Exam for ETS200 Software Testing
– *WITH SOLUTIONS*
Lund University, Department of Computer Science
Time: 2013-03-14, 14:00-19:00
Place: Sparta: C–D
Assessment: total 60 points, at least 30 points is required in order to pass the exam.
Answers may be written in Swedish or English.
Start answering each new task on a new page.

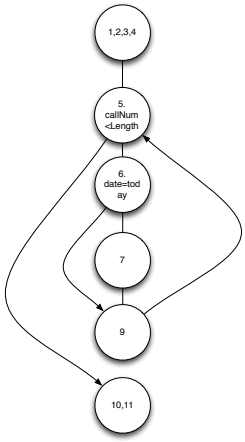
---

1. Define the following terms (one sentence each):
   1p per correct definition a) Testing
   The process of dynamically executing a piece of software with the purpose of evaluation/finding faults/demonstrating their absence/validation/verification b) Inspection
   The process of manual scrutiny of some software artifact with the purpose of finding faults/demonstrating their absence/validation/verification c) Verification
   The process of checking that the software meets requirements d) Validation
   The process of checking that the software meets customer expectations e) Failure
   A deviation in the execution from expected behavior f) Oracle
   The reference (document or piece of software) that allows testers to judge pass from fail
   (6p)

2. a) Define the two main purposes of software testing, which sometimes may conflict. (1p)
      Assess quality and find defects, 0.5p each
   b) Give an example on how the two test purposes may conflict in practice. (1p)
      Few defects found may be sign of bad testing or good quality.
   c) Give an example of a test metric that may be misleading due to the dual purpose of testing, and describe how this can be avoided. (2p)
      Defects found is misleading without any reference on how much test effort is spent, what is normally found etc.

3. Describe two different methods for test case design; one that uses a *black box* approach and one that uses *white box* approach. For each of the methods, present:
   a) a general description of the method, (2p)
      Example equivalence partitioning vs Coverage based testing
   b) information needed to derive test cases using the method, (2p)
      Specification vs code
   c) which test levels the method is most feasible for, and why, (2p)
      Integration and system test vs unit test; access to code and size of scope
   d) main advantages of the method, and (2p)
      generally applicable vs automated
   e) main drawbacks of the method.(2p)
      not automated vs impossible to apply to large scale

4. For the procedure *SpeakingTimeToday* below,
   (a) draw the control flow graph and calculate the McCabe Cyclomatic Complexity, (3p)



$CC = 7 - 6 + 2 = 3$ *or* $2 + 1 = 3$

(b) define the test cases needed to achieve 100% decision coverage, (2p)
One test case with a list of two calls, one today and one which is not today.

(c) set up def-use tables for all three variables, and define minimum test cases needed to achieve 100% def-use coverage. (5p)

1p per variable
callNum 3-5, 3-6, 3-7, 3-9, 9-5, 9-7
speakingTime 4-7, 7-7
listOfCalls 1-5, 1-6, 1-7
One test case with a list of two calls, one today and one which is not today. (2p) Note that test cases should include both input values and expected output.

```
1  procedure SpeakingTimeToday (in listOfCalls; out speakingTime);
2  begin
3    callNum = 0
4    speakingTime = 0
5    while (callNum < listOfCalls.Length)
6       if listOfCalls[callNum].date = today
7            speakingTime = speakingTime + listOfCalls[CallNum].time
8       end
9       callNum = callNum  + 1
10   end
11 end
```

5. A medium-sized company wants to introduce a tool for automatic test execution.
   a) How much test code should they expect to write, compared to the production code? (1p)
   about 50-50
   b) Which trade-offs should they do when deciding which test cases to automate? (1p)
   investment in implementing test vs number of times executed.
   c) Define the three test automation approaches of *recorded scripts*, *engineered scripts*, and *model-based testing.* (3p)
   d) Discuss pro's and con's for the three approaches. You should at least cover:
      - Upfront investment costs
        low-med-high
      - Test script maintenance costs
        high-med-low
      - The *oracle* problem
        high-high-med (3p)

(Cont'd on next page)

6. Derive test cases for the *NearestNeighbor* function, using *equivalence partitioning.*
`NearestNeighbor(in: listOfPos:matrixType, numNeighbors: int; out: pos:int);`

The *listOfPos* variable is a $3*(n+1)$ matrix of real numbers, representing 3D coordinates, as illustrated below. The input variable *numNeighbors*, represents the number of valid neighbors in the matrix, stored in positions 1 to *numNeighbors*. The reference coordinate is stored in position 0 of the matrix.

