



Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	2017-08-16
Sal	
Tid	8-12
Kurskod	TDDD04
Provkod	
Kursnamn/benämning	Programvarutestning
Institution	IDA
Antal uppgifter som ingår i tentamen	
Antal sidor på tentamen (inkl. försättsbladet)	6
Jour/Kursansvarig	Lena Buffoni
Telefon under skrivtid	013-28 40 46
Besöker salen ca kl.	09:00
Kursadministratör (namn + tfnnr + mailadress)	Anna Grabska Eklund
Tillåtna hjälpmedel	Ordbok

LiTH, Linköpings tekniska högskola
IDA, Institutionen för datavetenskap
Lena Buffoni

Written exam
TDDD04 Software Testing
2017-08-16

Permissible aids

Dictionary (printed, NOT electronic)

Teacher on duty

Lena Buffoni, tel: 013-28 40 46

Instructions and grading

You may answer in Swedish or English.

Your grade will depend on the total points you score on the exam. This is the grading scale:

Grade	3	4	5
Points required	50%	67%	83%

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.

Note: When I visit the exam, I will take a slow walk among all students, so you don't need to sit with your hand raised. Just call for my attention when I pass your desk.

1. Definitions (4p)

Describe the difference between a fault of commission and a fault of omission. Give an example of each.

A fault of commission occurs when we enter something into a representation that is incorrect. **Faults of omission** occur when we fail to enter correct information.
Commission – a floating point representation instead of an integer representation
Omission – a login feature missing

2. Coverage criteria (6p)

- a) Is it possible to have code for which branch coverage and statement coverage tests are the same? Justify your answer. 2p
Yes, eg: if a > 1 then 5 else 6
- b) Give examples of 2 artefacts other than code where coverage criteria can be applied. 2p
Requirement specification, Design specification
- c) Is 100% coverage a guarantee of test suite quality? Justify your answer. 2p
No, test cases can be chosen with ineffective values, for example really trivial

3. Defect classification (6p)

You are given the following description of a software defect:

“The application does not warn the user that they are running low on disk space allows them to edit the file. Once the user tries to save the file he is informed that this is not possible to do as there is not enough disk space left. The specification states that the application shall ensure sufficient space is available before attempting to save the file.”

Fill in appropriate values for the columns in the table below (copy to a separate paper first) when you classify the defect above.

Fault/Defect	Attribute	Value
Defect	Asset	application
Defect	Artefact	Design document
Defect	Effect	functionality
Defect	Mode	missing
Defect	Severity	high

4. True/false (6p)

- a. A complete analysis cannot give false negatives **F**
- b. Validation checks that a product meets the needs of the customer **T**
- c. Performance testing tests not functional requirements **T**
- d. Threads cannot be used for testing at system level **F**
- e. The goal of software testing is to aim for completeness **F**
- f. An error is the result of a fault in the specification **F**

5. Black box testing (16p)

You are asked to test PaySafe an application for online payment security. When making a payment online the user receives a 4 digit identifier and has then one minute to input their personal code (a 8-12 digit numerical value) and this identifier in the application to authorize the payment.

