Försättsblad till skriftlig tentamen vid Linköpings universitet

(fylls i av ansvarig)

Datum för tentamen	2010-08-18
Sal	TER4
Tid	08.00-12.00
Kurskod	TDDD04
Provkod	TEN 1
Kursnamn/benämning	Programvarutestning/Software Testing
Institution	*IDA*
Antal uppgifter som ingår i tentamen	15
Antal sidor på tentamen (inkl. försättsbladet)	7
Jour/Kursansvarig	Kristian Sandahl
Telefon under skrivtid	0706-68 19 57
Besöker salen ca kl.	9
Kursadministratör (namn + tfnnr + mailadress)	Gunilla Mellheden, 013-28 22 97, E-post: gunilla.mellheden@liu.se
Tillåtna hjälpmedel	Inga, endast skrivmaterial. No aids are allowed, only writing materials.
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	Credits Grade 0-32 U 33-40 3 40-47 4 48-56 5 Exam is shown 2010-08-31 at 12.30-13.30 in room Donald Knuth.
Vilken typ av papper ska användas, rutigt eller linjerat	
Antal exemplar i påsen	

Exam TDDD04 Software testing 2010-08-18

- Write clearly!
- Please use only one side of each paper and don't address more than one question per page.
- Justify your answers!

(I) Basic definitions

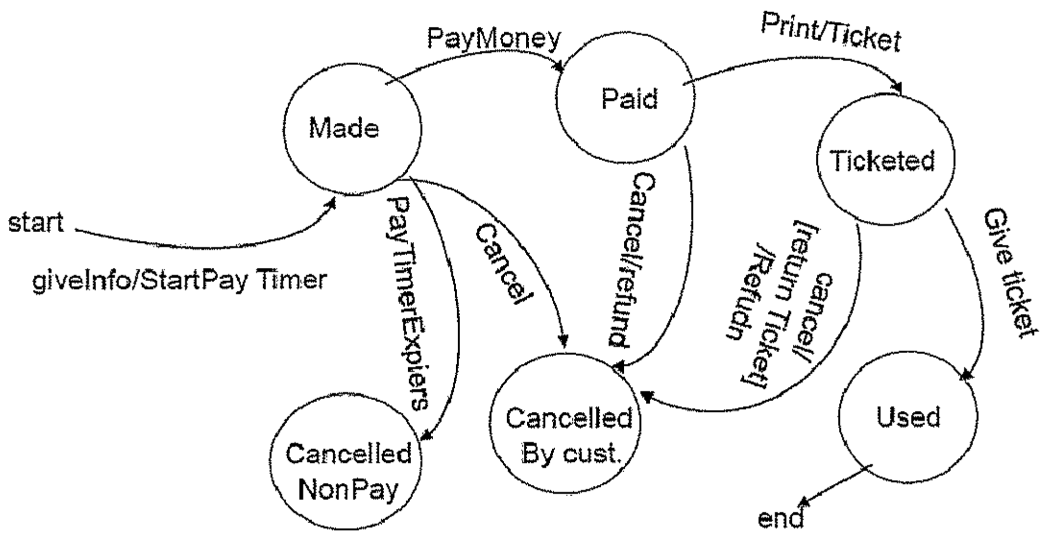
1. Describe/define the following terms in a software testing context: (3)
    a. Error
    b. Fault
    c. Failure

2. Draw a Venn-diagram of the expected and intended behavior of a program illustrating the relationship between: (3)
    a. Correct behavior.
    b. Wrong behavior due to omission
    c. Wrong behavior due to commission

(II) Unit and Integration testing

3. State-transition testing. (5)

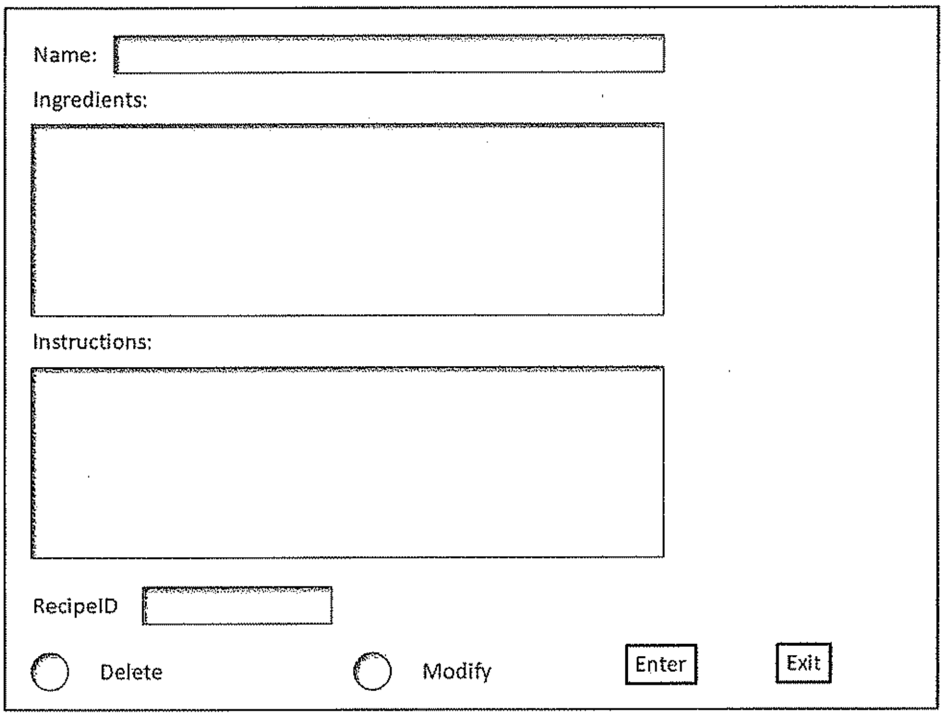
Consider the following example used in the book and lecture. It describes the ticket booking system of an airline.