	0	1	2	...	NumNeighbors	...	n
x	0.4	0.8	-0.4	...	2.4	...	
y	0.5	0.8	0.9	...	0.6	...	
z	0	-0.8	14.1	...	-2.9	...	

The function calculates distances between the reference coordinate and coordinate m as $d_{0,m} = \sqrt{(x_0 - x_m)^2 + (y_0 - y_m)^2 + (z_0 - z_m)^2}$, and returns the index *pos* for the nearest neighbor. Define:

a) equivalence classes for input and output variables (both valid and invalid), including assumptions made regarding the specification of the program, (4p)
EC1 0 < NumNeighbors <= n valid
EC2 NumNeighbors > n invalid
EC3 NumNeighbors <= 0 invalid
EC4 pos = 0 valid
EC5 0 <pos <=NumNeighbors valid
EC6 NumNeighbors < pos<=n invalid
EC7 pos < 0 invalid
EC8 pos > NumNeighbors invalid
0.5 p for each

b) one test case per equivalence class (input data, procedures and expected output). (6p)
1 p for each complete for EC1-EC5, 1p for commenting that EC6-8 cannot be generated through the interface, only by manipulating variables.

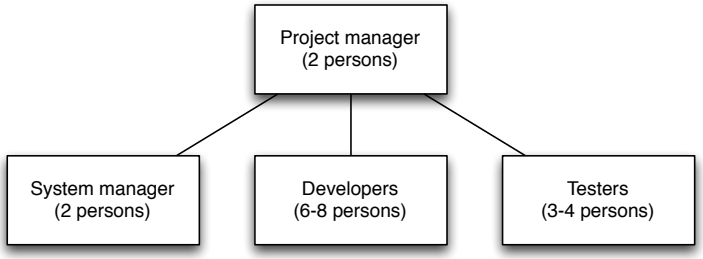
7. Two reviewers have inspected a document and found the defects listed in the table below, where 0 represents not found defect, and 1 represents found defect. The Lincoln-Peterson model estimates the number of defects as $\hat{N} = n_1 * n_2/n$, where
$\hat{N}$ = estimated total number of defects
n_1 = number of defects found by reviewer 1
n_2 = number of defects found by reviewer 2
n = number of defects found by both reviewers

Defect	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
Reviewer 1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1
Reviewer 2	1	0	1	0	1	0	1	0	0	0	0	1	1	0	0

a) Estimate how many undetected defects there are in the document. (1p)
$n_1 = 12$ $n_2 = 6$ $n = 3$ $N = 12 * 6/3 = 24$. *9 remaining (1 p for 9; .5 p for 24)*

b) Would you recommend release or re-review of the document, based on this estimate? Why? (3p)
1) Don't release; 2) too many left 3) unless the software is not critical.

8. The organization chart below comes from the PUSS course project.



a) Describe it in terms of the organizational models in Kit's book. (2p)
Testing performed by dedicated resource/organization. Staff organized based on their competency and task.

b) Select and describe an alternative organization based on another of Kit's models. (2p)
For example, test resources connected to development team.

c) Discuss pro's and con's for the two alternatives. You should at least cover:
   - competence provisioning,
     Easier with dedicated unit for each competence type
   - communication,
     Harder for dedicated unit, since developers and testers have larger organizational distance.
   - management, and
     Easier for dedicated unit, since managers have only one type of resources and tasks to manage.
   - scale-up to large-size organizations.
     No significant difference. Dedicated unit creates communication problems, connected teams may create management problems.

(4p)