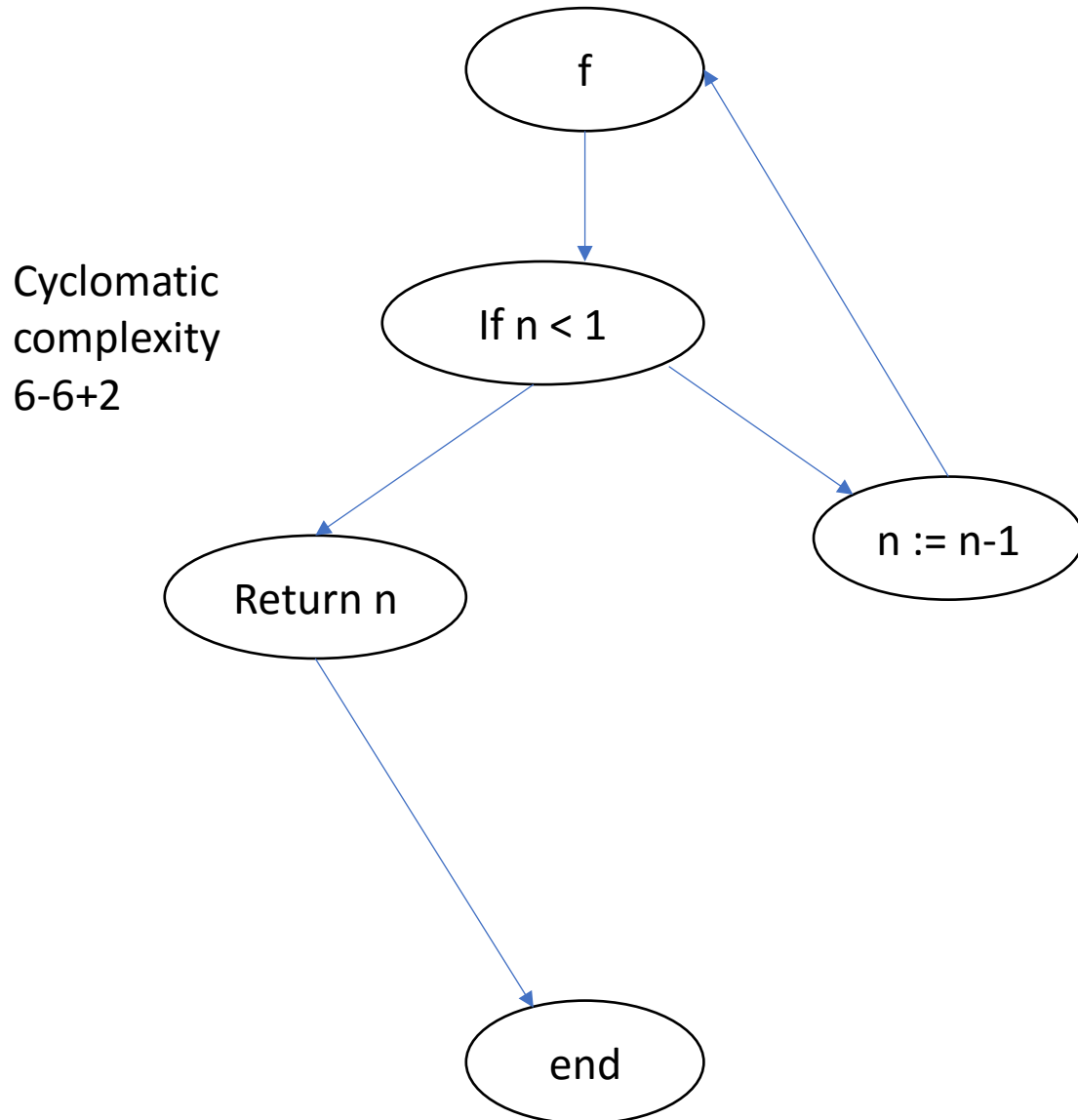
First the user inputs the 4 digit commerce identifier. If this identifier is wrong the application returns code UNKNOWN_TRANSACTION. If the identifier is correct, the application asks for the user code. If the code is correct, the application returns OK and the payment is processed. If the code is wrong the application asks to reenter the code again. After 3 tries the application is blocked and returns BLOCKED. If the user takes more than one minute to enter the code and identifier, the transaction is timed out, the application returns TIMED_OUT and the user has to restart.

