How to Answer Essay Questions in the Graduate Level Computer Sciences Exams

Cem Kaner

Florida Tech Computer Sciences

February 2003

1. The Problem

Last fall, I reviewed the grading of the Software Engineering comprehensive. The most striking problem that I saw in the failing and borderline-passing exams was the weak organization of the answers. For example:

- Several answers missed sections of a multiple-issue question, losing critical points simply because nothing was said (even though the student probably knew the answer).
- Several answers missed the point of the question, and spent a lot of space on irrelevant material.
- Several answers were so disorganized that it took significant work to find the relevant material in the mass of irrelevancy.

Ultimately, some students who might have passed on the basis of their knowledge of the material, failed because of their weak essay-answering skills.

Several of the exams in our Department ask essay-style questions.

We expect graduate students to be able to parse an essay-style question and write an organized answer.

That expectation is not going to change.

These notes suggest some ways to improve your approach to essay-style questions.

SUMMARY SUGGESTIONS FOR THE TYPICAL ESSAY EXAM IN COMPUTER SCIENCE

CEM KANER

FEBRUARY 2002

1. STUDYING

- **Put together a list of source materials for the exam.**

  This includes the relevant chapters of the course textbook and all required readings. Also note any references that were listed as "optional" but recommended. These often provide additional insights that will help you understand the material (including an exam question on the material).

  If you are lucky, the instructor has prepared this for you already in a "study guide."

- **Get copies of all of the source materials (including recommended readings)**
- **Create a list of all the key words used in the class. Prepare definitions for each one.**

  An exam might not have any questions that specifically ask for definitions, but you will often find it useful to define a term as part of your answer to a long essay question.

- **Create a set of sample questions that covers the material you are studying.**

  If you can, get questions from copies of previous exams, especially those written by the instructor who is teaching your course.

  If your course text is published with a "study guide", take sample questions from the guide.

  If your course text includes practice exercises, include these in your set of sample questions.

  For the rest, make a list of every significant topic in the course you are studying, and make up questions for every topic not already well covered with the sample questions you have on hand.

- **Now answer your sample questions.**

  No one would prepare to run a race primarily by talking about races with friends, or watching a race on television, or reading about races in books.

  - **To prepare to run a race, you have to practice running.**
  - **To prepare to write an exam, you have to practice outlining and answering questions of the kind you will face in the exam.**

  Your first attempt for each question should be open with no time limit. Check your lecture notes AND any other readings.

- **AFTER you have tried your own answers, compare notes with friends.**

  The best way to prepare for these tests is to attempt each question on your own. AFTER you have tried your own answers, compare notes with your friends. I recommend that you study with one or more partners. 3-5 people is a good sized group. 8 is too many.

2. STARTING THE EXAM

- **Skim the questions**
  - **Get a fast overview of the structure and coverage of the exam. Note whether any questions are related. Sometimes, thinking through the answer to one question can help you plan some of the details of your answer to another.**
- **Develop a budget for your time**

- The exam is time-limited, but some exams are intended to be easy to finish in the time available; others will be high-pressure. What's the situation with this exam?
- If this is a time-pressured exam, quickly build a time budget for it. For example, if the exam is 100 points long, and you have 60 minutes for the exam, then you have to earn 1.67 points per minute. A 5-point definition is only worth 3 minutes. A 25-point essay is worth 15 minutes. If you spend more than your budget on one question, you will be short on another one. *The ideal time budget should allow at least 10 minutes at the end of the exam for review.*

▪ Choose the order of your answers

In almost every exam, you can answer the questions in any order that you choose.

- If you are nervous, start by answering the question you find easiest. Then the second-easiest, and so on. By the time you have reached the harder questions, you will probably have relaxed enough to remember the relevant material.
- If you are not so nervous, answer questions in any order that seems sensible. For example, some students tend to spend too long on short answer questions. They should answer long essays first and cram in their short answers at the end, when time pressure is more obvious.

3. ANSWERING THE QUESTIONS

▪ Look for the call of the question

Every well written exam question asks specifically for some information. Your task is to provide the information that the question is calling for, and to not provide information that the question does not call for.

- Don't answer what has not been asked. For example, if I ask you to define one thing, don't define that and then give me the definition of something related to it. If you do, (a) I won't give you extra credit, (b) I'll think that you don't know the difference between the two things, and (c) if you make a mistake, I'll take off points.
- Give the number of items requested. For example, if I ask for two scenario tests, don't give one or three. If you give one, you miss points. If you give three, I will either grade the first two and ignore the third (this is my normal approach) or grade the first two that I happen to read (whatever their order on the page) and ignore the third. I will never read the full list and grade what I think are the best two out of three.

Beware of simply memorizing some points off a slide. If I think you are giving me a memorized list without understanding what you are writing, I will ruthlessly mark you down for memorization errors. In general, if you are repeating a set of bullet points, write enough detail for them that I can tell that you understand them.

o

▪ Be aware of the meaning of the question's words and answer appropriately. For example, these are typical meanings of some frequently used exam words:

- Compare: When you compare two things, point out (and perhaps explain or describe) similarities between them.
- Contrast: When you contrast two things, point out (and perhaps explain or describe) differences between them.
- List: When you provide a list of items, write briefly, using only one or a few words per item. No explanation of the items is needed.
- Identify: Same as list.
- Describe: A description of something attempts to give the reader a picture of it. A description of a process is often chronological, explaining the steps of the process in

order. A description of an abstract concept includes details (perhaps examples) that make it easier for the reader to understand or imagine.

        - **If you are asked to describe the relationship among things, you might find it easiest to work from a chart or a picture. You are probably not required to use a diagram or chart, but many professors will welcome it if it helps you get across your answer.**
        - **If the question asks you to describe or define something that is primarily visual (such as a table or a graph), your answer will probably be easier to write and understand if you draw an example of what you are defining or describing.**

    - **<u>Define:</u> Provide the meaning of a word or phrase. *Check with your professor about this.* Some professors expect formal definitions. Others just want you to describe the concept behind the word. Some professors welcome examples if they help you clarify the definition. Others don't. While a few professors might penalize you for this style, when I find it hard to define or describe something, I often write my answer around an example.**
    - **<u>Analyze</u>: Normally means, to break a concept or process or set down into component parts and discuss the relationships among the components. Sometimes you are asked to analyze in a specific way, such as "analyze the impact of X on Y." In this example, you might explain the effects of X (and the components of X) on Y.**
    - **<u>Explain</u>: Help the reader understand something. Explanations often focus on "why" or "how".**
    - **<u>Evaluate</u>: Discuss something in a way that allows you to reach a judgment (a conclusion) about it. In your answer, explain the rationale behind your conclusion. In a mathematically or formally oriented question, "evaluate" might mean to work through the formal material in order to reach the result. (Think of evaluating an expression.)**
    - **<u>Argue:</u> Pick a position in a debate or controversy, state it, and provide a series of arguments that support your position.**
    - **<u>Discuss</u>: This is a vague word. It is often intended as an open question, inviting you to choose the best way to address the question. For example, "Discuss the effect of X on Y" might be best addressed as "Describe and explain the effect of X on Y". If the relationship between X and Y is controversial, then "discuss" invites you to present multiple viewpoints and the data that support them.**

    If I ask you to analyze something according to the method described in a particular paper or by a particular person, I expect you to do it their way. If I ask you to <u>describe</u> their way, do so. If I ask you to <u>apply</u> their way, you don't have to describe it in detail, but I expect you to do the things they would do in the order they would do them, and to use their vocabulary to describe what you are doing.

