1. A number of researchers have addressed the issue of subsumption among coverage entities in a program. These researchers have addressed two issues. The first is:

Given a set of entities $E$ that need to be covered, identify the minimal subset of $E$ that, if covered, guarantees the coverage of all entities in $E$. For example, suppose that $E$ is the set of all branches in the program. Such an algorithm will identify which branches in $E$ must be covered to guarantee that all branches in $E$ are covered.

The second is:

Given a set of entities $E$ that need to be covered, identify an implication among them such that if $A$ and $B$ are in $E$ and $A$ implies $B$, the coverage of $A$ implies that $B$ is covered.

For each of these issues, identify a body of work (a paper or papers). For each body of work, give a description that discusses the following items:

a. A complete reference to the paper or papers for that work
b. An overview of the technique presented
c. An overview of empirical evaluation of the technique
d. A discussion of the benefits of the technique
e. A discussion of the limitations of the technique
f. A discussion of the open problems in the area
2. In *Code Complete*, McConnell describes a number of combinations of data states that could be computed for dataflow testing and advises that some of the patterns should be viewed "suspiciously." In particular, he includes *defined-defined*: the variable is defined and then defined again before it is used. To check these patterns, and possibly remove them from the code, a dataflow analysis can be performed. For the "defined-defined" pattern, give a data-flow framework to compute defined-defined pairs of statements and:
   a. Indicate if this problem is an *any-* or *all-*paths problem
   b. Indicate if the problem is a forward or backward data-flow problem
   c. Give the definitions of the GEN, KILL, IN, and OUT sets for the problem
   d. Give the initial values of the sets
   e. List and discuss the algorithm
   f. Discuss the complexity of the algorithm
   g. Provide a proof that your algorithm terminates
   h. Give an example to illustrate your algorithm
3. You have been asked to provide information to a test manager about improving the testing process in his/her development projects. In particular, the manager is interested in automated techniques that have been presented by researchers, their functionality, and their cost/benefits tradeoffs. Two areas of interest to the manager are automated test-case generation techniques and techniques for regression testing. Select one of these areas and do the following:
   a. Provide details of two existing techniques to accomplish the task
   b. Discuss current implementations of these techniques that are available
   c. Discuss cost/benefit tradeoffs of the techniques that would need to be considered in integrating them into an industrial environment
   d. Give and support your recommendation about the use of the techniques in practice.
4. You live in a 5-story building serviced by one elevator. Provide a set of natural language (i.e., plain English) requirements for how the elevator should operate. Based on these requirements, provide a complete Statechart model of the elevator system. Discuss any assumptions you have made in your model. Modify your single-elevator Statechart model to work for
   a. A building with 10 stories and 1 elevator
   b. A building with 5 stories and 2 elevators
   c. A building with 10 stories and 2 elevators
Discuss the challenges in performing the above modifications. Assess your initial design for extensibility with respect to these three different evolution requirements.
5. Imagine that you are on a future ICSE program committee that is charged with selecting the most significant paper in the area of software testing and/or program analysis for the period 2000-2007.
   a. Select and give an appropriate citation for the paper you would nominate
   b. What is the problem addressed by the paper?
   c. Provide a one-paragraph summary of the thesis and argument of the paper
   d. Describe the impact that you think the paper has had
   e. Identify at least two future research directions that build on the results presented in the paper. For each of the directions you identified:
      • Discuss why you think the direction is promising and worth pursuing
      • Discuss the main research challenges for the research
      • Propose a research plan for investigating the research direction and discuss how your research approach addresses the research challenges you identified
      • Propose a measure of success for your research plan. That is, how will you validate your research and assess whether it was successful and to what extent?
6. Software engineering research has been criticized as having had no effect on practicing software developers.
   a. Give three examples of techniques or tools that are the result of software engineering research that have affected actual software development practice
   b. Justify that they have in fact made a difference to the practice of software development
   c. Discuss why you think these three have managed to have an impact
   d. Give three examples of successful software engineering research techniques or tools that have not had an appreciable impact on practice. Please pick examples for which the reason is other than "there has not yet been enough time ..."
   e. Justify that these have been successful research techniques or tools
   f. Discuss why you think these three have not had an impact on practice
   g. Express your own opinion on the importance of "relevancy to software development practice" as reason for a researcher to select a research area
7. Software development process.
   a. Describe the main characteristics of agile processes and, in particular, eXtreme Programming (XP)
   b. Describe and differentiate two structured (more traditional) processes, one of which should be the Waterfall model
   c. Present two software development scenarios, one that describes a situation in which an agile method is appropriate and one that favors a more structured approach
   d. Predict the future evolution of software processes with respect to the amount of agility/structure they include and justify your prediction
8. Consider the following problem. Many users have portable laptop computers which may be occasionally disconnected from their organization’s file server. When reconnected, they wish to have their laptop files synchronized with those on the file server and vice versa. It is your job to design software to address this need.

a. Give a precise design specification, in terms of a UML static model diagram, of such a system, including method signatures.

b. The components in your solution are interdependent. For example, system clocks and directory naming conventions must be coordinated. One way of conveying these interdependencies is by specifying invariants that the software must maintain. Using a formal notation of your choice, provide invariants for the system that precisely specifies its expected behavior.

c. Programming languages don’t normally provide invariants as a first class feature. It is therefore up to the designer and programmer to reestablish an invariant when a reconnection is made. Describe three general design approaches (that will work for more than just this example) for maintaining invariants. Using the synchronization example, provide UML collaboration diagrams illustrating how the schemes will work.

d. What are the advantages and disadvantages of each approach? In what situations should each be used?
9. Provide concise, but complete answers to the following questions.
   a. OO software is claimed to be more maintainable than structured software. Provide at least two arguments to support this claim.
   b. In his classical collection of essays “The Mythical Man-Month,” Fred Brooks presents the often quoted statement “Adding people to a late software project makes it later.” Suggest at least one reason why this statement is true.
   c. If adding people to the team is not a solution, suggest two more appropriate solutions that a project manager can adopt when a project is falling behind schedule.
   d. Why is it typically the case that integration testing reveals errors even when adequate unit testing has been performed?
   e. One of the human factors to be considered when designing a user interface is that people make mistakes. Discuss two ways in which you can take into account this factor in the design of a UI.
   f. Does having an adequate set of unit tests help refactoring? Why?
   g. Most current object-oriented languages use the same class hierarchy for the purposes of sub-typing and code reuse. Discuss and give examples of how this overlap can lead to confusion and subtle bugs.