Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2013-03-30
Sal	KÅRA
Tid	8-12
Kurskod	TDDD04
Provkod	TEN1
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	13
Antal sidor på tentamen (inkl. försättsbladet)	8
Jour/Kursansvarig	David Byers
Telefon under skrivtid	013-282821
Besöker salen ca kl.	10:15
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	Inga

LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
David Byers

Written exam

TDDD04 Software Testing

2013-03-30

Permissible aids

Dictionary (printed, NOT electronic)

Teacher on duty

David Byers, 013-282821

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 100. The following grading scale is **preliminary** and the limits may be lowered after grading.

Grade	3	4	5
Points required	55	70	85

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. Your set of test cases **must be minimal**. 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 motivation or example when asked for one, a significant number of points will always be deducted.

Question 1: Terminology (4p)

Explain what "white-box testing" is. Briefly explain one test case design methodology that can be used for white-box testing. (4p)

Question 2: Coverage criteria (8p)

a) If a set of test cases achieves branch coverage does it also achieve statement coverage? (Yes/no is sufficient; 2p)

b) If a set of test cases achieves condition coverage, does it also achieve branch coverage? (Yes/no is sufficient; 2p)

c) Explain **either** Multiple Condition Coverage (MCC) **or** Modified Condition/Decision Coverage (MCDC). It is recommended (but not required) that you illustrate your explanation with a simple example. (4p)

Question 3: Termination (8p)

a) For each of the following criteria, state whether you think it is a reasonable criterion for deciding when to finish testing. Briefly motivate your answer to each criterion! (You will be assessed on your answer and motivation.)

   i. All known important defects have been fixed.

   ii. Every combination of inputs has been successfully tested.

   iii. Time and money has run out.

   iv. There is sufficient information to make an informed decision about whether to release the software or not.

Question 4: Easy points (6p)

Answer true or false:

a) The goal of unit testing is to verify that the units in a system interact as expected.

b) System testing is usually done after integration testing.

c) A failure is the observable effect of executing a bug.

d) Code inspections are an effective way to find faults.

e) Boundary value analysis is a structure-based test case design technique.

f) Linear test scripts are robust with respect to changes in the system under test.

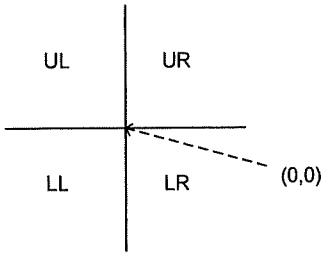
(It's not worth guessing: you get 1p for correct answer, 0p for no answer, and -1p for incorrect answer; you can get negative points on this question.)

Question 5: Equivalence class testing (10p)

A program calculates the absolute value of integers from -100 to 100 (inclusive). All other values are rejected. Specify a set of test cases that cover all equivalence classes. Briefly explain how you came up with the test cases to show that you understand equivalence class testing. (You will be assessed on the quality and completeness of your test cases, as well as your explanation.)

Question 6: Boundary value testing (10p)

A program takes a coordinate value as input (a pair of signed 32 bit integers) and outputs the quadrant in which the coordinate is located, as indicated by the following diagram:



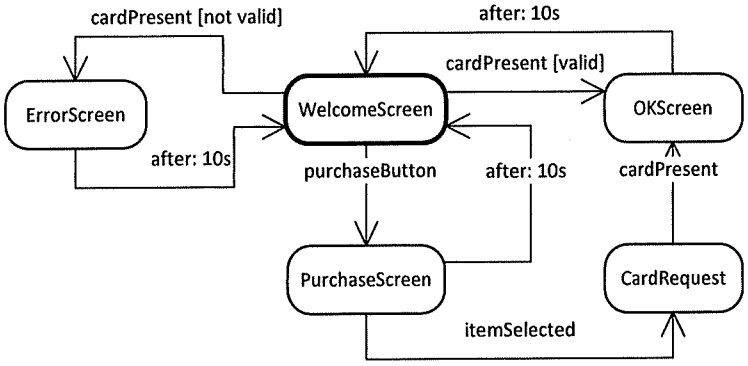
Using boundary value testing, specify a complete set of test cases (i.e. that properly checks all boundaries). Briefly explain how you came up with the test cases to show that you understand boundary value testing. (You will be assessed on the quality and completeness of your test cases, as well as your explanation.)