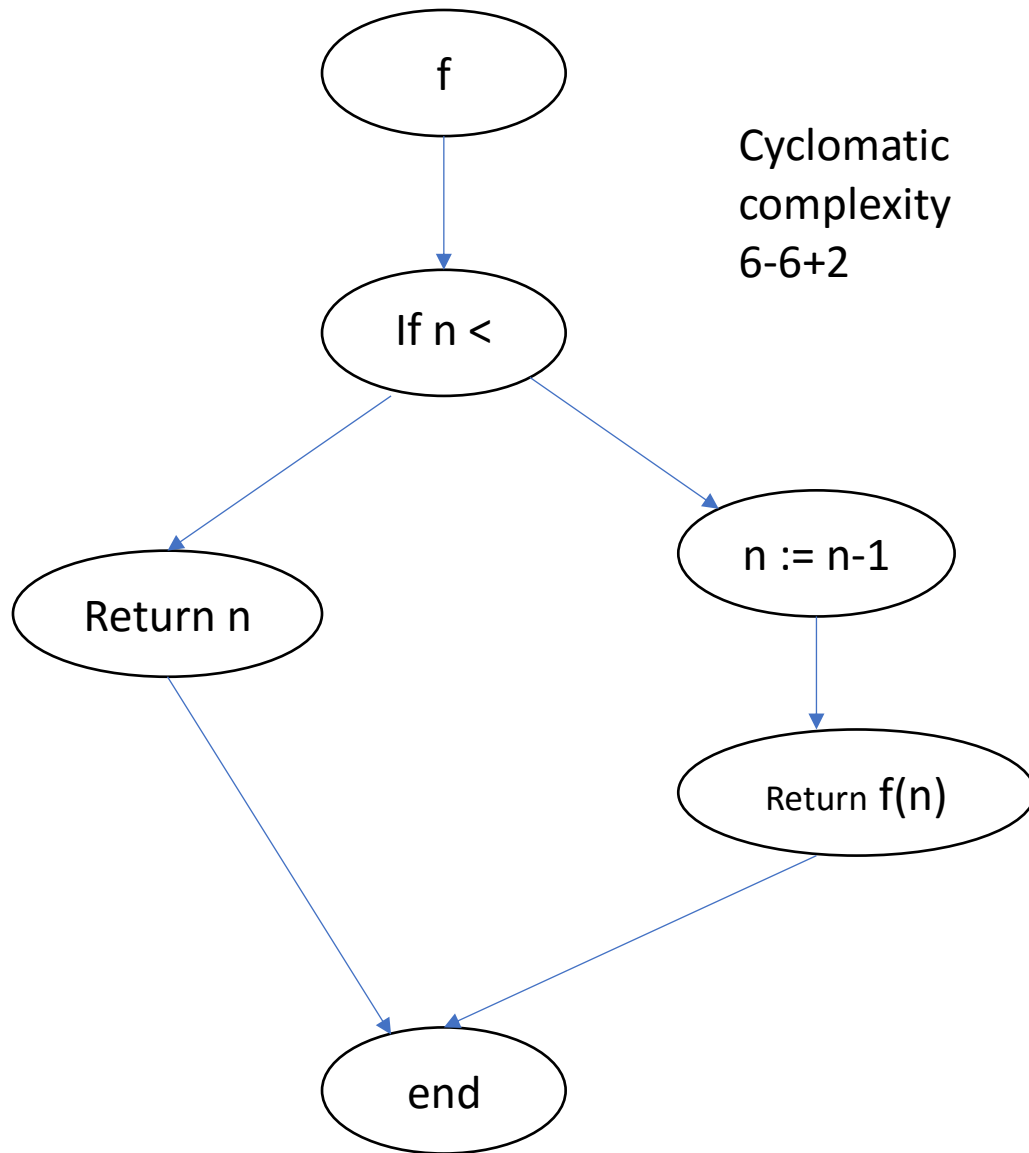
Follow a suitable test case design method in order to test the given method. Justify your choice of test case design method, and any assumptions you make about the system.

You can receive up to 8 points for an appropriate representation of the problem, and up to 8 points for an appropriate translation of your representation into test cases. An excessive amount of test cases (based on the choice of test case design method) will lead to a deduction of points.


```
public static long fibonacci(int n) {  
    if (n <= 1) return n;  
    else return fibonacci(n-1) + fibonacci(n-2);  
}
```

Two acceptable representations:





Two test cases:

Input	Expected Output
-1	-1
2	1

7. Integration testing (6p)

- Name the types of additional scaffolding code that may be needed for integration testing. **Stubs and drivers**
- Give an advantage of path based testing over bottom up or top down integration testing. **It is based on the execution flow which is a more natural decomposition than structural decomposition.**
- What is the role of regression testing? **To ensure new changes do not introduce and bugs.**

8. Exploratory testing (4p)

- Define charter and session for exploratory testing.
 - Schedule: an uninterrupted block of time devoted to testing (1-2 hours)**

ii. Charter: a guide defined before the testing session covering

- what to test
- available documentation
- test tactics
- risks involved

b. List two advantages and one disadvantage of exploratory testing.

- Does not detect omission errors

+ low cost

+ if a defect is detected the test session can be adapted to explore it further

9. System-level testing (4p)

a. Describe how transition based testing works and provide an example of a domain suited to such testing. (2p)

i. the system is modeled as transitions between states and test cases are generated to provide the required level of coverage (eg: all transitions), systems with events and that need to keep track of their state are well suited for this type of representation

b. Describe MM-path testing (1p)

Finding paths of inter-module communications, and devise tests for each communication path

c. What is alpha and beta acceptance testing? (1p)

Alpha test: at the developer's site, controlled environment

Beta test: at one or more customer site.

10. Model based testing (6p)

a. Give two examples of correctness properties that can be verified by model-based checking. 1p

i. absence of deadlocks, livelocks, starvation, violations of constraints/assertions

b. Name two limitations of model-based testing 2p

i. Requires different skills than manual test case design

ii. Mostly limited to functional testing

c. Describe three activities in the model-based testing process. 3p

i. Model the SUT and/or its environment

ii. Use an existing model or create one for testing

iii. Generate abstract tests from the model

iv. Concretize the abstract tests to make them executable

v. Execute the tests on the SUT and assign verdicts

vi. Analyze the test results.