- **<u>Outline your answer:</u>**
    - **Unless your answer will be trivially simple, develop a structure for the information that you will provide.**
    - **Often, the best outline explicitly connects to the question, using the key words in the question.**
    - **The outline should explicitly cover every issue raised in the question.**
    - **If the question contains multiple parts, make a separate section for each part.**
- **<u>Structure your answer according to the outline AND SHOW THE STRUCTURE:</u>**
    - **For example, use the structure of the outline as headings.**
    - **Make your organization of the answer easily visible to the grader.**
    - **If the question contains multiple parts, make a separate section for each part.**

- **Provide appropriate amounts of information or detail.**
  - **If a question asks about "some", that means at least two. I normally expect three items in response to a "some". Similarly if the question asks for a list, I am expecting a list of at least three.**
  - **If I ask you for the result of a calculation, such as the number of paths through a loop, show your calculations or explain them. Let me understand how you arrived at the answer.**

4. BE AWARE OF GRADER BIAS

- **Some factors, in general, bias markers. These are generalizations, based on research results. I try, of course, to be unbiased, but it's a good idea to keep these in mind with ANY grader for ANY exam:**
  - ***Exams that are hard to read tend to get lower grades*. Suggestions: Write in high contrast ink (such as black, medium). Write in fairly large letters. Skip every second line. Don't write on the back of the page. If your writing is illegible, print. *If I can't read something you wrote, I will ignore it.***
  - ***Start a new question on a new page*. More generally, *leave lots of space on the page.* This gives you room to supplement or correct your answer later (when you reread the exam before handing it in) and it gives me room to write comments on the answer, and it makes the answer easier to read.**
  - ***Answers that are well-organized are more credible*. Suggestions: If the question has multiple parts, start each part on a new line, and identify each part at its start. In a list, start each list item on a new line—maybe bullet the list. For example, consider the question: "What is the difference between black box and white box testing? Describe the advantages and disadvantages of each." You could organize this with five headings:**

Additional points to consider.

4. EXAM ERGONOMICS

- **Use a good pen.**
  - **Lawyers and others who do lots of handwriting buy expensive fountain pens for a reason. The pen glides across the page, requiring minimal pressure to leave ink. If you use a fountain pen, I suggest a medium point (write large) to avoid clogging. Also try gel pens or rollerballs. Get one that you can write with easily, to avoid writer's cramp. Basic ballpoints are very hard on you. Look at how tightly you hold it and feel how hard you press on the page.**
-

Study Guide Suggestions -- Page 2

- **Difference between black and white**
- **Advantages of black box**
- **Disadvantages of black box**
- **Advantages of white box**
- **Disadvantages of white box**

-

2. Some Examples

The following examples are from the Fall 2002 software engineering comprehensive. I was not involved in the writing of these and so have no pride of authorship in them.

Every well written exam question asks specifically for some information. For example, consider this from the Fall 2002 Software Engineering comprehensive:

> *Compare and contrast two lifecycle models. What advantages or disadvantages does each model have over the other?*

Some students circle or underline keys words in a question like this, and use arrows to highlight relationships. They check off circles and arrows as they answer that part of the questions. A marked-up version of this question might look like:

> ***[Compare and contrast]*** **<u>two</u>** ***[lifecycle models.] What <u>advantages</u> or <u>disadvantages</u> does [each model] have [over the other]?***

This works well for some people. I sometimes find it too concise to work with, and so I generally list the elements of a multi-part question:

- **Compare two lifecycle models**
- **Contrast two lifecycle models**
- **What advantages does each model have *<u>over the other</u>*?**
- **What disadvantages does each model have *<u>over the other</u>?***

This defines the call of the question. It specifies what information is requested (called for).

- **Notice that you have NOT been asked to define a lifecycle model (or a model in general)**
- **Notice that you have NOT been asked for empirical research results**
- **Notice that you have NOT been asked for a longer list of models -- just two, please.**
- **Notice that when you describe an advantage of one model, it should be an advantage *OVER THE OTHER* For example, suppose you contrast spiral and evolution. Both of these have a common advantage over the waterfall model--they both expect requirements to change over time and have sensible ways to deal with change, whereas the waterfall does not. HOWEVER, even though the ability to cope with changing requirements is an advantage for any incremental process over waterfall, it is not clear that spiral has this as an advantage over evolution (or that evolution has this as an advantage over spiral). You have to explain what makes spiral better than evolution and what makes evolution better than spiral.**

To me, the most difficult part of this question is the distinction between advantages and disadvantages.

- ***You COULD rationalize away this distinction, saying that an advantage for one approach is a disadvantage for the other, so all you really have to do is list advantages of each.***

  - ***You COULD do this, but you SHOULD NOT.***
  - ***NEVER ASSUME THAT A DISTINCTION MADE IN THE EXAM QUESTION IS MEANINGLESS OR UNIMPORTANT. (Many distinctions are unimportant -- they are just there because the exam writer is inexperienced or didn't have enough time to carefully edit the questions. But most distinctions (for example between*** **advantages OR disadvantages*****) are intentional.***
- **Instead, I ask**

Define the characteristics of a "good test" and a "bad test." Explain the differences, why good is good and why bad is bad. Provide and justify a definition of a good tester.

In analyzing a question like this, it is essential to look for the *call of the question* -- the specific issues you are being asked to address:

- *DEFINE* the characteristics of a good test
- *DEFINE* the characteristics of a bad test
- *EXPLAIN THE DIFFERENCES:* why good is good and why bad is bad
- *PROVIDE and JUSTIFY a definition of a good tester.*

Many answers failed because they simply omitted some parts of this question. For example, several answers failed to contract good versus bad tests. Others failed to *justify* the definition of a good tester.

Here's another example:

What is the most expensive phase in software engineering? Justify your answer

Parse the question:

- What is *THE MOST EXPENSIVE PHASE* in software engineering?
- *JUSTIFY* your answer.

It should be obvious that you are being asked to talk about ONE phase.

It should also be obvious that most of the points for the answer to this question will be in the justification of your choice.

Several failing answers wrote about several phases instead of the one the student considered consider the most expensive. In these cases, we assigned grades based on the first phase mentioned by the student and ignored anything said about any of the other phases.

Some of the answers got so muddled up in outlining costs across the lifecycle that they never drew the conclusion (which is the *most* expensive phase), and so nothing in the answer "justified" the answer that was never provided. These answers could justifiably have been given a grade of zero.

One more example:

Describe a process for performing a software review. Clearly identify each step, and describe the purpose of each step.

To answer this:

- *Describe* a
- *Process* for performing
- *A software review*
- For each step in your process:
    - *Clearly identify* that step
    - *Describe the purpose of* that step.

My answer to a question like this would:

- Start with an overview of a software review
- Then list, step by step, from start to end, the main steps involved in setting up, running, and using the results of the review
- Then take each of these steps in turn. Make a heading for each. Discuss the purpose of the step under the appropriate heading.

3. Exams

Most of the comprehensives are based on a general study guide, and the first time you see the exam questions is in the exam.

A few of the exams are based on a study guide that lists the pool of questions from which your exam will be constructed -- your studying is focused on these questions and should include writing practice answers.

The exams that are based on a fixed pool are graded more harshly because we know that you had time to think through the question, look up references, and write practice answers as part of your studying.

I write more extensively about the fixed-pool approach in a paper that describes my approach to testing in the Software Testing course. You can find this paper at www.testingeducation.org/conference/wtst_kaner.doc.