a. Create a set of test-cases such that all paths are executed at least once during the test. Use a tabular representation of the test cases.
b. Create a set of test-cases such that all transitions are executed at least once during the test. Use a tabular representation of the test cases.
c. Give an example of a system where State-transition testing is not well applicable. Don't forget to motivate the answer.

4. Decision table testing (4)

Consider a small database interface for your collection of cooking recipes where you can add, delete and modify recipes. To add a new recipe: enter name, ingredients, instructions, and press Enter. The recipe is added to the database and returns a new RecipeID. To delete or modify a recipe: enter the RecipeID, select the Delete or Modify radio button and press enter.



a. Create a decision table reflecting this application. The conditions are:
   - Entered recipe data
   - Entered RecipeID
   - Selected Delete
   - Selected Modify

   The actions are:

   - Create new recipe
   - Delete recipe
   - Modify recipe

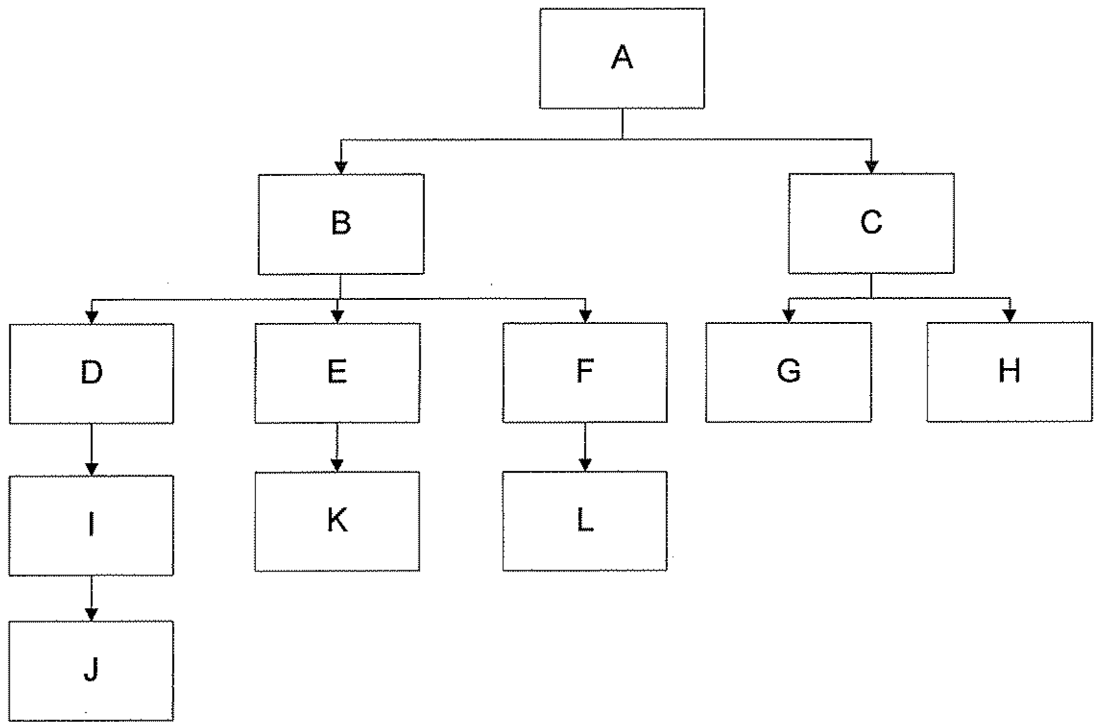
   Make sure that the table is deterministic and non-redundant

b. Show with an example what a non-deterministic table might look like. You can illustrate by modifying a part of the table in a.

c. Show with an example what a redundant table might look like. You can illustrate by modifying a part of the table in a.

5. Integration testing. (6)

The following figure illustrates the component hierarchy in a software system.



a. Describe the sequence of tests for integration of the components using a bottom-up approach and a top-down approach.

b. How many stubs are needed for top-down integration? Don't forget to explain how you calculated the result, since there are different conventions of how to calculate this.

c. How many drivers are needed for bottom-up integration? Motivate clearly.

