

Report No	TENTA_TDDD04
Organization	Linköping University, Sweden
Last Modified	13 June 2011
Modified by	Peter Bunus (peter.bunus@liu.se)

Tentamen TDDD04 (Programvarutestning)

Examination TDDD04 (Software Testing)



Tentamen: TDDD04 – Programvarutestning (2011-05-30)

Examinator: Peter Bunus

Information

Poängavdrag kommer att göras om punkterna nedan inte åtföljs!

- 1) Använd endast framsidan (delfrågor kan vara på samma sida).
- 2) Sortera inlämnade svar med avseende på uppgiftsnummer i stigande ordning.
- 3) Svaren får vara på svenska eller engelska.
- 4) Dina svar skall tydligt visa lösningsmetod. Enbart rätt svar kommer inte att ge poäng. I det fall du är osäker på frågeställning, skriv ner din tolkning och lös uppgiften utifrån din tolkning.

Betygsgränser

[0..55)	poäng	Betyg U
[55..70)	poäng	Betyg 3
[70..85)	poäng	Betyg 4
[85..100]	poäng	Betyg 5

Lycka till!

Examination: TDDD04 – Software Testing (2011-05-30)

Examiner: Peter Bunus

Information

Please also observe the following; otherwise it might lead to subtraction of points:

- 1) Use only the front side of the sheets.
- 2) Sort the solution sheets according to the task number.
- 3) Answers may be in English or Swedish.
- 4) Your answers should clearly show solution methods, reasons, and arguments. Short answers should be briefly motivated. If you have to make an assumption about a question, write down the assumptions you make.

Grading

To pass the exam you have to do at least 55 points from 100 possible.

[0..55)	points	Grade Fx
[55..70)	points	Grade C
[70..85)	points	Grade B
[85..100]	points	Grade A

Good Luck!

Bonne chance!

Viel Glück!

Sëkmès!

祝你好運

祝福

1. Associate to each one of the concepts from the first column the correct corresponding definitions from the left column: (5p)

1. Debugging	a. Checking whether the software performs correctly
2. Regression Testing	b. Checking that a previously reported defect has been corrected.
3. Testing	c. Identifying the cause of a defect, repairing the code and checking the fix is correct.
4. Retesting	d. Checking that no unintended consequences have occurred as a result of a fix.

Answer: 1-c, 2-d, 3-a, 4-b

2. Which pair of definitions is correct? (5p)

- a. Regression testing is checking that the reported defect has been fixed; retesting is testing that there are no additional problems in previously tested software.
- b. Regression testing is checking there are no additional problems in previously tested software; retesting enables developers to isolate the problem.
- c. Regression testing involves running all tests that have been run before; retesting runs new tests.
- d. Regression testing is checking that there are no additional problems in previously tested software, retesting is demonstrating that the reported defect has been fixed.

The correct answer is d.

Regression testing is testing that nothing has regressed. Retesting (or confirmation testing) confirms the fix is correct by running the same test after the fix has been made. No other option has both of these as true.

3. When is testing complete? (5p)

- a. When time and budget are exhausted.
- b. When there is enough information for sponsors to make an informed decision about release.
- c. When there are no remaining high priority defects outstanding.
- d. When every data combination has been exercised successfully.

Sometimes time/money does signify the end of testing, but it is really complete when everything that was set out in advance has been achieved.

4. Which of the following is in the correct order (typically)? (5p)

- a. Unit testing, system testing, acceptance testing, maintenance testing.
- b. System testing, unit testing, acceptance testing, maintenance testing.
- c. Acceptance testing, system testing, maintenance testing, unit testing.
- d. Unit testing, maintenance testing, system testing, acceptance testing.

a) Unit testing, system testing, acceptance testing, maintenance testing.

5. Which of the following are examples of iterative development models? (5p)

- (i) V-model
- (ii) Scrum
- (iii) Waterfall model
- (iv) Agile development model

- a. (i) and (ii)
- b. (ii) and (iii)
- c. (ii) and (iv)
- d. (iii) and (iv)

The correct answer is c.

The other two models are sequential models.