4. Common Problems

Many students aren't used to answering essay questions and so they deal with them ineffectively. This problem is not unique to computer science students. For example, teaching first year law students how to answer essay questions is a critical task, repeated in course after course.[1] Many universities publish guidelines for undergraduates on how to study for, and organize, essay questions.[2]

These notes describe the problems that I often see and how I handle them as a grader. The examples are all taken from my Software Testing course.

4.1 Weak Structure

Consider the following question as an example:

> Define a scenario test and describe the characteristics of a good scenario test. Imagine developing a set of scenario tests for the Outlining feature of the word processing module of *OpenOffice*. What research would you do in order to develop a series of scenario tests for Outlining? Describe two scenario tests that you would use and explain why each is a good test. Explain how these tests would relate to your research.

This has several components:

- Define a scenario test
- Describe the characteristics of a good scenario test
- What research would you do in order to develop a series of scenario tests for Outlining?
- Describe two scenario tests you would use.

- Explain why each of the two scenario tests is a good test
- Explain how these two scenario tests would relate to your research

A well organized answer will have at least six sections, one for each of the bulleted components. You might have two additional sections, by splitting *Describe two scenario tests you would use* and *Explain why each of the two scenario tests is a good test* into two sections, one for each test.

Without structure, it is easy to miss a section and thereby to lose points.

Students must learn to focus their answer to match the "call of the question" (the specific issues raised in the question).

4.2 Shotgun Answers

A student using a shotgun strategy responds with a core dump of everything that seems to be relevant to the general topic. Much of this information might be correct, but if it is non-responsive to the call of the question, it is irrelevant and I will ignore it. However, to the extent that irrelevant information is incorrect, if I notice an error, I will deduct points for it.

Here's an example of a question that yielded a lot of shotgunning and not enough points:

> Imagine that you are an external test lab, and Sun comes to you with *OpenOffice*. They want you to test the product. When you ask them what test documentation they want, they say that they want something appropriate but they are relying on your expertise. To decide what test documentation to give them, what questions would you ask (up to 7 questions) and for each answer, how would the answer to that question guide you?

In the Testing course, we looked over a long list of requirements-eliciting questions.[3] Students were free to use the ones we discussed or to supply their own.

The question does *not* call for definition / discussion of IEEE 829 or for a list of the common test documentation components. It doesn't call for a description of test matrices or a discussion of how to create them. I got these and much more on a recent exam. Unless this information was couched in terms of a question or the interpretation of the question/answer, it was irrelevant--a waste of the students' time (and in a time-limited exam, a tax on the student's ability to complete the rest of the test in the time available).

4.3 Time Management

Many students have time management problems in essay exams. The comprehensives are not intended to be time-pressured exams. Nor are my exams. But many students run out of time.

Budget your time.

If you have a 100 point exam and 75 minutes, you should be earning 1.333 points per minute. Don't spend 20 minutes answering a 10-point question. Spend between 5 and 10 minutes, preferably 7.5 or fewer.

4.4 Lack of Preparation

If you are writing a comprehensive exam, do not (not, not, not) expect that the questions will be taken from a course that you took or that they will be marked the same way as your instructor marked them. Many different people teach the same course. Their grading approach, exam style, and prioritizing of material are all different. The study guide for a comprehensive exam is the description / list of material that will be on the exam. Anything in the study guide might be on the exam. Topics not covered in the study guide will probably not be on the exam.

Read the study guide.

If the study guide lists a pool of questions, study them in detail and develop answers for each one.

If the study guide lists a set of topics and some reference materials, *Read the Reference Materials.*

For each topic, figure out which references provide information on a given topic and write out a summary of that information.

4.5 Weak Group Preparation

If the study guide lists a pool of questions, the best way to prepare for these tests is for each student to attempt each question on her own. The first attempt should be open book with no time limit. After each student has her own answers, she should compare notes with other students. The diversity of approaches will highlight ambiguities in the question, hidden assumptions on the part of the student, and muddled, disorganized thinking about the structure and call of the question. Independent preparation by several students is essential.

Unfortunately, many students form study groups in which they either:

- Divide up the questions. One or two students attempt to answer each question and then report back to the group. The rest of the students then attempt to memorize the answers.
- Or, attempt to develop the answers in-group, four or more students arguing and together.

Neither of these approaches works well. There are so many questions in the study list that few (or no) students can effectively memorize all the answers. As a result, I often see answer fragments, relevant material mixed with irrelevant (something memorized for a different question), or answers that have been distorted (such as forgotten words, points made so far out of sequence that they don't make sense, etc.)

The group-think approach works better but often produces weak answers. The group tends to latch onto the first answer that appears to make sense. Or it latches onto the answer advocated by the loudest or most persuasive or most persistent student in the group.

It is much more effective to start from a diverse group of prepared answers, with the people who understand and can explain why they prepared the answers in they way they did.

students are picking up concepts in the testing course.

Appendix A: Exam Study Guide Questions

Note: I don't include all of the following questions in every list and I change the list from year to year. I cover different topics from year to year and some of these will be irrelevant in a given year.

It's important to keep the workload manageable. Depending on your school's culture, you might make your list longer or shorter--but don't underestimate your students. Many students will rise to a challenge, especially if they believe you are genuinely interested in their work.

The exams are closed book.

Timing, Coverage and Difficulty of the Exam

The questions in each section below vary in difficulty and length.

In drafting an exam, I answer each question that is a serious candidate for inclusion in the exam and clock my answer. To clock the answer, I write the answer out once, to get my thinking and structure down. Then I write a second draft and time that. (Remember, students have been drafting their answers in the course of studying for the exam, so on the exam, they are generating the Nth draft answer.) I allow students twice as long as it took me to hand write my second draft. For a 75 minute exam, I cumulate questions to total 55-65 minutes, leaving the extra 10-20 minutes for students who write slowly. For example, the exam might include 4 definitions, 4 short answers and 2 long answers. This particular exam offers 100 points worth of questions, but some of my 75 minute exams are out of 95 or 105 -- the total count is less important than the estimated time and difficulty of the complete product.

I rate questions as Easy, Medium, and Hard and drive the difficulty of the exam by the mix of the ratings.

Finally, I pick the questions in a way that reasonably represents what we covered in class. In some cases, the questions rely on explicitly required readings rather than on material we covered in the lecture. In some cases, the questions on the list cover material that I don't actually reach in time for the exam. These questions are excluded from the exam.

Ambiguity

One of the advantages of circulating the questions in advance is that the students can challenge them before the exam. Surprisingly, a question might be perfectly clear to the students in one semester but ambiguous to the students in the next semester.

I encourage students to draw ambiguities to my attention. I resolve the ambiguities by sending an electronic mail message to the students. I may exclude the question from the exam if the correction came too late or the answer to the corrected question is too complex.

Part 2: Short Answers (10 points each)

Short answers should take about 5 minutes to answer. In planning the timing of the exam, I allow about 6 minutes per short answer question.

1. Give two examples of defects you are likely to discover and five examples of defects that you are unlikely to discover if you focus your testing on line-and-branch coverage.
2. Give three different definitions of "software error." Which do you prefer? Why?
3. Ostrand & Balcer described the category-partition method for designing tests. Their first three steps are:

   (a) Analyze

   (b) Partition, and

   (c) Determine constraints

   Describe and explain these steps.
4. A program asks you to enter a password, and then asks you to enter it again. The program compares the two entries and either accepts the password (if they match) or rejects it (if they don't). You can enter letters or digits. How many *valid* entries could you test? (Please show and/or explain your calculations.)
5. A program is structured as follows:
   - It starts with a loop, the index variable can run from 0 to 20. The program can exit the loop normally at any value of the index.
   - Coming out of the loop, there is a case statement that will branch to one of 10 places depending on the value of X. X is a positive, non-zero integer. It can have any value from 1 to MaxInt.
   - In 9 of the 10 cases, the program executes X statements and then goes into another loop. If X is even, the program can exit the loop normally at any value of its index, from 1 to X. If X is odd, the program goes through the loop 666 times and then exits. In the 10th case, the program exits.