6. Test-driven development (6)

a. What are the principle steps of Test-driven development (TDD)?

b. Give two potential benefits of TDD

c. Give two potential drawbacks of TDD

7. Describe one benefit of the Inversion of Control (IoC) software design principle when it comes to testing. (1)

8. Data-flow dependencies (4)
   a. In data flow analysis we have the different activities define; kill; and used (d,k,u). List all possible two letter combinations for d, k and u. For each combination describe whether that combination represents an acceptable on non-acceptable treatment of the variable.
   b. Define the coverage criteria All P-uses/Some C-uses.
   c. Describe one stronger, and one weaker strategy compared to All P-uses/Some C-uses

9. Control flow coverage (4)

   For the following section of code

   (Assume standard C++ behavior in that: if the first predicate in an OR statement is true, the second is not evaluated; if the first predicate is false in an AND statement, the second is not evaluated):

```
if ((a > 1) || (b > 1))
{
    //BLOCK A
}
else if ((c > 0) && (d > 0))
{
    //BLOCK B
}
else
{
    //BLOCK C
}
```

   a. Write down minimal sets of test cases to get 100% coverage with respect to (*ignoring testing strategies such as domain testing and boundary testing*):
      - Decision coverage
      - Condition coverage
      - Condition/decision coverage
   b. Draw a flowgraph of the program; add a start and an end node. Calculate the Cyclomatic number of the flow graph

(III) System testing

10. Cause-effect testing (4)

   Specification: While driving a car with cruise control on you can press none, one or two pedals. The pedals are gas, brake and clutch. If the brake or the clutch is pressed the cruise control is shut off. If you only press the gas pedal the speed will increase but the cruise control will be in idle mode, resuming the set speed as soon as the gas pedal is released.

Based on the specification on the previous page:

a. Identify causes and effects.
b. Design a cause-effect graph for the identified causes and effects.
c. Propose a decision table for testing the software.

11. Acceptance test(3)

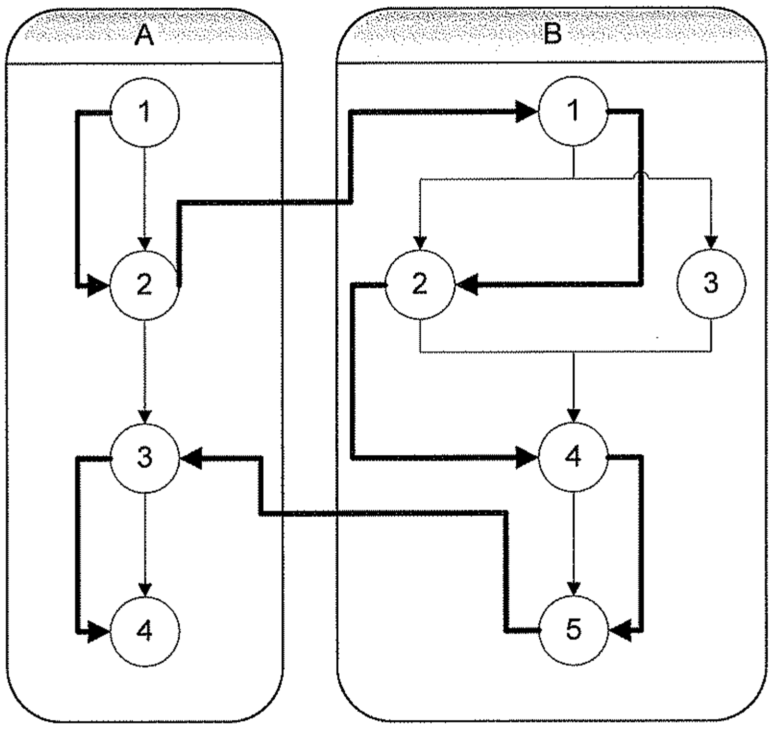
Describe the following concepts:

a. Benchmark test
b. Pilot test
c. Parallel test

12. Path-based integration (4)

Definition: An MM-path is an interleaved sequence of module execution paths (MEP) and messages.

In the following figure module A calls module B. A MM-path is indicated with thick arrows, the internal control-flow with thin arrows.



a. Identify Source nodes and Sink nodes in the modules A and B
b. Identify Module execution paths (MEP)

(IV) Automated testing

13. Model-based testing (3)
    a. Is model-based testing automating black-box or white-box test design? Don't forget to motivate your answer
    b. The models must be concise and precise. Describe what these two properties mean, and motivate why this is necessary for good application of model-based testing.

14. Write down a short description of the linear scripting techniques. Give one advantage and one potential drawback of linear scripting techniques. (3)

15. Test planning (3)

    The IEEE-Std 829-2008 recommends that you define Integrity levels.

    a. Define the concept integrity level
    b. Give examples of consequence-based integrity levels
    c. Component A has a higher integrity level than component B. Describe how this makes the planning of the testing of component A and B different. Don't forget to motivate your answer