Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2013-08-21
Sal	TER2
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)	9
Jour/Kursansvarig	David Byers
Telefon under skrivtid	013-282821
Besöker salen ca kl.	9:30
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund
Tillåtna hjälpmedel	Inga

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

Written exam

TDDD04 Software Testing

2013-08-21

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. 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 motivation or example when asked for one, a significant number of points will always be deducted.

Question 1: Terminology (4p)

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

Question 2: Coverage criteria (8p)

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

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

c) Explain ALL-P-USES coverage. It is recommended (but not required) that you illustrate your explanation with a simple example. (4p)

Question 3: Code inspections and walkthroughs (8p)

Explain how formal code inspection is organized and perform. Contrast formal code inspections to walkthroughs (i.e. what are the differences and similarities in goals, structure and methods).

Question 4: Easy points (6p)

Answer true or false:

a) The overall goal of testing is to demonstrate that a SUT contains no defects.

b) (Some) unit testing typically occurs before system testing begins.

c) Defect, bug and error are different names for the same thing.

d) Code inspections are not a cost-effective method for finding bugs.

e) Boundary value analysis is a black-box test case design technique.

f) Keyword-driven test scripts typically require considerable maintenance when the SUT changes.

(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)

In the USA, the MPAA has defined a number of film ratings. A program is designed to take the age of a person and return the ratings allowed for that person (upper bounds are not inclusive; lower bounds are inclusive, so e.g. "0-13" means up to but not including 13.

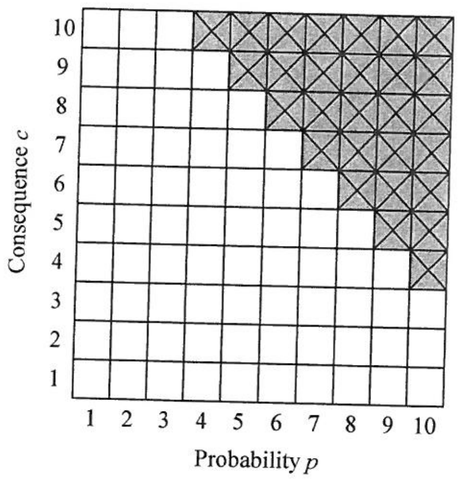
Age	Parent present	Ratings allowed
0-13	No	G
0-13	Yes	G, PG
13-17	No	G, PG, PG-13
13-17	Yes	G, PG, PG-13, R
17-	Yes or No	G, PG, PG-13, R, NC-17

Use equivalence class testing to create a minimal set of test cases for this program. You will be assessed on the completeness and appropriateness of your test cases. Briefly discuss whether equivalence class testing is suitable for this program, and what other black-box methodologies might be more suitable.

Question 6: Domain analysis testing (10p)

In a risk analysis method, the probability *p* and consequence *c* of every risk is estimated on a scale from one to ten. Risks that have sufficiently high probability and/or consequence are subject to risk mitigation.

The shaded/marked cells in the grid below indicate which combinations of *p* and *c* indicate that risk mitigation must be undertaken.



Consider a function that, given values for *p* and *c* is to return whether a risk is subject to mitigation or not (*true* or *false*) – i.e. returns *true* for the shaded/marked cells and *false* for the blank cells.

Use domain analysis to design a suitable set of test cases for this function. The set of test cases should be close to minimal (a few extra test cases is OK, but not many).

Explain your process briefly. It is sufficient to create test cases for the domain where the expected result is *true*. For example, test cases with c=0 or p=0 are not required.

You will be assessed on the quality and completeness of your test cases, as well as your explanation.

Question 7: Cause-effect diagrams (10p)

The following text specifies business rules a company uses to determine which trade credit terms to offer customers:

> *Net-30 terms are offered to existing customers with good history. New customers (i.e. with no credit history) are offered Net-30 terms if the order value is under 10.000 and Net-10 terms if the order value is 10.000 or more. Customers with bad credit history are offered prepay terms. Regardless of the above, blacklisted customers are offered no terms at all.*

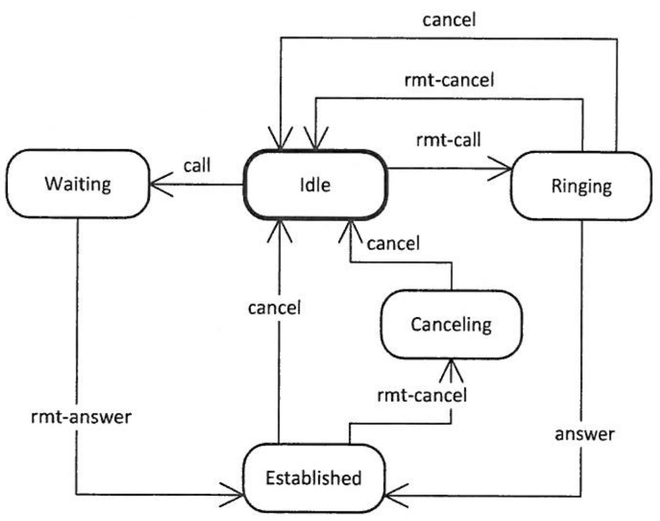
The exact meaning of net-30, net-10 and prepay are irrelevant to solving this problem.

Use cause-effect graphs to model the specification above. Based on these create decision tables and test cases for this system.

You will be assessed on the correctness, completeness and quality of your model, decision table and test cases.

Question 8: State transition testing (10p)

A system is defined using the following statechart. The state **Idle** is the quiescent state of the system. Each transition is labeled with the event that causes it to be taken. There are no conditions in this model.



Select an appropriate coverage criterion and specify a set of system-level test cases that satisfy that criterion. Inputs are sequences of events. 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)

a) Explain what big-bang integration testing is.

b) Is there any situation in which big-bang integration might be appropriate? If so, what situation. Motivate your answer.

c) Integration testing strategies are often based on the hierarchical decomposition of a system. Examples of this include top-down, bottom-up and sandwich testing. Explain an alternative integration strategy (e.g. pairwise, neighborhood, path-based). Illustrate your explanation with an example.

Question 10: Exploratory testing (8p)

a) In terms of exploratory testing, what is a *charter*?

b) In terms of exploratory testing, what is a *tour*?

c) Exploratory testing is frequently said to be suitable only for very experienced programmers. Briefly discuss how exploratory testing can be adapted so that less experienced programmers may also be able to perform testing effectively.

Question 11: Data flow coverage (10p)

Specify test cases for the following function that result in **all-uses** coverage with respect to variables **`bill`** and **`usage`**. The set of test cases **does not** have to be minimal.

Your answer must include a suitable graphical representation of the program that identifies each def and use. Each test case must indicate which DU path it tests.

```
int calculate_bill(int usage) {
  int bill = 0;

  if (usage > 0) {
    bill = 400;
  }
  if (usage > 100) {
    if (usage <= 200) {
      bill = bill + (usage - 100) * 5;
    }
    else {
      bill = bill + 50 + (usage - 200) * 1;
      if (bill >= 100)
        bill = bill * 0.9;
    }
  }
  return bill;
}
```

Question 12: Model-based testing (4p)

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

b) Which is model-based testing more suitable for: functional testing or performance testing? (2p)

Question 13: More easy points (4p)

Answer the following questions true or false:

a) Automated testing can be used to test programs with GUIs.

b) Automated testing is effective for finding previously unknown bugs.

c) Automated testing ensures that good testing practices are followed.

d) Automated testing is appropriate in an agile environment.

(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.)