   Ignore the possibility of invalid values of the index variable or X. How many paths are there through this program? Please show and/or explain your calculations.
6. Consider a program with two loops, controlled by index variables. The first variable increments (by 1 each iteration) from -3 to 20. The second variable increments (by 2 each iteration) from 10

to 20. The program can exit from either loop normally at any value of the loop index. (Ignore the possibility of invalid values of the loop index.)

- If these were the only control structures in the program, how many paths are there through the program?
    - If the loops are nested
    - If the loops are in series, one after the other
- If you could control the values of the index variables, what test cases would you run if you were using a domain testing approach?
- Please explain your answers with enough detail that I can understand how you arrived at the numbers.

7. List and briefly explain three strengths of the waterfall lifecycle.
8. List and briefly explain three strengths of the evolutionary lifecycle.
9. Describe the characteristics of a good scenario test.
10. List and explain four claimed strengths of manual scripted tests and four claimed weaknesses.
11. List (and briefly describe) four different missions for a test group. How would your testing strategy differ across the four missions?
12. Distinguish between using code coverage to highlight what has not been tested from using code coverage to measure what has been tested. Describe some benefits and some risks of each type of use. (In total, across the two uses, describe three benefits and three risks.)

Part 3: Long Answers (20 points each)

4. Describe a traceability matrix.
    - How would you build a traceability matrix for the word processor in *OpenOffice*?
    - What is the traceability matrix used for?
    - What are the advantages and risks associated with driving your testing using a traceability matrix?
    - Give examples of advantages and risks that you would expect to deal with if you used a traceability matrix for the word processor. Answer this in terms of two of the main features of the word processor. You can choose which two features.
5. What is regression testing? What are some benefits and some risks associated with regression testing? Under what circumstances would you use regression tests?
6. Why is it important to design maintainability into automated regression tests? Describe some design (of the test code) choices that will usually make automated regression tests more maintainable and explain (briefly) why each choice increases maintainability.
7. Imagine testing a file name field. For example, go to an Open File dialog, you can enter something into the field. Do a domain testing analysis: List a risk, equivalence classes appropriate to that risk, and best representatives of the equivalence classes. For each test case (use a best representative), briefly explain why this is a best representative. Keep doing this until you have listed 12 best-representative test cases.
8. Consider testing the *OpenOffice* function that lets you enter data into a spreadsheet on a Presentation slide.

- How would you develop a list of risks for this capability? (If you are talking to people, who would you ask and what would you ask them?) (If you are consulting books or records or databases, what are you consulting and what information are you looking for in it?)
- Why is this a good approach for building a list of risks?
- List 10 risks associated with this function.
- For each risk, briefly (very briefly) describe a test that could determine whether there was an actual defect.

9. Imagine that you were testing the feature, Save With Password in the *OpenOffice* word processor.
   - Explain how you would develop a set of scenario tests that test this feature.
   - Describe a scenario test that you would use to test this feature.
   - Explain why this is a particularly good scenario test.
10. Imagine that you were testing the feature, Save With Password in the *OpenOffice* word processor.
    - Explain how you would develop a set of soap operas that test this feature.
    - Describe one test that might qualify as a soap opera.
    - Explain why this is a good soap opera test.
11. Imagine that you were testing the feature, Insert Object in the *OpenOffice* Presentation module.
    - Explain how you would develop a set of scenario tests for this feature.
    - Describe a scenario test that you would use to test this feature.
    - Explain why this is a particularly good scenario test.
12. Define a scenario test and describe the characteristics of a good scenario test. Imagine developing a set of scenario tests for the Outlining feature of the word processing module of *OpenOffice*. What research would you do in order to develop a series of scenario tests for Outlining? Describe two scenario tests that you would use and explain why each is a good test. Explain how these tests would relate to your research.
13. The oracle problem is the problem of finding a method that lets you determine whether a program passed or failed a test.

    Suppose that you were doing automated testing of spell-checking in the *OpenOffice* word processor.

    Describe three different oracles that you could use or create to determine whether this feature was working. For each of these oracles,

    - identify a bug that would be easy to detect using the oracle. Why would this bug be easy to detect with this oracle? and
    - identify another bug that your oracle would be more likely to miss. Why would this bug be harder to detect with this oracle?
14. You are using a high-volume random testing strategy for the *OpenOffice* word processing program. You will evaluate results by using an oracle.
    - Consider testing the spell-checking feature using oracles. How would you create an oracle (or group of oracles)? What would the oracle(s) do?
    - Now consider the placement of footnotes at the bottom of the page. How would you create an oracle (or group of oracles) for this? What would the oracle(s) do?
    - Which oracle would be more challenging to create or use, and why?

15. You are using a high-volume random testing strategy for the *OpenOffice* Presentation program will evaluate results by using an oracle.
    - Consider inserting a spreadsheet into a slide. When you enter values into the spreadsheet, you can insert functions into the cells of the spreadsheet. Think about testing those functions using oracles. How would you create an oracle (or group of oracles)? What would the oracle(s) do?
    - Consider entering a chart into a slide. Once you have entered data into the chart, *OpenOffice* draws the chart. Think about testing the chart as drawn to determine whether it properly shows the chart data. How would you create an oracle (or group of oracles)? What would the oracle(s) do?
    - Which oracle would be more challenging to create or use, and why?
16. Imagine that you were testing the *OpenOffice* word processor feature that lets you save a document in HTML format.

    Describe four examples of each of the following types of attacks that you could make on this feature, and for each one, explain why your example is a good attack of that kind.
    - Input constraint attacks
    - Output constraint attacks
    - Storage constraint attacks
    - Computation constraint attacks.

    (Notes for you while you study. Refer to Jorgensen / Whittaker's paper on how to break software. Don't give me two examples of what is essentially the same attack. In the exam, I will not ask for all 16 examples, but I might ask for 4 examples of one type or two examples of two types, etc.)
17. What is the Defect Arrival Rate? Some authors model the defect arrival rate using a Weibull probability distribution. Describe this curve and briefly explain three of the claimed strengths and three of the claimed weaknesses or risks of using this curve.

APPENDIX B: The Study Guide that I Give Students

- **Your test will be sampled from a list of questions that I give you in advance. Those questions include definitions, short answer and long answer questions. Solutions might require short essays, mathematical derivations, or code fragments.**
- **You may not use any reference materials during the test (closed book test).**
- **I recommend that you study with one or more partners. 3-5 people is a good sized group. 8 is too many.**
- **The best way to prepare for these tests is to attempt each question on your own. Your first attempt for each question should be open with no time limit. Check the lecture notes AND the required readings.**
- ***AFTER* you have tried your own answers, compare notes with your friends.**
- **Working with others will help you discover and work through ambiguities *before* you take the test. If a question is unclear, send me a note before the test. If you tell me early enough, I can fix it. If a question takes too long to answer, send me a note about that too.**
- **When you write the test, keep in mind that I am reading your answer with the goal of finding reasons to give you points:**
    - **If the question contains multiple parts, give a separate answer for each part.**
    - **If a question asks about "some", that means at least two. I normally expect three items in response to a "some". Similarly if the question asks for a list, I am expecting a list of at least three.**

