Exam in Testing

Justin Pearson

Dec. 2011

Cover Sheet

Problem no.	Solution provided	Max	Your points
1		10	
2		7	
3		12	
4		8	
5		10	
6		12	
	Total:	59	

Anonymous Exam Code . : .....

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%.

I will not be able to come to the exam to answer questions, if you have trouble understanding a question then please state clearly any assumptions that you have to make to answer the question.

Hjälpmedel:

Pen,pencil, ruler,rubber, dictionary, calculator.

1. General Questions on Testing.

   (a) Describe the difference between a *software fault* and a *software failure*. (**points 2**) Further, give an actual example illustrating the difference. (**points 2**)

   (b) Define *software validation* and *software verification*. (**points 2**)

   (c) Describe why it is sometimes necessary for the software tester to be a different person from the programmer. (**points 1**)

   (d) Even during the requirements gathering phase of a project there is a role for the software tester. Describe some of the things that a software tester can contribute to requirements gathering and requirements definition phase. (**points 3**)

2. Test cases, paths and coverage.

   (a) Define what a test case is. (**points 1**)

   (b) Describe, and give examples to illustrate, the difference between a test requirement and a test case. (**points 2**)

   (c) Given a structural control flow graph of a piece of code, define a what test path is. (**points 2**)

   (d) Is it true that for every test path there is a test case? Justify your answer! (**points 2**)

3. Consider the following fragment of Java (or C) like code:

```
public int foobar(int x, int y,int z) {
   int k = 0;
   for(int i=0; i < z; i++) {
    for(int j=0; j<y; j++) {
      if(i<j) {
       k=k*y+x; } else {
        k = k -1;
      }
    }
   }
   return(k);
}
```

(a) Draw a structural control flow graph that represents the above piece of source code. (**points 2**).

(b) For you graph, enumerate a complete set of paths that guarantee node coverage. (**points 2**).

(c) Construct a finite set of test cases that guarantee node coverage. For each test case indicate which execution path is covered. (**points 2**)

(d) For the above fragment of code is there any difference in requiring complete edge coverage rather than node coverage? Justify your answer. (**points 2**)

(e) Enumerate all the prime paths of the graph you have constructed. (**points 2**)

(f) For each prime path you have found construct a test case for that path. Indicate which prime path is covered by your test cases. (**points 4**)

4. Logic Coverage. Given that the variables `a,b` are integers, consider the following expression (note `||` means logical or and `&&` means logical and):

```
(( a == b) || (a < 5) ) || (  (a != b) && ( a + b  == 10) )
```

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

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

5. Input Space Partitioning.

   (a) For an input domain define what a partition of a domain is. (**points 2**)

   (b) When deriving test cases, why should the blocks in a partition be disjoint? (**points 2**)

   (c) The test designer has many possible partitions to pick. What criteria does the test designer use to design partitions? (**points 2**)

   (d) One way to use input space partitions is to test for extreme and boundary value of loops, conditions, and input variables. Give an example of such a partition and explain what it would test for. (**points 4**)

6. General and Practical Questions on Testing.

   (a) Integration testing is very different from unit testing. During integration testing it is often necessary to write *stubs* and *drivers*. Describe what *stubs* and *drivers* are. (**points 2**)

   (b) When writing a *stub* for some as yet unimplemented functionality, the tester has to produce return values. Describe the choices that the developer has. (**points 3**)

   (c) What is a test plan? (**points 1**) There are many different types of test plans. Describe in more detail one type (your choice) of test plan. Explain what should go into the test plan and what the purpose of such a document would be. (**points 3**)

   (d) The mantra of test driven development is *Red*, *Green* and *refactor*. Explain this mantra. (**points 3**)