# Försättsblad till skriftlig tentamen vid Linköpings universitet

| | |
|---|---|
| **Datum för tentamen** | 2015-08-19 |
| **Sal (1)** | TER2 |
| **Tid** | 8-12 |
| **Kurskod** | TDDD04 |
| **Provkod** | TEN1 |
| **Kursnamn/benämning**<br>**Provnamn/benämning** | Programvarutestning<br>Skriftlig tentamen |
| **Institution** | IDA |
| **Antal uppgifter som ingår i tentamen** | 11 |
| **Jour/Kursansvarig**<br>Ange vem som besöker salen | Ola Leifler |
| **Telefon under skrivtiden** | 070-1739387 |
| **Besöker salen ca klockan** | 10:00 |
| **Kursadministratör/kontaktperson**<br>(namn + tfnr + mailaddress) | Anna Grabska Eklund, Email:<br>anna.grabska.eklund@liu.se<br>Phone: +46 13 282362 |
| **Tillåtna hjälpmedel** | Dictionary (printed, NOT electronic) |
| **Övrigt** | |
| **Antal exemplar i påsen** | |

LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
Ola Leifler

# Written exam

# TDDD04 Software Testing

# 2015-06-03

**Permissible aids**

Dictionary (printed, NOT electronic)

**Teacher on duty**

Ola Leifler, tel. 070-1739387

**Instructions and grading**

You may answer in Swedish or English.

Your grade will depend on the total points you score on the exam. The maximum number of points is 83. This is the grading scale:

| **Grade** | **3** | **4** | **5** |
|---|---|---|---|
| Points required | 41 | 54 | 68 |

## 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 (4p)

Explain what “test coverage” is. Also explain if it is used during white-box or black-box testing. (4p)

## 2. Coverage criteria (8p)

a) Is there any guarantee that test cases that achieve 100% condition coverage will achieve 100% statement coverage? Justify your answer.

(2p)

b) Explain why 100% statement coverage is insufficient to test the execution paths of a method. Justify your answer with an example that contains at least one decision and one iterative construct. (2p)

c) Explain why 100% path coverage is insufficient for testing an application by referring to properties external to the control flow graph. (4p)

## 3. Test automation (6p)

Describe a suitable test automation technique that may be used in order to

a) reduce the cost of creating test cases. (2p)

b) reducing the cost of executing test cases. (2p)

c) improve the probability that faults resulting from complex user interactions are detected. (2p)

## 4. True/False(6p)

Answer true or false:

a) White-box testing can assess whether a program demonstrates faults of omission.

b) Big-bang testing has no advantages over other forms of integration testing.

c) A fault and a failure are synonymous concepts in Software Testing.

d) Data-flow testing subsumes control-flow testing.

e) Decision-table testing subsumes Model-based testing.

f) You can automate exploratory testing.

(It's not worth guessing: you get 1p for correct answer, 0p for no answer, and -1p for incorrect answer; you can get negative points on this question.)

## 5. Black-box testing (16p)

You are to test an alcohol ignition interlock ("alkolås" in Swedish) of a car, with the following specifications:

The alcohol detector is activated by turning the ignition key from "Off" to "On". The key may not be turned passed "On" to the state "Starting ignition" unless the alcohol detector has cleared the driver. To clear the driver, he or she must blow at least 1l of air into the detector nozzle, and the concentration of alcohol vapour must be less than or equal to 0.25mg/l. If the concentration is below that level, the alcohol lock will allow the ignition key to be turned passed state "On" to "Starting ignition". The information of the outcome of the test must be presented to the driver through a display within 10 seconds from blowing into the nozzle. If the concentration of alcohol vapour is more than 0.25mg/l, the driver may not attempt again before half an hour has passed.

You will be graded on your choice of test method for this specification, your test cases and the degree to which your test cases cover the specifications of the system above.

## 6. White-box testing (10p)

Select an appropriate method for testing the following method, and select an appropriate coverage metric. For full points, you need to take care to select an appropriate type T as well as different values for the parameter `collection`.

```
/**
 * Are all members of a collection equal?
 *
 * @param collection
 *            The <code>Collection</code> to examine.
 * @return True if all members are the same.
 */
public static <T> boolean allSame(final Collection<T> collection) {
        T datum = null;
        boolean first = true;
        for (T t : collection) {
                if (first)
                        datum = t;
                else if (t != datum)
                        return false;
                first = false;
        }
        return true;
}
```

## 7. Model-based testing (6p)

In Model-based testing, which of the following activities are **not** automated?

1. Creation of test cases from a specification of a model
2. Creation of the execution steps that are necessary to execute the model
3. The specification of the model

Justify you answer.

## 8. Integration testing (6p)

a) In thread-based integration testing, do you need drivers, stubs, both or neither? (2p)

b) What information is necessary to calculate the number of *drivers* that are needed in bottom-up integration testing? (2p)

c) How do you determine the number of *test sessions* that are necessary in top-down integration testing? (2p)

## 9. Exploratory testing (6p)

a) Explain why successful exploratory testing requires more domain expertise than other types of testing. (2p)

b) How can *sessions* be used in exploratory testing, and what is the purpose of a session? (2p)

c) Name one instance of an application and development stage when exploratory testing is advisable, and one when it would not be advisable. (2p)

## 10. Modified condition/decision coverage (10p)

Specify a *minimal* set of test cases for the following function that result in *maximal modified condition/decision coverage.*

```
static boolean foo(int x, int y) {
        if (x+y < 20) {
                return true;
        } else if (x < 10 || y < 10) {
                return x + y < 25;
        } else if (x > 15 && y > 15) {
                return false;
        } else {
                return x > 15 || y > 15;
        }
}
```

## 11. Testing case selection (5p)

Combine situations with test case selection methods. Each situation should be matched with exactly one test case selection method.

| Situation | Test case selection method |
|---|---|
| 1 Testing of a partially developed new web application | A Decision-table testing |
| 2 Testing two different GUI:s for the same underlying ftp client software | B State-transition testing |
| 3 Verifying an Ethernet card driver for Linux | C Path-based integration testing |
| 4 Verifying business rules in a social security web application | D Model-based testing |
| 5 Perform user acceptance testing of new functionality | E A/B testing |