Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2017-10-17
Sal	
Tid	14-18
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	013-28 40 46
Besöker salen ca kl.	15:00
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

2017-10-17

Permissible aids
Dictionary (printed, NOT electronic)

Teacher on duty
Lena Buffoni, tel: 013-28 40 46

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. **Terminology** (8p)
    a. Define the term **test-case** and give **three** of the elements a good test-case must contain. (2p)
    b. Define functional-based and structural-based testing. Give an example of a testing technique for each of them. What types of faults are these techniques useful at detecting? (hint: you can use diagrams to make your explanation clearer) (6p)

2. **State transition testing** (4p)
    a. Order the following test coverage criteria for state transition testing from weakest to strongest: event coverage, path coverage, transition coverage, state coverage. Are all of them always applicable? Justify your answer. (3p)
    b. What type of systems is transition testing best suited for? (1p)

3. **Decision tables** (10p)
   "You need to write a function for the university gym management system that calculates the correct discount rate for a new member signing up. The gym offers different discounts to the members. Students get 15% discount, teachers get 10% discount and two people in the same family can get an additional 5% discount each if both of them sign up."
    a. Draw a decision table for the problem above. If you make any assumptions, state them clearly. (6p)
    b. Generate a set of test cases based on this decision table (4p)

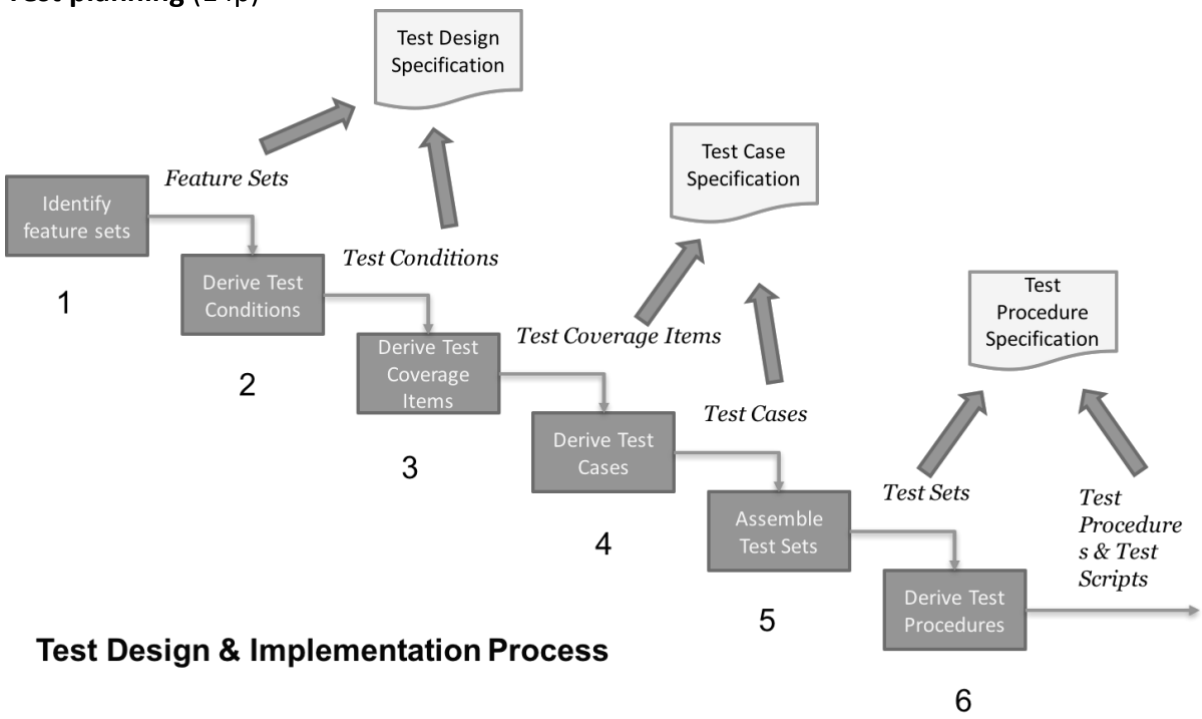
4. **White-box testing** (18p)

```
1   public int strangeFunction(int a, int b){
2       int c;
3
4       if(a<0 || b<0)
5           return 0;
6
7       if(a>b)
8           c = a - b;
9       else if (a==b)
10          c = 4*a;
11      else
12          c = b-a;
13
14  return c;
15   }
```

    a. Calculate the cyclomatic complexity for the code above. (6p)
    b. Determine a basis set of paths for the code (4p)
    c. Provide a set of test cases based on the basis path test set you have generated, choosing values that are interesting in addition to providing you 100% coverage (4p)
    d. Explain the principles of **mutation testing**. Provide two examples of different mutation operators that can be applied to the code above. (4p)

5. **Model-based testing** (6p)
    a. A good model must be precise and concise. Explain what this means in practice. (2p)
    b. Give two possible inconveniences of model-based test-case generation over manual test-case generation (2p)
    c. Describe two of the steps involved in model-based testing (2p)

6. **Test planning** (14p)



Test Design & Implementation Process

The Figure above shows the stages of the test design and implementation process. As a tester you are asked to apply this process to design a test set for the following program:
"A hotel reservation rebooking service takes in the current date and the booking date, plus the membership level of a traveller it then returns the type of modification that can be made (non modifiable, modifiable for free, modifiable at a fee). Premium level customers can change their booking without a fee up to the booking date. Standard level customers can change their booking for free up to two days before the booking date and at a fee up to the day before the booking date. On the day of the booking no modifications can be made."

    a. For stages 1 to 6 describe what needs to be done for the program above and provide a set of test cases based on a coverage criteria of your choice. (12p)
    b. Give one advantage and one disadvantage of scripted testing compared to exploratory testing (2p)

7. **System-level testing** (12p)
You have the following simple random number generator:

```
1 public int[] pseudoRandomGenrator (int seed){
2         int newRandom, newSeed;
3         if(seed < 0)
4                 return null;
```

```
5           newSeed = updateSeed(seed);
6           newRandom = updateRandom(newSeed);
7           return new int[] {newRandom, newSeed};
8   }
9
10  private int updateSeed(int seed) {
11          int newSeed = 7 * seed % 101;
12          return newSeed;
13  }
14
15  private int updateRandom(int seed) {
16          int newRandom = (seed -1)%10 + 1;
17          return newRandom;
18  }
```

a. For the code above build a flow graph representation, select an interesting test-case and provide the list of Module Executions Paths (MEPs). (6p)
b. From the MEPs defined in a. build a MM-path graph (4p)
c. Give an advantage of MM-path testing over top-down and bottom-up approaches for integration. (1p)
d. What are two other kinds of system-level testing apart from functional testing? (1p)

8. **Coverage criteria** (6p)
    a. Give two examples of coverage criteria that can be used in a project. For each of them state one limitation. (4p)
    b. How does automated test-case generation influence the test-suite size as a progress indicator? Explain your answer. (2p)

9. **Easy points** (6p)

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. Path-based testing is both structural and functional based
b. The priority of a fault is a direct consequence of the fault severity
c. Black box testing can be applied on integration level
d. Domain analysis testing can be applied when multiple variables are logically related to each other
e. Additional test cases for the same equivalence class require more effort but increase test-suite effectiveness at finding faults
f. Some faults can be redetected before they become software defects.