6. Which of the following statements are true? (5p)

- (i) Defects are likely to be found earlier in the development process by using reviews rather than static analysis.
- (ii) Walkthroughs require code but static analysis does not require code.
- (iii) Informal reviews can be performed on code and specifications.
- (iv) Dynamic techniques are generally used before static techniques.
- (v) Dynamic techniques can only be used after code is ready to be executed.

- a. (i), (ii), (vi).
- b. (ii), (iii), (v).
- c. (i), (iv), (v).
- d. (i), (iii), (v).

The correct answer is d.

The other answers are incorrect because:

ii. Walkthroughs do not require code and static analysis does require code.

iv. Static techniques do not execute the code and therefore can be run before and after the code is ready for execution.

7. A system is designed to accept values of examination marks as follows: (5p)

Fail 0–39 inclusive
Pass 40–59 inclusive
Merit 60–79 inclusive
Distinction 80–100 inclusive

In which of the following sets of values are all values in different equivalence partitions?

- a. 25, 40, 60, 75
- b. 0, 45, 79, 87
- c. 35, 40, 59, 69
- d. 25, 39, 60, 81

The correct answer is b.

8. Consider the following pseudo code:

```
1 Begin
2   Read Time
3   If Time < 12 Then
4       Print(Time, "am")
5   Endif
6   If Time > 12 Then
7       Print(Time -12, "pm")
8   Endif
9   If Time = 12 Then
10      Print (Time, "noon")
```

```

11      Endif
12 End

```

How many test cases are needed to achieve 100 per cent decision coverage? (5p). Motivate your answer.

- a. 1
- b. 2
- c. 3
- d. 4

The correct answer is c.

The three decisions are in sequence and the conditions are all mutually exclusive (if any one is true the others must be false). Hence a test case that makes the first decision true will make the second and third decisions false and so on. So test case 1 (say Time = 6) would exercise the path True, False, False, test case 2 (say Time = 15) would exercise the path False, True, False. Test case 3 would have to be Time = 12. This combination achieves 100 per cent decision coverage because each decision has been exercised through its true and its false outcomes.

9. Which of the following is a structure-based (white-box) technique? (5p)

- a. Decision table testing
- b. State transition testing
- c. Statement testing
- d. Boundary value analysis

The correct answer is c.

All other options are specification-based (black-box) techniques, and the main distracter is answer (a) because decision table testing could be confused with decision testing.

10. A software component has the code shown below:

```

Program Biggest
  A, Biggest: Integer
  Begin
    Read A
    Biggest = 10
    While A > 0
      Do
        If A > Biggest
          Then Biggest = A
        Endif
        Read A
      Enddo
    End
  End

```

The component has exit criteria for component testing that include 100% statement coverage. Which of the following test cases will satisfy this criterion? (5p)

- (a) 0
- (b) 10, 0
- (c) 10, 5, 0
- (d) 10, 11, 0

The correct answer is (d).

11. Consider the following pseudo code: (10p)

```

1 Begin
2   Read Time
3   If Time < 12 Then
4     Print(Time, "am")

```

```

5      Endif
6      If Time > 12 Then
7          Print(Time -12, "pm")
8      Endif
9      If Time = 12 Then
10         Print (Time, "noon")
11     Endif
12 End

```

If the test cases Time = 11 and Time = 15 were input, what level of decision coverage would be achieved?

- a. 100% or 6/6
- b. 50% or 3/6
- c. 67% or 4/6
- d. 83% or 5/6

Please also give a short explanation of your answer.

Answer:

Test case 1 exercises the decision outcomes True, False, False

Test case 2 exercises the decision outcomes False, True, False

This leaves the True outcome of decision 3 not exercised. Of the 6 possible decision outcomes, 5 have been exercised, so the decision coverage is 5/6 (about 83%).

