UNIVERSITY
OF SKÖVDE

School of Informatics

WRITTEN EXAMINATION

Course Software Testing G1F, 7.5hp

Sub-course

Course code IT373G

Credits for written examination 5hp

Date 2023-10-23

Examination time 14:15-19:30

Examination responsible

Teachers concerned

Aid at the exam/appendices

Other

Instructions

- ☐ Take a new sheet of paper for each teacher.
- ☒ Take a new sheet of paper when starting a new question.
- ☒ Write only on one side of the paper.
- ☒ Write your name and personal ID No. on all pages you hand in.
- ☒ Use page numbering.
- ☒ Don´t use a red pen.
- ☒ Mark answered questions with a cross on the cover sheet.

Grade points

Examination results should be made public within 18 working days

Good luck!

Total number of pages

Grading

This exam contains five sections. Each section examines the student with respect to one or two of the learning goals specified in the course plan. In order to pass the exam, the student therefore, needs to pass all five sections. For your convenience, each section list the relevant examination criteria.

==========

Section 1

Examination criteria: (i) The student can describe the process for model-driven test design (mdtd) as well as its different activities and (ii) The student can describe the advantage of agile approaches such as test-driven development (tdd) and briefly describe the tdd approach.

Question 1

MDTD allows one test engineer to do the math and leaves the rest of the activities (finding test values, automation, execution etc.) to traditional testers and programmers. For each of the below artifacts developed in an MDTD process, **describe** and **exemplify** what the test engineer does to create it.

1. Model/structure
2. Test requirements
3. Refined test requirements / test specification

Question 2

What is a continuous integration server and why is it necessary when using an agile approach such as test-driven development?

The level of detail and number of arguments determines the grade.

==========

Section 2

Examination criterion: The student can explain the limitations of software testing

Question 3

Why is it, that software testing never can prove correctness and why should we strive for high-quality tests if we still cannot prove that the software is free from faults? ***Note**: This is a question with many answers so you should elaborate on this and give at least three arguments and / or examples.*

Number of arguments and examples determine the grade

==========

Section 3

Examination criterion: The student can explain given test techniques in sufficient detail

Question 4

Restrictive inactive clause coverage (RICC):

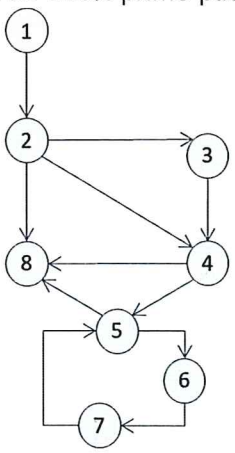
1. Describe the technique (i.e., how it works)
2. List the RICC test requirements for predicate P (i.e., required rows) based on the table below. **Give all feasible alternatives for each clause.**

Row	*A*	*B*	*C*	*Predicate P*	P_A	P_B	P_C
1	T	T	T	T	T	F	T
2	T	T	F	F	T	F	T
3	T	F	T	T	T	F	T
4	T	F	F	F	F	F	T
5	F	T	T	F	T	F	T
6	F	T	F	T	T	T	T
7	F	F	T	F	T	F	F
8	F	F	F	F	F	T	F

Question 5 (not for Nibar)

Prime-path coverage

1. Describe the technique (i.e., how it works)
2. List all the prime-paths you need for 100% prime-path coverage of the below graph



Question 5 only for Nibar (replacing question 5 above)

Given the IDM below and a base-choice combination strategy

1. What base-choice values would you select and why?
2. Which test requirements would this give you? **Note**: some might be infeasible in which case you should mark them as infeasible.

Parameter	*Class 1*	*Class 1*	*Class 3*
First instance of y	x[0]	!= x[0] && != x[length-1]	x[length-1]
Length of x	length=0	length>0	
Number of y instances	0	1	More than 1

Question 6

Mutation, the AOR mutation operator

1. Describe the AOR operator (i.e., how it works)
2. List the AOR mutants for the given code fragment

```
int count = 0;
for(i=0; i < x.length; i++){
   if (x[i]%2 == 1 || x[i] > 0) {
      count++;
   }
}
return count;
```

Question 7

For a higher grade also: Compare and contrast GACC and RACC coverage. Your comparison should focus on (*i*) the effectiveness with respect to the probability to expose failures, (*ii*) the cost with respect to number of test requirements, and (*iii*) the usability with respect to infeasible test requirements.

==========

Section 4

Examination criteria: (i) The student can explain the common concepts in software testing and test automation and (ii) The student can describe the main functionality given by a test automation framework such as JUnit.

Question 8

Four requirements (RIPR) have to be satisfied by a test in order for a test engineer to observe a failure when running it. Which are these requirements and what do they mean?

Question 9

During the lectures we have talked about the two concepts *observability* and *controllability*. Give an example of software that typically have low observability and explain why the observability is low in that type of software.

Question 10

One of the features in JUnit is assertions. A typical assertion is assertTrue(string, boolean). Give an example and describe what this assertion does. Make sure that the description is detailed enough to explain the parameters.

Level of detail determines the grade.

==========

Section 5

Examination criterion: The student can give at least one very good argument for why the use of coverage criteria help testers get high-quality tests.

Question 11

In this course we have discussed several advantages of using coverage criteria. Two of the advantages concerns the quality of the test set and a well-defined stopping criterium. How does the use of coverage criteria help the tester improve the test set and how does a well-defined stopping criterium help empowering the test manager?

Number of arguments and level of detail determines the grade.