| | |
|---|---|
| Report No | TENTA_TDDD04 |
| Organization | Linköping University, Sweden |
| Last Modified | 6 June 2012 |
| Modified by | Peter Bunus (peter.bunus@liu.se) |

# Tentamen TDDD04 (Programvarutestning) Examination TDDD04 (Software Testing)

Linköping University, Department of Computer and Information Science
SE 581-83 Linköping Sweden

# Tentamen: TDDD04 – Programvarutestning (2012-05-28)

# Examinator: Peter Bunus

## 1 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.

## 2 Betygsgränser

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

Lycka till!

Linköping University, Department of Computer and Information Science
SE 581-83 Linköping Sweden

# Examination: TDDD04 – Software Testing (2011-05-28)

# Examiner: Peter Bunus

## 3 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 a make an assumption about a question, write down the assumptions you make.

## 4 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!

Sėkmės!

祝你好運

祝福

Linköping University, Department of Computer and Information Science
SE 581-83 Linköping Sweden

1. Which of the following statements is NOT correct?

a) A minimal test set that achieves 100% path coverage will also achieve 100% statement coverage.
b) A minimal test set that achieves 100% path coverage will generally detect more faults than one that achieves 100% statement coverage.
c) A minimal test set that achieves 100% statement coverage will generally detect more faults than one that achieves 100% branch coverage.

Please motivate your answer by giving a short explanatory example. (10p)

---

2. Which of the following requirements is testable?

a) The system shall be user friendly.
b) The safety-critical parts of the system shall contain 0 faults.
c) The response time shall be less than one second for the specified design load.
d) The system shall be built to be portable.

Please motivate your answer. (5p)

---

3. In prioritizing what to test, the most important objective is to:

a) find as many faults as possible.
b) test high risk areas.
c) obtain good test coverage.
d) test whatever is easiest to test.

Please shortly motivate your answer. (5p)

---

4. In a system designed to work out the tax to be paid: An employee has £4000 of salary tax free. The next £1500 is taxed at 10%. The next £28000 is taxed at 22%. Any further amount is taxed at 40%. Which of these groups of numbers would fall into the same equivalence class?
a) £4800; £14000; £28000
b) £5200; £5500; £28000
c) £28001; £32000; £35000
d) £5800; £28000; £32000

Please shortly motivate your answer. (5p)

---

5. For the example from the previous question, If, I would like to apply a boundary value testing approach. Which input values shall I choose for the tests?
Please shortly motivate your answer. (5p)

---

6. Which of the following is true about White and Black Box Testing Technique? (5p)
a) Equivalence partitioning, Decision Table and Control flow are White box Testing Techniques.
b) Equivalence partitioning, Boundary Value Analysis, Data Flow are Black Box Testing Techniques.
c) Equivalence partitioning, State Transition, Use Case Testing are black box Testing Techniques.
d) Equivalence partitionng, State Transition , Use Case Testing and Decision Table are White Box Testing Techniques.

---

7. Which are the minimum tests required for statement coverage and branch coverage for the following small example:

```
Read P
Read Q
If p+q > 100
thenPrint "Large"End if
If p > 50 then
Print "pLarge"
End if
```

a) Statement coverage is 2, Branch Coverage is 2
b) Statement coverage is 3 and branch coverage is 2
c) Statement coverage is 1 and branch coverage is 2
d) Statement Coverage is 4 and Branch coverage is 2

Please complete your answer by providing the test cases for statement and branch coverage. (5p)

---

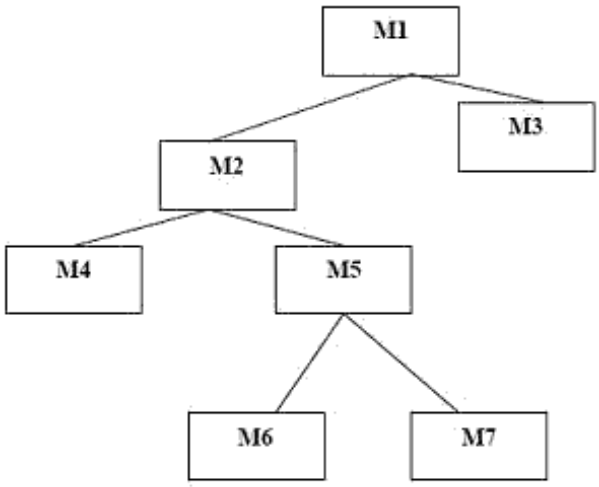
8. The following program is used in a hypothetical retail situation. The owner of a shop has decided that her staff can have a 10 percent discount on all their purchases. If they spend more than €15, then the total discount is increased by 50 cents. The price of each item being purchased is input into the program. When -1 is entered, the total price is displayed, as well as the calculated discount and the final price to pay. For example, the values €5.50, €2.00 and €2.50 are input, equaling €10.00. The total discount would equal €1.00 (10% of €10.00), with the total price to pay equaling €9.00. A second example would have purchases of €10.50 and €5.00, equaling €15.50. In this case, as the total value is over €15, the discount would be €2.05 (10% of €15.50 is €1.55, plus 50cents as the original total is over €15), meaning that the total price to pay would be €13.45.

The source code, written in pseudo code, for a program which has been written to perform the task described above, is shown below:

```
1     program Example()
2     var staffDiscount, totalPrice, finalPrice, discount, price
3     staffDiscount = 0.1
4     totalPrice = 0
5     input(price)
6     while(price != -1) do
7           totalPrice = totalPrice + price
8           input(price)
9     od
10    print("Total price: " + totalPrice)
11    if(totalPrice > 15.00) then
12          discount = (staffDiscount * totalPrice) + 0.50
13    else
14          discount = staffDiscount * totalPrice
15    fi
16    print("Discount: " + discount)
17    finalPrice = totalPrice – discount
18    print("Final price: " + finalPrice)
19 endprogram
```

a) Write down the minimal set of tests that would help you to achieve statement coverage (5p)
b) Write down the minimal set of tests that would help you to achieve boundary value testing (5p)
c) Draw a flow chart and a control flow graph to represent the following code: (5p+5p)
d) Calculate the cyclomatic complexity of the control graph (5p)
e) Write down input values for test cases that satisfy McCabe's base path coverage (5p)
f) Draw the data flow graph of the program and annotate the data flow graph with "definition" and "use" information of each module variable. (5p)
g) List all the definition-use paths (du-path) for the variable "price" and write a test for each of these test paths. (5p)

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




(a)

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)