- **Be aware that different words in questions have different meanings. For example, each of the following words calls for a different answer: identify, list, define, describe, explain, compare, contrast. If I ask you to list and describe some things, give me a brief identification (such as a name) of each and then a description for each one.**
    - **If you find it hard to define or describe something, try writing your answer around an example.**
    - **If you are asked to describe the relationship among things, you might find it easiest to work from a chart or a picture. You are not required to use a diagram or chart (unless I ask for one), but feel free to use one if it will help you get across your answer.**
    - **If I ask you to describe or define something that is primarily visual (such as a table or a graph), your answer will probably be easier to write and understand if you draw an example of what you are defining or describing.**
- **If I ask you for the result of a calculation, such as the number of paths through a loop, show your calculations or explain them. Let me understand how you arrived at the answer.**
- **If I ask you to analyze something according to the method described in a particular paper or by a particular person, I expect you to do it their way. If I ask you to describe their way, do so. If I ask you to apply their way, you don't have to describe it in detail, but you have to do the things they would do in the order they would do them, and you have to use their vocabulary to describe what you are doing.**
- **The test is time-limited—75 minutes. Plan to spend no more than 4 minutes on any definition, no more than 10 minutes on any short answer, and no more than 15 minutes on any long answer. Spend less on most answers. Suppose the test has 4 definitions (20 points), 2 short answers (20 points), and 3 long answers (60 points). You should plan to spend, on average, about 3 minutes per definition, about 8 minutes per short answer, and about 12 minutes per long answer (total = 64 minutes). Use the remaining 11 minutes to check your work.**
- **Pick the order of your answers. If you spend too long on definitions, start writing your long answers first. If you are nervous, start with the questions you find easiest to answer.**

Study Guide Suggestions -- Page 2

- **Be aware of some factors that, in general, bias markers. These are generalizations, based on research results. I try, of course, to be unbiased, but it's a good idea to keep these in mind with ANY grader for ANY exam:**
    - ***Exams that are hard to read tend to get lower grades*. Suggestions: Write in high contrast ink (such as black, medium). Write in fairly large letters. Skip every second line. Don't write on the back of the page. If your writing is illegible, print. *If I can't read something you wrote, I will ignore it.***
    - ***Start a new question on a new page*. More generally, *leave lots of space on the page.* This gives you room to supplement or correct your answer later (when you reread the exam before handing it in) and it gives me room to write comments on the answer, and it makes the answer easier to read.**
    - ***Answers that are well-organized are more credible*. Suggestions: If the question has multiple parts, start each part on a new line, and identify each part at its start. In a list, start each list item on a new line—maybe bullet the list. For example, consider the question: "What is the difference between black box and white box testing? Describe the advantages and disadvantages of each." You could organize this with five headings:**
        - **Difference between black and white**
        - **Advantages of black box**

        - **Disadvantages of black box**
        - **Advantages of white box**
        - **Disadvantages of white box**
    - ***Don't answer what has not been asked*. For example, if I ask you to define one thing, don't define that and then give me the definition of something related to it. If you do, (a) I won't give you extra credit, (b) I'll think that you don't know the difference between the two things, and (c) if you make a mistake, I'll take off points.**
    - ***Give the number of items requested.* For example, if I ask for two scenario tests, don't give one or three. If you give one, you miss points. If you give three, I will either grade the first two and ignore the third (this is my normal approach) or grade the first two that I happen to read (whatever their order on the page) and ignore the third. I will never read the full list and grade what I think are the best two out of three.**

Additional points to consider.

- **Beware of simply memorizing some points off a slide. If I think you are giving me a memorized list without understanding what you are writing, I will ruthlessly mark you down for memorization errors. In general, if you are repeating a set of bullet points, write enough detail for them that I can tell that you understand them.**
- **Use a good pen. Lawyers and others who do lots of handwriting buy expensive fountain pens for a reason. The pen glides across the page, requiring minimal pressure to leave ink. If you use a fountain pen, I suggest a medium point (write large) to avoid clogging. Also try gel pens or rollerballs. Get one that you can write with easily, to avoid writer's cramp. Basic ballpoints are very hard on you. Look at how tightly you hold it and feel how hard you press on the page.**

Appendix C Grading the Exam

Here are examples of how I have graded exam questions. These notes are a bit cleaned up, to make them understandable to another reader. However, I've left in several discussions that I inserted in my grading notes at the time that I graded the question.

- Most of these are about how to grade the question
- Others are about what will have to be taught in the future, or provided in reference materials, to help students achieve better grades next time.
- I often include reference material or lists in my grading notes. These are often there simply to jog my memory. Sometimes, however, they are there to remind me of what I have to treat as acceptable. The problem is that students, relying on reputable sources (including other courses) will give answers that, in my humble opinion, are mistaken or ridiculous. I will accept many of these answers even though I disagree with them, but to maintain grading consistency, I need a list of what I will accept and what I will not accept. Otherwise, my tolerance of some of these answers will vary too much with my mood.

The critical features of my grading structure (what you'll always see in my working notes) are:

- Table format, with one column for each point-deserving type of information. (One row per student.) For all but the most trivial questions, I actually fill in the table for each question. This gives me detailed information about each student's performance on each question, which I use to good advantage when a student comes to me to question her grade.

- For most questions, including all complex questions, I show the points available within the column on the column itself.
- An outline of a sufficient answer.
- On a complex question, I will often paste in a list or discussion from lecture notes or one of the readings.

Suppose that a question is worth 20 points.

- In the columns, I might allow up to 30 points. This is primarily because different people legitimately approach the same question in different ways and so my grading structure has to allow for this.
- No matter what the student's total point count is, I sometimes reserve the 20th point for style and organization. That is, you can get 19/20 based on the standard point count, but I won't give a perfect grade unless I think the answer is well written. This is an especially common decision when the total of available points is beyond 20, and so a disorganized answer that covers lots of ground would otherwise get a perfect grade.
- Also, some questions are just too hard. I might allow 1 or 2 points merely for attempting the question.
- One of my frequent columns is "clue." This is a source of discretionary points. I define the discretionary rule on a question-by-question basis. Some examples:
    - If I allow up to 1 point for Clue, the default is 0 points. If the student's answer shows more insight than I think is reflected in the point count for the answer, I will raise the grade by 0.5 or 1 points.
    - If I allow up to 2 points for Clue, the default is probably 0 points, but the most common score is probably 1 point.

Definitions 5 points each

Power of a test	Probability notion	If bug is there	Ability to reveal bug	Stat comparison	Grade

Power of a test.

If two tests are potentially capable of exposing the same type of defect, one test is more powerful if it is more likely to expose the defect.

- likelihood of revealing (or ability to reveal) a defect if the defect is there <I give a max of 3.5/5 if the answer doesn't point out that this depends on whether the bug is there.>
- if two tests can reveal the same defect equally well, the more powerful test can also reveal other defects
- analogous to the concept of statistical power

Exploratory testing	Test and learn in parallel	about product, risks, market, test methods	Don't follow pre-existing detailed plan	Whittaker's attacks	Other	Final / 5

Exploratory Testing involves simultaneous learning, testing, evaluating, planning, and reporting.

Alternative answer: simultaneous testing and learning, plus mention of Whittaker's attacks. We discuss *How to Break Software* in class. In a different class, the appropriate source of examples might come from Hendrickson's *Bug Hunting* slides, etc.

Storage constraints					

[Note to reader: the concept of "storage constraints" comes from a paper by Whittaker & Jorgenson, that was expanded in Whittaker's book, *How to Break Software.* We look at four fundamental types of constraints, input constraints, output constraints, storage constraints, and computational constraints. Many bugs are caused by a programmer's failure to consider the possibility of violation of one of these types of constraints.]

