Software Testing 1DL610
2020-01-07 08:00-13:00.

Question	Points	Score
1	8	
2	4	
3	10	
4	4	
5	6	
6	8	
Total:	40	

Exam Rubric

All answers to be written in English or Swedish. A mark of 50% is required for a 3, the marks 4 and 5 are evenly distributed in the interval from 50% to 100%.

The exam is a home exam administered via studium, and available for students who have already registered for the exam. You are free to write you exam by hand or do the exam electronically. If you write your exam by hand, then you need to photograph each page. I prefer to have the files as a single PDF. Please number each handwritten page and on each handwritten place please make it clear what question and what question part you are answering.

Please make it clear which question you are answering on each page. If you chose to do the exam using a word processor or something similar then I will only accept a PDF file.

The exam is a take home exam, but it should be treated as any other exam. It is a closed book exam, and you are not permitted to use books/lectures slides or other material from the internet during the exam.

I will be online during most of the exam, and I will be able to answer questions. I will only answer questions on clarifying exam questions. You can contact me via email `justin.pearson@it.uu.se`. I can guarantee that I will be online 8:00-9:00 to answer any initial problems. I will be online during most of the exam, but I cannot guarantee an instance response.

Questions

1. General Questions on Testing.
   (a) (2 points) Describe the difference between *black-box* and *white-box* testing.
   (b) (2 points) Describe the difference between a *Software Fault* and a *Software Failure*. For each give an example.
   (c) (2 points) Describe the difference between *validation* and *verification*.
   (d) (2 points) Why would you have to do more testing even though your test suite gives 100% statement (or line) coverage?

2. (4 points) Dijkstra famously said: "You can test for the presence of bugs, not their absence". With reference to Turing's halting problem explain what Dijkstra meant. For the practical software engineer testing is still useful. Explain why you should still do testing even if you believe Dijkstra's pronouncement.

3. Consider the following fragment of Java/C like code:

```
public int foobar(int x, int y) {
  int z=0;
  if(x == 0) {
   while(x>y ) {
     z=z+x;
     x = x - 2;
   }
  } else { z = 42; }
  if(x<0) {
    while(x < y) {
      x = x + 2;
   }
  }
  return(z);
}
```

   (a) (2 points) Draw a Structural Graph that represents the above piece of source code.
   (b) (2 points) For you graph, enumerate a complete set of paths that guarantee node coverage.
   (c) (2 points) Construct a finite set of test cases that guarantee node coverage. For each test case indicate which execution path is covered. Remember the difference between test cases and test paths.
   (d) (2 points) For the above fragment of code is there any difference in requiring complete edge coverage rather than node coverage? Explain your answer.
   (e) (2 points) For your graph enumerate all prime paths. You must show how you derive the prime paths of your graph.

4. Logic Coverage. Given that the variables `a,b` are integers consider the following expression:

```
(( a == b) && (a > 5) ) && (  (a != b) || ( a > 10) )
```

(a) (2 points) Define *clause coverage* and for the above expression derive a set of test cases that give clause coverage.

(b) (2 points) Define *active clause coverage* and for the above example derive a set of test cases that give active clause coverage.

5. Input Space Partitioning.

(a) (2 points) For an input domain define what a partition of a domain is. Given an example of an input domain and a partition.

(b) (2 points) Example the significance of the disjointedness of condition of a partition to generating test cases.

(c) (2 points) Consider a function that takes a list `l` and an object `e`. The function should return `true` if the object `e` appears twice in the list `l` and false otherwise. Give two different partitions of the input domain that you would use to test this function. One partition should be done using the interface-based approach and one partition should be done for the functionality based approach. For each of your partition describe what you are trying to test.

6. General Practical Questions on Testing.

(a) (4 points) What is a test plan? Why is one necessary or useful? What sort of things go into a test plan? (I don't expect you to give a complete description of test plans, but I do expect you to include some of the important items that should go into a test plan.)

(b) (2 points) What is mocking and why is it necessary during testing?

(c) (2 points) How does integration differ from Unit Testing? Give an example to illustrate your answer.