Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2018-10-22
Sal	
Tid	
Kurskod	TDDD04
Provkod	
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	
Antal sidor på tentamen (inkl. försättsbladet)	6
Jour/Kursansvarig	Lena Buffoni
Telefon under skrivtid	
Besöker salen ca kl.	
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund
Tillåtna hjälpmedel	Ordbok

LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
Lena Buffoni

Written exam

TDDD04 Software Testing

2018-10-22

Permissible aids
Dictionary (printed, NOT electronic)

Teacher on duty

Instructions and grading
You may answer in Swedish or English.

Your grade will depend on the total points you score on the exam. This is the grading scale:

Grade	3	4	5
Points required	50%	67%	83%

Important information: how your answers are assessed

Many questions indicate how your answers will be assessed. This is to provide some guidance on how to answer each question. Regardless of this it is important that you answer each question completely and correctly.

Several questions ask you to define test cases. In some cases you are asked to provide a minimal set of test cases. This means that you can't remove a single test case from the ones you list and still meet the requirements of the question. Points will be deducted if your set of test cases is not minimal. (Note that "minimal" is not the same as "smallest number"; even when it would be possible to satisfy requirements with a single test case, a set of two or three could still be minimal.)

You may find it necessary to make assumptions in order to solve some problems. In fact, your ability to recognize and adequately handle situations where assumptions are necessary (e.g. requirements are incomplete or unclear) will be assessed as part of the exam. If you make assumptions, ensure that you satisfy the following requirements:

- You have documented your assumptions clearly.
- You have explained (briefly) why it was necessary to make the assumption.

Whenever you make an assumption, stay as true to the original problem as possible.

You don't need to be verbose to get full points. A compact answer that hits all the important points is just as good – or better – than one that is long and wordy. Compact answers also happen to be quicker to write (and grade) than long ones.

Please double-check that you answer the entire question. In particular, if you don't give a justification or example when asked for one, a significant number of points will always be deducted.

1. **Integration Testing (10p)**
    a. Explain the difference between decomposition based and call based integration and give an example of an integration strategy for each. (4p)
    b. How many stubs need to be written for bottom up integration of software a module composed of three sub-modules? Justify your answer. (2p)
    c. Explain what MM-path integration is and give two advantages and one disadvantage of this approach over bottom-up integration. (4p)
2. **State transition testing (12p)**
   You are asked to test the following car sharing application: "Every time a driver set-up a trip, a listing is sent to the website for the trip details (date, time, starting point, destination) and number of available places in the car (3 passengers maximum). Users can book one or more places in the car in one booking. If the number of places requested is more than the number of places available, the booking is rejected. Once there are 0 places left in the car or the trip date has passed, the booking becomes unavailable"
    a. Draw a state-transition diagram for the application. If you make any assumptions about the model state them clearly (6p)
    b. Based on this generate a set of test cases providing transition coverage (4p)
    c. Is path coverage possible for this application? Justify your answer. (2p)
3. **Model Checking (6p)**
    a. Give the definitions of **sound** and **complete** algorithms for program verification. (2p)
    b. Is it possible to build an algorithm for program verification that is both sound and complete? Justify your answer. (2p)
    c. Give two examples of correctness properties that can be verified by a model checking algorithm (2p)
4. **Control flow testing (14p)**
    a. For the code below, determine a set of basis paths (6p)
    b. From the given set of basis baths deduce a set of test cases that provide decision coverage (4p)
    c. How does this set need to be modified to provide condition coverage? (2p)
    d. Is this set of basis paths unique? Justify your answer. (2p)

```
    public int rollDice(int dice1, int dice2, int bonusDice){
        int winPoints;

        if(dice1 == dice2)
            winPoints = dice1*4;
        else if(dice1==6||dice2==6)
            winPoints = 12;
        else
            winPoints = dice1 + dice2;

        if(winPoints > 10 || bonusDice==6)
            winPoints += bonusDice;

        return winPoints;
    }
}
```

5. **Defect classification (10p)**
    a. You are asked to classify the following report using the table below (copy it out on paper first). (6p)

"To conform with the General Data Protection Regulation the specification of the HR management tool has been updated to add a requirement that all personnel data stored needs to be encrypted, however during a code review Sam noticed that the interface for this functionality was not implemented in the database module"

    b. Give two ways defect taxonomies can be helpful in testing (2p)
    c. Is defect severity directly related to priority? Justify your answer. (2p)

Fault/Defect	Attribute	Value
	Asset	
	Artefact	
	Effect	
	Mode	
	Severity	

6. **Test Planning 8p**
    a. Explain what data-based coverage criteria mean. Give an example of a testing method for achieving data coverage. (2p)
    b. You are hired as a test team leader in a team that is developing a social application connected to a fitness watch that allows users to interact with each other. The application is being produced for the watch manufacturer in a team that uses continuous delivery and top-down development. Outline the criteria you would use to stop testing for this project and the test strategy to achieve this. To get full points for this question you must motivate your answer. (4p)
    c. Give two scenarios when exploratory testing is preferable over scripted testing (2p)
7. **Software Test Automation (8p)**
    a. What is meant by **sensitive** tests and **robust** tests? Illustrate your answer with an example. (3p)
    b. What are reference tests and when are they useful? (2p)
    c. Give one disadvantage of test automation (1p)
    d. Give two strategies for automated test generation (2p)
8. **Easy Points – True or False (8p)**

You get 1p for each correct answer, 0p for no answer, and -1p for each incorrect answer. However, you cannot get negative points for this question.

    a. Equivalence Class testing is a kind of Boundary Value testing

b. A risk identification taxonomy is helpful in classifying faults
c. Test automation does not remove the need for human analysis of the test results
d. The defect detection percentage of a test suite keeps increasing after the software is released
e. Automated pairwise testing algorithms generate the minimal number of test cases testing all combinations of two variables
f. Using line coverage as a coverage criteria is better than using nothing at all
g. In model based test case generation, concertizing test cases means organizing them into feature sets
h. Test driven development requires regular refactoring of the code