I'd like to see something about data structures. I expect to see discussion of limitation of storage in terms of the types of data that are stored or the place of storage, not input/output/computation overflows. The attacks on storage are attacks on the data structure, *how* the data is stored.

Future notes: We need the additional details available Why Software Fails for this to be a good question.

From James Whittaker and Alan Jorgensen's (2000) paper, *How to Break Software*

> "Data is the lifeblood of software; if you manage to corrupt it, the software will eventually have to use the bad data and what happens then may not be pretty. It is worthwhile to understand how and where data values are established.
>
> "Essentially, data is stored either by reading input and then storing it internally or by storing the result of some internal computation. By supplying input and forcing computation, we enable data flow through the application under test. The attacks on data follow this simple fact as outlined in attacks 12-14. However, without access to the source code, many testers do not bother to consider these attacks. We believe, though, that useful testing can be done even though specifics of the data implementation are hidden. We like to tell our students to practice "looking through the interface." In other words, take note of what data is being stored while the software system is in use. If data is entered on one screen and visible on another, then it is being stored. Information that is available at any time is being stored.
>
> "Some data is easy to see. A table structure in a word processor is one such example in which not only the data but the general storage mechanism is displayed on the screen. Some data is hidden behind the interface and requires analysis to discover its properties.
>
> "Once the nature of the data being stored is understood, try to put yourself in the position of the programmer and think of the possible data structures that might be used to store such data. The more that programming and data structures are understood, the easier it will be to execute the following attacks. The more completely you understand the data you are testing, the more successful the attacks will be at finding bugs.
>
> **"Twelfth attack: Apply inputs using a variety of initial conditions**
>
> Inputs are often applicable in a variety of circumstances. Saving a file, for example, can be performed when changes have been made, and it can also be performed when no changes have been made. Testers are wise to apply each input in a number of different circumstances to account for the many such interactions that users will encounter when using the application.
>
> **"Thirteenth attack: Force a data structure to store too many/too few values**
>
> "There is an upper limit on the size of all data structures. Some data structures can grow

to fill the capacity of machine memory or hard disk space and others have a fixed upper limit. For example, a running monthly sales average might be stored in an array bounded at 12 or fewer entries, one for each month of the year.

"If you can detect the limits on a data structure, try to force too many values into the structure. If the number is particularly large, the developer may have been sloppy and not programmed an error case for overflow.

"Special attention should be paid to structures whose limits fall on the boundary of data types 255, 1023, 32767 and so on. Such limits are often imposed simply by declaration of the structure's size and very often lack an overflow error case.

"Underflow is also a possibility and should be tested as well. This is an easy case, requiring only that we delete one more element than we add. Try deleting when the structure is empty, then try adding an element and deleting two elements and so on. Give up if the application handles 3 or 4 such attempts.

"Fourteenth attack: Investigate alternate ways to modify internal data constraints

"The phrase "the right hand knoweth not what the left hand doeth" describes this class of bugs. The idea is simple and developers leave themselves wide open to this attack; in most programs there are lots of ways to do almost anything. What this means to testers is that the same function can be invoked from numerous entry points, each of which must ensure that the initial conditions of the function are met.

"An excellent example of this is the crashing bug one of our students found in PowerPoint, regarding the size of a tabular data structure. The act of creating the table is constrained to 25×25 as the maximum size. However, one can create such a table, then add rows and columns to it from another location in the program—crashing the application. The right hand knew better than to allow a 26×26 table but the left hand wasn't aware of the rule."

Equivalence class					

Two tests are members of the same equivalence class if you expect the same results from each. Tests are equivalent with respect to a theory of error. Two tests might be equivalent relative to one potential failure and entirely different with respect to a different potential failure.

Short Answers 10 points each

	Nested analysis	Series analysis	First variable -3,0,20	Second variable 10,20	Used invalid values	Total
<Points>	4	3	2	2	ignore	10

Consider a program with two loops, controlled by index variables. The first variable increments (by 1 each iteration) from -3 to 20. The second variable increments (by 2 each

iteration) from 10 to 20. The program can exit from either loop normally at any value of the loop index. (Ignore the possibility of invalid values of the loop index.)

- **If these were the only control structures in the program, how many paths are there through the program?**
  - **If the loops are nested**
  - **If the loops are in series, one after the other**
- **If you could control the values of the index variables, what test cases would you run if you were using a domain testing approach?**
- **Please explain your answers with enough detail that I can understand how you arrived at the numbers.**

Analysis

- The first variable has 24 possible values (-3, -2, -1, 0, 1, ..., 20)
- Second variable has 6 possible values (10, 12, 14, 16, 18, 20)

a) Analysis if loops are nested.

Suppose loop 1 was 1, 2, 3, Suppose loop 2 was 4,5

Loop 1 with loop 2 inside it =

2 one through loop (1,4) (1,5)

4 twice through loop (1,4,2,4), (1,4,2,4,5), (1,4,5,2,4), (1,4,5,2,4,5)

8 three through loop (1,4,2,4,3,4), (1,4,2,4,3,45), (1,4,2,45,3,4), (1,4,2,45,2,45), (1,45,2,4,3,4), (1,45,2,4,3,45), (1,45,2,45,3,4), (1,45,2,45,3,45)

Illustrates the general rule:

If N1 = number of values of the outer loop and N2 = number of values of the inner loop,

Number of paths = sum (i=1 to N1) N2^i

Example 2 + 2*2 + 2*2*2 for our sample loop

The sum is therefore (sum)(i=1-to-24) 6^i

1	6
2	36
3	216
4	1296
5	7776
6	46656
7	279936
8	1679616
9	10077696
10	60466176
11	362797056
12	2176782336
13	13060694016
14	78364164096
15	4.70185E+11
16	2.82111E+12
17	1.69267E+13
18	1.0156E+14

19	6.0936E+14
20	3.65616E+15
21	2.1937E+16
22	1.31622E+17
23	7.8973E+17
24	4.73838E+18
	5.68606E+18

b) Analysis if the loops are in series

- Total = 24 x 6 = 144 paths
- First variable -3, 0, 20 (Ignore the possibility of invalid values of the loop index)
- Second variable 10, 20

[Note to the reader: Students who handled the nested analysis well handled everything else well. In contrast, some students who correctly counted the variables' values, figured out the series, and seemed to handle the material reasonably comfortably, blew the nested analysis.

- If the nested analysis is done well, it probably deserves more than 4 points out of 10 -- but the student doesn't need the points.
- If the nested analysis is not done well, the 4 point allowance serves as a cap on the damage this part of the question can do to the grade for the full question.]

	Not tested	Measure	Benefit N	Risk N	Benefit M	Risk M	
	2	2	1 each	1 each	1 each	1 each	

Distinguish between using code coverage to highlight what has not been tested from using code coverage to measure what has been tested. Describe some benefits and some risks of each type of use. (In total, across the two uses, describe three benefits and three risks.)

Grading notes:

Emphasis on what has not been tested: you are looking for such things as blind spots in the testing, or reality check on the process or on the projected ship date. There is no necessary claim that this is a valid progress measure. You are merely identifying a set of tasks that have not been done.

Emphasis on measurement: Assumption is that coverage is a valid measure of testing process. You are looking for status check or productivity of staff member or group or nearness to completion.

Benefits and risks of highlighting the negative:

Benefits:

- reveal problems with the testing process
- reveal weaknesses or blind spots of the testing strategy
- reveal the overall utility of a collection of testing artifacts (no point maintaining a large test suite that achieves only 2% coverage)

- reveal impossibility of a ship date

Risks:

- blaming tone
- might persuade managers to rely on this, in a way that encourages them to use coverage as a measurement of progress later.

