Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2018-08-22
Sal	
Tid	
Kurskod	TDDD04
Provkod	
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	8
Antal sidor på tentamen (inkl. försättsbladet)	6
Jour/Kursansvarig	Kristian Sandahl
Telefon under skrivtid	+46 (0)706 68 19 5
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

2018-08-22

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. **Definitions (4p)**
   Explain the notions of white-box and black-box testing and give two examples of each technique.
2. **Decision table testing (10p)**
   You are asked to test a program with the following functionality:
   A billing system sends out a monthly invoice for an electricity company, the total is calculated by the following rules:
   - If the client has a fixed rate subscription, a minimum monthly rate R is charged for all consumption below 100kwh and rate A is applied to every kwh over that.
   - If the client has a flexible subscription, then rate A is applied to every kwh for the first 100 and rate B is applied to every kwh after that.
   - If the client has a subsidy, then 60% of the bill will be covered by the state, so only 40% is charged

   a. Draw a decision table to illustrate how the billing is computed. If you make any assumptions, state them clearly. (6p)
   b. Generate a set of test cases based on this decision table (4p)

3. **Path testing (10p)**
   For the code below:
   a. Calculate the cyclomatic complexity
   b. Determine a set of basis paths
   c. Write a set of test cases that provide path coverage

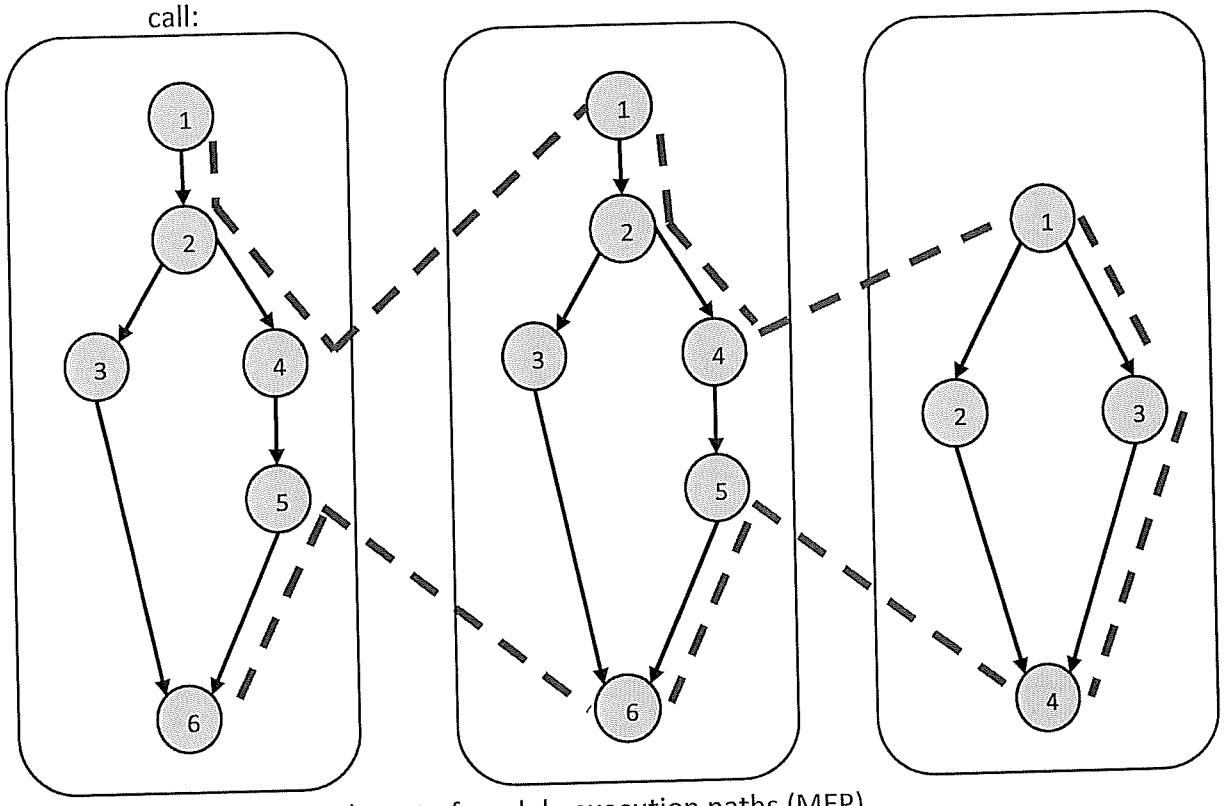
```
public int myFunction (int a, int b){
        String msg = "";
        int diff = 0;

        if (a <0 || b<0){
                diff = -1;
                msg = "error";
        }
        else if (a == 0){
                diff = b;
                msg = "b is bigger by " + diff;
        }
        else if (b == 0){
                diff = b;
                msg = "a is bigger by " + diff;
        }
        if (msg == "")
                return myFunction(a-1, b-1);
        else {
                print(msg);
                return diff;
        }
}
```

4. **Exploration testing (6p)**
   a. Give three elements that can be covered in a test charter
   b. Describe two types of scenarios when exploration testing can be useful
   c. Give one disadvantage of exploration testing
5. **System-level testing (14p)**

a. In top-down integration testing with three levels of decomposition, how many drivers will be needed? Justify.
b. Give two advantages of MM-path testing over top-down integration testing

For the control flow graph below where the dashed line represents the function call:



c. Calculate the set of module execution paths (MEP)
d. Draw a MEP path graph for the dashed call

6. **Defect classification (8 points)**
   a. Does a high defect severity automatically imply high defect priority? Illustrate with an example.
   b. What is the difference between a fault and a defect? Give an example of each.
   c. Give two advantages of using taxonomies for defect classification.
7. **Coverage criteria (6 points)**
   a. Give an example of code for which:
      #tests for statement coverage < #tests for decision coverage < #tests for path coverage
      Justify your answer.
   b. Give two other types of artefacts besides code that coverage criteria can be applied to.
   c. Is full path coverage sufficient to evaluate the quality of a test suite? Justify your answer.
8. **Easy points (6 points)**
   Answer true or false, correct answers give 1p, incorrect answers -1p.

   a. One goal of software testing is to verify that the system under test (SUT) contains no errors.
   b. Unit testing can be used for integration testing

c. Exploratory testing is a repeatable process that can provide measurable, quantitative results on software quality.
d. A minimal test set that achieves 100% path coverage will generally detect more faults than one that achieves 100% statement coverage.
e. MM-path testing can be used for both unit testing and integration testing.
f. Decision-table testing subsumes Model-based testing.