Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2014-06-03
Sal	T2 / U1
Tid	14-18
Kurskod	TDDD04
Provkod	TEN1
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	12
Antal sidor på tentamen (inkl. försättsbladet)	7
Jour/Kursansvarig	Ola Leifler
Telefon under skrivtid	070-1739387
Besöker salen ca kl.	15:00
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund
Tillåtna hjälpmedel	Inga

Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2014-06-03
Sal	T2 / U1
Tid	14-18
Kurskod	TDDD04
Provkod	TEN1
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	12
Antal sidor på tentamen (inkl. försättsbladet)	7
Jour/Kursansvarig	Ola Leifler
Telefon under skrivtid	070-1739387
Besöker salen ca kl.	15:00
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund
Tillåtna hjälpmedel	Inga

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

Written exam

TDDD04 Software Testing

2014-06-03

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

Grade	3	4	5
Points required	42	58	73

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

2. Coverage criteria (8p)

a) Order the following coverage criteria with respect to their requirements on test cases in ascending order:

1. Decision Coverage
2. Condition Coverage
3. Branch Coverage

(2p)

b) Explain *Modified decision/condition coverage* with a code example (2p)

c) Explain coverage criteria that can be used for loops (4p)

3. Test automation (6p)

When should you use a test automation framework such as CPPUNIT/JUnit instead of

a) stepping through a debugger (2p)

b) writing debug information to the console (2p)

c) running an application and inspecting the results (2p)

4. Easy points (6p)

Answer true or false:

a) One goal of software testing is to verify that the system under test (SUT) contains no errors.

b) MM-Paths can be used for both unit testing and integration testing.

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

d) Requirements reviews are in general better than code inspections at finding faults in applications.

e) Equivalence-class testing subsumes decision-table testing.

f) The control script for data-driven test scripts requires little effort to set up.

(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)

The Swedish Tax Authority Skatteverket has the following rule for paying road user charges (tolls) for foreign heavy vehicles, simplified for this question:

"[The toll] depends on what distance you are intending to drive on a toll road, the emission category of your vehicle (EURO 0, 1 or 2) and the number of axles of the truck or truck/trailer combination. You can pay for day [or week]. For more details, please see the [tariff table below].

Note that the chargeable period for day is 00.00-24.00. So, if a journey on a toll road begins, for example, at 21.30 on one day and finishes at 02.30 next day, payment for two days is required.

Tolls are paid in SEK."

Toll Tariff 2014

Max axles	**3**	**3**	**3**	**4**	**4**	**4**
Emission Class	**0**	**1**	**2 or cleaner**	**0**	**1**	**2 or cleaner**
1 day	67	67	67	67	67	67
1 week	220	194	169	347	313	279

For a hypothetical software system that produces the toll charge given the information above, select a test case design methodology and use it to define a minimal set of test cases (according to that methodology). Justify your choice of methodology! You will be evaluated on both your choice of methodology, justification, and application of the methodology.

6. Basis path testing (10p)

Use *basis path testing* to define test cases for the following programs.

```
int f(int x, int y) {
  bool x_neg;
  if (x > 0) {
    x_neg = false;
  }
  else {
    x_neg = true;
```

```
    x = -x;
  }

  if (y > 0) {
    if (x_neg) {
      y = -(y * y);
    }
    else {
      y = y * y;
    }
  }
  else {
    y = -(y * y);
  }
  return x + y;
}
```

You are required to draw the appropriate graphical representation of the program and illustrate the process of selecting paths. For each test case, indicate which basis path the test case corresponds to, in addition to the other information required for a test case. You will be assessed on the quality and completeness of your test cases, as well as your explanation of the basis path selection process.

7. Code Complexity measures (6p)

Explain *cyclomatic complexity* and *essential complexity*, and their relationship to testing. For full points, provide a code example that describes the relationship between *cyclomatic complexity* and *essential complexity*.

8. Integration testing (6p)

a) State one advantage big-bang integration has over most other methods. (2p)

b) In the context of integration testing, what is a *driver*? (2p)

c) In the context of integration testing, what is a *stub*? (2p)

9. Exploratory testing (6p)

a) Explain the difference between exploratory testing and ad-hoc testing. (2p)

b) How can exploratory testing be justified as a test method? (2p)

c) In terms of exploratory testing, what is a *charter*? (2p)

10. Multiple condition coverage (10p)

Specify a minimal set of test cases for the following function that result in 100% multiple condition coverage.

```
int rules(bool a, bool b, bool c, bool d) {
  if (a || b) {
    if (~b || c) {
      return 1;
    }
    return 2;
  }
  else if (c && d) {
    return 3;
  return 4;
}
```

11. Testing in agile development (4p)

Describe two techniques for system-level testing in agile development, where fast releases and continuous delivery is critical. For full points, use concrete examples of how to implement your chosen techniques.

12. More easy points (4p)

Answer the following questions true or false:

a) Automated testing is effective for ensuring that old bugs are not reintroduced.

b) Automated tests are cheap to create.

c) Automated testing is appropriate in an agile environment.

d) By automated testing people usually mean testing where tests are run automatically, but inputs and outputs are created by hand.

(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.)