Benefits and risks of measurement:

Benefits:

- for exam purposes, I will accept the notion that we can check nearness to completion of testing with this measure
- can note progress against a plan
- can report results in a way managers are used to hearing
- one factor in a ship decision

Risks:

- encourages people to do things that are counted rather than things that are more likely to reveal problems
- discourages people from tests (e.g. configuration tests) that are not counted
- gives mgmt the false perception of progress because it omits key tests that are not counted.
- Encourages premature release of the product
- Discriminates against testers who do tests that are "redundant" under this measure

[References: Marick's writings on coverage, such as Classic Testing Mistakes and How to Misuse Code Coverage.]

7 ***Month Field***	Boundary chart *	Realize Inter-depence	28 day month	29 day leap year	30 day	31 day	Month Range	Year Range	Invalid pairings	Invalid max min	Invalid chars (shotgun penalty -- they should **not** appear)	Tests of full field (all 3 values in 1 test)	Overall analysis	Total / 24	Grade / 10
Points	2	2	2	2	2	2	2	2	2	2	-2	4	2	24	10*

* = discretionary boost of up to one point allowed for "Clue", should be rarely given.

Imagine testing a date field. The field is of the form MM/DD/YYYY (two digit month, two digit day, 4 digit year). Do an equivalence class analysis and identify the boundary tests that you would run in order to test the field. (Don't bother with non-numeric values for these fields.)

Grading notes--

- "overall analysis"--refers to the discussion and presentation of the analysis.
- A boundary chart is not compulsory, but some organized presentation of the material is. These 2 points are for the presentation of the answer
- Interdependence: the valid days will differ depending on which month and whether we are in leap year or not.
- Invalid pairings: there must be tests of invalid combinations, such as Feb 29 in a non-leap-year or June 31. [If the student shows no thinking about invalid combinations, deduct additional points from "overall analysis"]
- There's a 10% grading penalty for wasting space on non-numeric values. These don't belong in the answer (read the question), so the student is writing a shotgun answer. I don't always have the opportunity to penalize for defocused shotguns, but this is such an obvious situation that I am glad to take advantage of it. It lets me make a point about sticking to the call of the question when I review mid-term test results.

Equivalence classes

- <u>valid days.</u> There are 4 equivalence classes for days.
- All are 0-x, where x=28, 29, 30, 31 depending on the months and leap year
    1. month = 28 day
    2. month = 29 day leap year
    3. month = 30 day
    4. month = 31 day
- <u>valid months</u> 1-12
- <u>years</u> 0-9999 or some other plausible range

List tests

(a) {28 day month} {0,1,28,29} {0,2000,9999,10000, leap year}

(b) {29 day month} {0,1,29,30} {0,2000,9999,10000, leap year}

(c) {30 day month} (0,1,30,31} {0,2000,9999,10000, leap year}

(d) {31 day month} {0,1,31,32} {0,2000,9999,10000, leap year}

In other words, for tests of type (a), pick a 28 day month (February) and test with one of the numbers in the set {0, 1, 28, 29} and with one of the numbers in the set {0,2000,9999,10000, any leap year}.

Minefield question	E1 2	E2 2	E3 2	E4 2	E5 2	Total 10

Three different missions	Mission 1	Strategy 2	Mission 1	Strategy 2	Mission 1	Strategy 2	Clue 1	

List three different missions for a test group. How would your testing strategy differ across the three missions?

Grading Notes:

A plausible relationship between the mission and the strategy / activity is sufficient for full credit, but the strategy / activity description has to go beyond a statement of fulfilling the mission. For example, if the mission is “Find defects”, the testing strategy has to say something more than “find defects” such as concentrating on stress tests, concentrating on high risk areas of the product, etc.

Varying definitions of test groups (from course slides)

- Find defects
- Maximize bug count
- Block premature product releases
- Help managers make ship / no-ship decisions
- Assess quality
- Minimize technical support costs
- Conform to regulations
- Minimize safety-related lawsuit risk
- Assess conformance to specification
- Find safe scenarios for use of the product (find ways to get it to work, in spite of the bugs)
- Verify correctness of the product
- Assure quality

“Clue” is discretionary, typical value is 0.5. If the mission strategies are weak but I give them a full credit, I’ll not award the clue point (it was already awarded).

Goodness of tests	D1 2.5	D2 2.5	D3 2.5	D4 2.5	total

List and describe four different dimensions (different “goodnesses”) of “goodness of tests”.

Grading Notes:

1 point for the item and 1.5 for the description

List from course:

- More powerful
- More credible

- Provides better support for troubleshooting
- Representative of a broader group of tests
- Is representative of events more likely to be encountered by the customer
- Is more likely to help the tester or developer develop an insight into the program
- Is easier to automate, easier to evaluate, more feasible, lower opportunity cost

Long Answers 20 points each

11. Scenario	Explain how develop	Describe	Explain why good	total
	2 pts each good idea -2 if not for set max 8	Max 5 (well described, good scenario)	Tie facts of test to stated elements, 2 per element Max 8	

Imagine that you were testing the feature, *Save With Password* in the *OpenOffice* word processor.

- **Explain how you would develop a set of scenario tests that test this feature.**
- **Describe a scenario test that you would use to test this feature.**
- **Explain why this is a particularly good scenario test.**

Grading Notes

Scenario tests:

- **Explain how you would develop a set of scenario tests for this feature. I expect a range of possible ideas**
  - Research
    - Customers
    - Competitors
    - In-house documentation for tasks
  - Implementation

  Note that we're talking about a SET, not just one. If the explanation is appropriate only for a single scenario test (not for a set), deduct points.
- **Describe a scenario test you would use. I evaluate it against**
  - Realistic
  - Complex
  - Unambiguous

    - Persuasive / credible to stakeholder
- **Why is THIS a particularly good test**
    - Tie the facts to the elements

Name	How many Combs 2	What is all Pairs table 4	Create Table 10	Why Correct 4	Total 20

We are going to do some configuration testing on the *OpenOffice* word processor. We want to test it on

- **Windows 95, 98, and 2000 (the latest service pack level of each)**
- **Printing to an HP inkjet, a LexMark inkjet, and a Xerox laser printer**
- **Connected to the web with a dial-up modem (28k), a DSL modem, and a cable modem**
- **With a 640x480 display and a 1024x768 display**
    - **How many combinations are there of these variables?**
    - **Explain what an all-pairs combinations table is**
    - **Create an all-pairs combinations table**
    - **Explain why you think this table is correct.**

Grading notes--

- Combinations 3x3x3x2=54
- "explain" and "why correct?" are essentially the same question. The second one is an opportunity for the student to look back and check the work as she starts writing her criterion.
- Students who blow the combination chart (by missing all pairs) are capped at 50%. This looks harsh, but this is a very easy table and the students have had it for plenty of time. They shouldn't get this wrong.

Test Plan	Strategy	Q1	Guide 1	Q2	guide 2	Q3	Guide 3	Q4	Guide 4	Q5	Guide 5	Q6	Guide 6	Total / 20

Imagine that you are an external test lab, and Sun comes to you with *OpenOffice*. They want you to test the product. How will you decide what test documentation to give them? (Suppose that when you ask them what test documentation they want, they say that they want something appropriate but they are relying on your expertise.) To decide what to give them, what questions would you ask (4 to 6 questions) and how would the answers to those questions guide you?

Grading notes-

[Note: I have refined the wording of this question since grading the exam in which this question appeared. This analysis will be different next time because you won't be asked for an overall strategy. "How will you decide what test documentation to give them?" is deleted.]