12. Let us consider the following code:

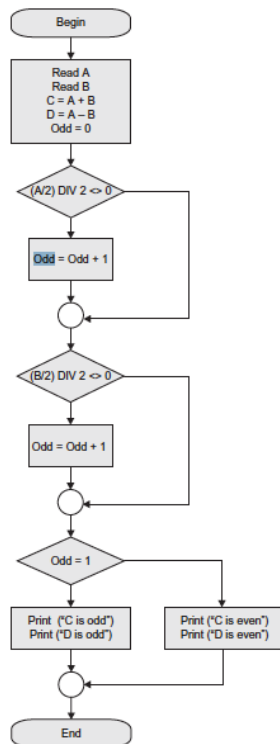
```

1      Program OddandEven
2
3      A, B: Real;
4      Odd: Integer;
5
6      Begin
7          Read A
8          Read B
9          C = A + B
10         D = A - B
11         Odd = 0
12
13         If A/2 DIV 2 <> 0 (DIV gives the remainder after division)
14         Then Odd = Odd + 1
15         Endif
16
17         If B/2 DIV 2 <> 0
18         Then Odd = Odd + 1
19         Endif
20
21         If Odd = 1
22         Then
23             Print ("C is odd")
24             Print ("D is odd")
25         Else
26             Print ("C is even")
27             Print ("D is even")
28         Endif
29 End

```

- a) Draw a flow chart and a control flow graph (5p) to represent the following code: (5p)
- b) Calculate the cyclomatic complexity of the control graph (5p)
- c) Write down input values for test cases that satisfy McCabe's base path coverage (5p)

a)



b) $V(G) = E - N + 2 * P = 12 - 10 + 2 * 1 = 4$

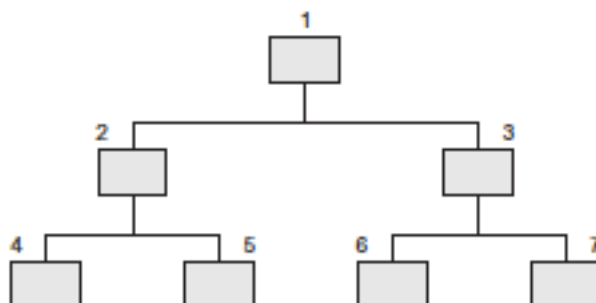
c) We would need to test the following decision points that will satisfy McCabe's base path coverage

(A/2) DIV 2 <> 0	(B/2) DIV 2 <> 0	Odd == 1
(A/2) DIV 2 == 0	(B/2) DIV 2 <> 0	Odd == 1
(A/2) DIV 2 <> 0	(B/2) DIV 2 == 0	Odd == 1
(A/2) DIV 2 <> 0	(B/2) DIV 2 <> 0	Odd != 1

At a closer inspection, only the last three test cases will be valid (if (A/2) DIV 2 <> 0 and (B/2) DIV 2 <> 0 Odd can never be == 1)

(A/2) DIV 2 == 0	(B/2) DIV 2 <> 0	T1: A = 4, B = 3
(A/2) DIV 2 <> 0	(B/2) DIV 2 == 0	T2: A = 4, B = 6
(A/2) DIV 2 <> 0	(B/2) DIV 2 <> 0	T3: A = 4, B = 5

13. 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(5p) and a top-down approach. (5p)

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.(5p)

c. How many drivers are needed for bottom-up integration? Motivate clearly.(5p)

a)

Bottom Up

S1: 4, driver(2)
S2: 5, driver(2)
S3: 4, 5, driver(2)
S4: 6, driver(3)
S5: 7, driver(3)
S6: 6, 7, driver(3)
S7: 4, 5, 2, driver(1)
S8: 6, 7, 3, driver(1)
S9: 4, 5, 2, 6, 7, 3, 1

Top Down:

S1: 1, stub(2), stub(3)
S2: 1, 2, stub(3)
S3: 1, stub(2), 3
S4: 1, 2, stub(4), stub(5), 3, stub(6), stub(7)
S5: 1, 2, 4, stub(5), 3, stub(6), stub(7)
S6: 1, 2, stub(4), 5, 3, stub(6), stub(7)
S7: 1, 2, stub(4), stub(5), 3, 6, stub(7)
S8: 1, 2, stub(4), stub(5), 3, stub(6), 7
S9: 1, 2, 4, 5, 3, 6, 7

b) No of stubs = nodes - 1 = 7-1=6

No of sessions = (Node_leaves) + edges = 3 + 6 = 9;

b) No of drivers = nodes_leaves = 3

No of sessions = (Node_leaves) + edges = 3 + 6 = 9;