1. Consider the diagram shown in Figure 10.8 on p. 234 (4th edition).

* 1. (A) How many logical paths are there? List them all.

c1 -> s1 c1 -> c2 c1 -> s1 -> c2 c1 -> s1 -> c2 -> s3

* 1. (B) How many paths are required to cover all the statements?

c1 -> s1 -> c2 -> s3 (1)

* 1. (C) How many paths are required to cover all the branches?

1

1. In code inspection, what would you set as the condition (e.g., how many discovered defects) for re-inspection? (pg 220)

If the following conditions are met, I’d re-inspection

* Number of discovered defects still high
* New defects have been introduced
* Code performance significantly reduced
* Defect not resolved

1. List the four techniques discussed to perform verification and validation. (pg 200)

* Testing: process ode designing test cases, executing the program in a controlled environment, and verifying that the output is correct
* Inspections and reviews: which can be applied to a program or to intermediate software artifacts.
* Formal methods: mathematical techniques used to “prove” that a program is correct
* Static analysis: process of analyzing the static structure of a program or intermediate software product.

1. List two techniques you can use to perform **validation**- that is, to ensure your program meets user requirements.

* Error guessing: test cases are designed in an attempt to figure out the most plausible faults, usually based on the history of faults discovered on similar projects.
* Error-prone: identifies through reviews and inspections those areas of requirements and design that seem to continuously contain defects.

1. Briefly explain the concept of static analysis, and to which software products it can be applied. (pg 200 + 223)

* Static analysis: process of analyzing the static structure of a program or intermediate software product.
* static analysis can be applied to the following situations
  + Intermediate documents
  + Source code
  + Executable file
  + FindBugs – open source bytecode checker

1. Briefly explain two different ways to decide when to stop testing. (pg 218 – 220)

* Keeping track of the test results and observing the statistics. If the number of problem discovered stabilizes to near zero, we would usually consider stopping the testing
* Pepper the existing code with defects and observe how many of the seeded defects are discovered. If discovering 70% of the real defects was the target, then the testers could proclaim victory and stop testing.

1. Consider the simple case of testing two variables, X, and Y, where X must be a non-negative number, and Y must be a number between -5 and +15. Using boundary value analysis, list the test cases.

X >= 0  
Y >= -5 && Y<=15  
  
X = [1, 0, 1, 155, maxint -1, maxint, maxint +1]  
Y = [ -6, -6, -4, 0, 5, 14, 15, 16]  
  
foreach(X)  
 foreach(Y)  
 test(X, Y);

1. Describe the steps involved in a formal inspection process and the role of a moderator in this process. (pg 220 + 221)

* Planning; inspection team and a moderator are designated and materials are distributed to team members several days prior to the actual inspection meeting
* Overview: overview of the work product and related areas is presented, similar to a walk-through
* Preparation: every inspector is expected to study the work product and related materials thoroughly in preparation for the actual meeting
* Examination: an actual meeting is arranged where the inspectors review the product together
* Rework: after the meeting, the author corrects all defects, if any
* Follow-up: the corrections are checked by the moderator, or the corrected work is inspected again, depending on the result of the inspection
* Moderator: controls flow of session and verify results

1. What is the difference between performance testing and stress testing. (pg 204)

* Performance testing: Involves verification that the program behaves according to its performance specifications, such as some number of transactions per second or some number of simultaneous users.
* Stress testing: Ensures that the program behaves correctly and degrades gracefully under stress conditions, such as high load or low availability of resources. Stress testing will extend the testing of software beyond the performance specification to see where the breaking points are.

1. Describe your level of comfort for quality if a group of software developers tell you that they used test-driven development. Explain why.

Depends on the size and purpose of their product, I’d react differently. If the size is small and the application does not contains lot of logics/ algorithms, I don’t see any significant gains in doing test-driven development. If it were the opposite scenario I’d ask them for the reason they chose to do, if the reason(s) is valid, I’d be interested in it.