driver, stub, scaffolding
   c. Stub, scaffolding, driver
   d. **Scaffolding, stub, driver**
   e. Stub, driver, scaffolding

9. JUnit is used primarily for _______ testing.
   a. black box
   b. **white box**
   c. customer acceptance test
   d. system

10. ____________ should not be done by the programmer; ________ testing should be done by the programmer.
    a. unit, system
    b. unit, integration
    c. integration, unit
    d. **system, unit**
    e. system, acceptance

11. Which white-box testing guideline is the **easiest** to achieve?
    a. 100% decision coverage
    b. **100% method coverage**
    c. 100% statement coverage
    d. 100% condition coverage

12. Which is **not** a role in an inspection?
    a. Moderator
    b. Scribe
    c. Reviewer
    d. Author
    e. **Manager**

13. Which statement expresses the essence of when the Strategy pattern should be used?
    a. Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and the algorithm changes automatically.
    b. **Encapsulate a request as an object, thereby letting you parameterize clients with different requests.**
c. Allows an object to alter its behavior when its internal state changes.
d. Define a family of algorithms, encapsulate each one, and make them interchangeable so that the algorithm can vary independently from the clients that use it.
e. Represent an operation to be performed on the elements of an object structure; allow the definition of new operations without changing the classes of the elements on which it operates.

14. Which statement expresses the purpose of test oracles?
   a. Store a specified input of a test in a test database
   b. Specify the expected output for a specified input of a test
   c. Invoke a module under test
   d. Test interaction between modules

True/False (2 points each)

For each of the sentences below, mark an X in the appropriate box on the answer sheet. Naturally, mark an X in the T box if the statement is true and an X in the F box if the statement is false.

15. Condition and branch coverage are only different when dealing with compound predicates. [T]
White Box Testing [15 Points]

Consider the following method and answer the three questions below:

```java
int creditLimit(String year, double GPA, int age) {
    int limit = 2000;
    if (year == "FR" || age < 18) {
        limit = limit - 1000;
    }
    limit = limit + 1500;
    if (GPA < 3.0) {
        limit = limit - 500;
    }
    return limit;
}
```

a. Write JUnit-style asserts for test cases for covering all branches AND all conditions.
   Note: write each newly covered branch and conditions followed by the JUnit-style assert such as 3F, …:
   `assertEquals(creditLimit (...), ...) where you should replace … with the actual test input, or expected output. A newly covered branch by the test case should be denoted with the line number postfixed by (T) or (F) indicating True or False, respectively. A newly covered condition should be denoted with the condition expression postfixed by (T) or (F) indicating True or False, respectively.

There can be multiple ways of writing these test cases (different orders or different test cases). Below are one such example test case sequence.

3(T), 6(T)
year == "FR"(T), GPA < 3.0(T)
`assertEquals(creditLimit ("FR", 2.8 [<3.0], 19 [doesn’t matter]), 2000)`

year == "FR"(F), age < 18(T)
`assertEquals(creditLimit ("SN" != FR], 2.8 [<3.0], 17 [<18]), 2000)`

3(F)
age < 18(F)
`assertEquals(creditLimit ("SN" != FR], 2.8 [<3.0], 19 [>=18]), 3000)`

GPA < 3.0(F)
`assertEquals(creditLimit ("SN" != FR], 3.5 [>=3.0], 19 [>=18]), 3500)`
Black Box [20 Points]

Consider the following requirement for a user management system through which a user can open an account.

Write five black box test cases to test the functionality of this requirement (obviously there are many more than five possible test cases). For each test, say what “strategy” you are following with this test case considering the various test case strategies we discussed in class.

The system shall allow a user to create his or her userid and password. The userid is a 9-digit number. The user assigns a password of between 6-10 (inclusive) alphanumeric characters.

### Examples

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Description</th>
<th>Expected Results</th>
<th>Actual Result</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>Userid: 805102789 Password: edgecliff</td>
<td>User created</td>
<td></td>
<td>Equivalence class (based on userid and password)</td>
</tr>
<tr>
<td>BadUserId1</td>
<td>Userid: 80510876 Password: edgecliff</td>
<td>Error, userid/password not established</td>
<td></td>
<td>Boundary analysis (based on userid)</td>
</tr>
<tr>
<td>BadUserId2</td>
<td>Userid: 8051021987 Password: edgecliff</td>
<td>Error, userid/password not established</td>
<td></td>
<td>Boundary analysis (based on userid)</td>
</tr>
<tr>
<td>BadPassword1</td>
<td>Userid: 805102343 Password: muchtoolong</td>
<td>Error, userid/password not established</td>
<td></td>
<td>Boundary analysis (based on password)</td>
</tr>
<tr>
<td>BadPassword2</td>
<td>Userid: 805102346 Password: short</td>
<td>Error, userid/password not established</td>
<td></td>
<td>Boundary analysis (based on password)</td>
</tr>
</tbody>
</table>

5 test cases, 1 point each for unique id, description, expected results, strategy. Description must be repeatable and specific.