Question 7: Decision table testing (10p)

Explain what decision tables are and how they can be used in testing. Provide a concrete example of how test cases are constructed from decision tables. Your example must result in at least four test cases, and should include "don't care" entries in the table. (You will be assessed on your explanation as well as the correctness of your example.)

Question 8: State transition testing (10p)

A system is defined using the following statechart. The state **WelcomeScreen** is the quiescent state of the system. Each translation is labeled with the event that causes it to be taken (e.g. "after: 10s") and optionally a condition in square brackets (e.g. "valid").



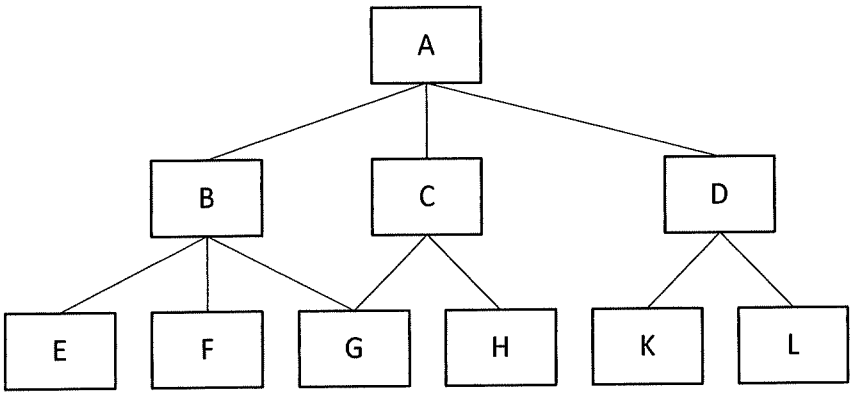
Select an appropriate coverage criterion and specify a set of system-level test cases that satisfy that criterion. Inputs are sequences of events and conditions (the only conditions in the state chart are "valid" and "not valid"). Expected behavior can be defined as a sequence of states and transitions that are traversed.

Motivate your choice of criterion, relating it to at least one other coverage criterion for state transition testing.

(You will be assessed on your choice of criterion; completeness and quality of test cases; and motivation.)

Question 9: Integration testing (8p)

The following figure illustrates the decomposition of a system.



a) In which order should the components be integrated when using a top-down approach? (There are several possibilities; any correct answer is acceptable; 2p)

b) In which order should the components be integrated when using a bottom-up approach? (There are several possibilities; any correct answer is acceptable; 2p)

c) How many stubs are needed for top-down integration? Explain how you came up with that number. (2p)

d) How many drivers are needed for bottom-up integration? Explain how you came up with that number. (2p)

Question 10: Exploratory testing (8p)

a) What distinguishes exploratory testing from ad-hoc testing?

b) State two situations in which exploratory testing is particularly suitable.

c) State and briefly explain two drawbacks or limitations of exploratory testing.

Question 11: Branch coverage (10p)

Specify test cases for the following function that result in 100% branch coverage.

```
int min_or_max(int[] a, int ignore, int mode) {
    int res = Integer.MAX_VALUE;

    if (mode == GET_MAX)
        res = Integer.MIN_VALUE;

    while (pos < a.length) {
        if (a[pos] != ignore)
            if (mode == GET_MAX) {
                if (a[pos] > res)
                    res = a[pos];
            }
            else {
                if (a[pos] < res)
                    res = a[pos];
            }
        }
        pos += 1;
    }
    return res;
}
```

Question 12: Model-based testing (4p)

a) Briefly explain what model-based testing is? (2p)

b) For what kind of testing is model-based testing appropriate? (2p)

Question 13: More easy points (4p)

Answer the following questions true or false:

a) Graphical user interfaces cannot be tested using automated tests.

b) Automated testing can easily be implemented in any organization.

c) Automated testing is particularly useful for finding new bugs.

d) Automated testing prevents bad testing practices.

(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.)