With "typical" points and 4 questions, the student gets 16 plus up to 3 for strategy.

With "typical" points and 6 questions offered, the student can get 24 plus up to 3 for strategy,

Maximum points possible are 39 (1 per question, 5 per guide across 6 questions, plus up to 3 for strategy)

I reserve discretion over "A" and may slightly raise or lower an answer if it is in the 18-20 total range.

Strategy: How will you decide what test documentation to give them? *An answer that says, I'll ask them questions, is worth 0 points because I've said to ask questions.* On the other hand, if they add extra research ideas beyond asking questions, they can have 1 (typical) to 3 (professional-level) points.

Questions:

- The question alone gets one point for itself. They got a list of questions in the course, there is nothing original here, just very simple memory work.
- The question alone gets no guidance points, zero. Guidance should take the form of a specific statement of impact (of the answer) on the content or structure of the test documentation. An exceptionally insightful and useful guidance answer can earn 5 points. An adequate (the typical) answer earns 3 points. A weak answer earns 0-2. *For examples of impact discussions, see the test documentation chapter in Lessons Learned in Software Testing. This was required reading in the course.*
- Some people misread the question as calling for an aggregate judgment on the value of the answers. I added back up to 6 points for the aggregate evaluation.

See course slides on requirements questions:

- *Is test documentation a product or tool?*
- *Is software quality driven by legal issues or by market forces?*
- *How quickly is the design changing?*
- *How quickly does the specification change to reflect design change?*
- *Is testing approach oriented toward proving conformance to specs or nonconformance with customer expectations?*
- *Does your testing style rely more on already-defined tests or on exploration?*
- *Should test docs focus on what to test (objectives) or on how to test for it (procedures)?*
- *Should control of the project by the test docs come early, late, or never?*
- *Who are the primary readers of these test documents and how important are they?*
- *How much traceability do you need? What docs are you tracing back to and who controls them?*
- *To what extent should test docs support tracking and reporting of project status and testing progress?*
- *How well should docs support delegation of work to new testers?*
- *What are your assumptions about the skills and knowledge of new testers?*
- *Is test doc set a process model, a product model, or a defect finder?*

- *A test suite should provide prevention, detection, and prediction. Which is the most important for this project?*
- *How maintainable are the test docs (and their test cases)? And, how well do they ensure that test changes will follow code changes?*
- *Will the test docs help us identify (and revise/restructure in face of) a permanent shift in the risk profile of the program? Should docs (be) automatically created as a byproduct of the test automation code?*

Additionally, we might see the Phoenix questions (these are listed in Thinkertoys) or other context-free questions (see Gause & Weinberg, Exploring Requirements).

Oracle	Hyphenation 8	Footnotes 8	Compare 6	Total 20

You are using a high-volume random testing strategy for the *OpenOffice* word processing program. You will evaluate results by using an oracle.

- **Consider testing the hyphenation feature using oracles. How would you create an oracle (or group of oracles)? What would the oracle(s) do?**
- **Now consider the placement of footnotes at the bottom of the page. How would you create an oracle (or group of oracles) for this? What would the oracle(s) do?**
- **Which oracle would be more challenging to create or use, and why?**

Note: If you don't understand hyphenation, substitute "spell checking" for "hyphenation in this question.

Grading notes.

In 2002, I applied gentle grading because we didn't spend enough time on this in class for a detailed answer. Additionally, just before the exam, it became clear that several foreign students were befuddled about hyphenation. So, I sent out a note allowing spell checking instead, and included this as one of the easy questions on the exam.

Hyphenation:

- How would you create an oracle?
  - Use of prior version is worth 4 points (of 8). The problem is that there's no reason to believe the prior version works. Add points for discussion of this issue.
  - Compare to competitor is worth 4-8 points depending on whether the answer deals with the question of how we know the competitor works
  - We could run both word perfect and word in parallel and raise the flag if they disagree with our result

- We could build random sentences from a small vocabulary of words that have known hyphenation characteristics, then check whether they were hyphenated properly against a list
- We might run only a partial oracle that looks at hyphenation under some simple rules.

▪ What would it do?

Spell checking

▪ How

▪ What would it do

Footnotes

▪ There are several things to check here--placement on the page, agreement between the reference mark (e.g. footnote number) in the body and the one in the footnote, formatting of the footnote, formatting of the reference mark, break of the long footnotes across pages, formatting of tables in footnotes, etc.

▪ How
  - Use of prior version is worth 3 points (of 7). The problem is that there's no reason to believe the prior version works. Add points for discussion of this issue.
  - Compare to competitor is worth 3-7 points (of 7) depending on whether the answer deals with the question of how we know the competitor works
  - We could run both word perfect and word in parallel and raise the flag if they disagree with our result
  - We might run a partial oracle that looks only at some of the footnoting issues.
  - If you write your own, how do you know it works and how do you decide what to include. Are you designing it in a way that makes it likely to be suitable for high-volume work?

▪ What would it do

Which oracle is more challenging and why

▪ Footnoting is much more challenging because there are so many variables in play. Consider the problem of placement at the bottom of the page, carrying long notes across pages, formatting of tables and pictures inside footnotes, etc.

▪ However, a *reasoned* argument in favor of the other (hyphenation or spellcheck) will be accepted to the extent that it is credibly argued.

follow-up testing	Steps 3	Options 3	Configs 3	Generality (3)	Other (3)	Example1 4	example2 4	Example3 4	Total 20

Suppose that you find a reproducible failure that doesn't look very serious.

- **Describe three tactics for testing whether the defect is more serious than it first appeared.**
- **As a particular example, suppose that the display got a little corrupted (stray dots on the screen, an unexpected font change, that kind of stuff) in *OpenOffice*'s word processor when you drag the mouse across the screen. Describe three follow-up tests that you would run, one for each of the tactics that you listed above.**

Grading Notes

Describe three tactics for testing: If you list an item without describing it, only 1 point.

Each description is worth 3 points, each example is 4 points. That totals 21, but from 19 to 20 is my discretion

My examples of follow-up tests

- Tests related to my steps
    - Enter more data into the table
    - Enter data into the table lots of times (repeat same entries)
- Tests related to the structure of the situation
    - Vary the size of the table
    - Vary the contents of the table
    - Vary the color, alignment, font, line width, etc of the table entries
- Tests related to the failure
    - ?? what else causes mouse droppings ??
    - print preview the screen
- Tests related to the persistent variables / options
    - Location of the program within the window
    - Whether the program is maximized
    - Default cell format, such as alignment within the cells, font, line width, etc.
- Tests related to the hardware
    - Different video resolution
    - Different monitors
    - Different video cards
    - Different OS (check one of the ports, is this unique? If not, then anything specific to windows might be irrelevant)
    - Different mouse / mouse driver
    - Different memory

Appendix D: Sample Assignments

Assignment 1: Create a first test case chart

In lecture, we brainstormed answers to the following:

> **The program reads three integer values from a card. The three values are interpreted as representing the lengths of the sides of a triangle. The program prints a message that states whether the triangle is scalene, isosceles, or equilateral.**
>
> » From Glen Myers, *The Art of Software Testing*
>
> - ***Write a set of test cases that would adequately test this program.***
> - ***Please write your name on your answer so that I can return it to you. Hand it in when you are done.***

Let's take one more crack at Myers' exercise. I'd like a table like this from you that lists 5 good test cases:

Test	Test Case	Risk	Why this test is powerful	Expected Result
1				
2				
3				
4				
5				

Where the columns are defined as:

- Test—fill in the test case number
- Test case—be specific about the inputs that you'll feed the program. For example, 1,1,2
- Risk—be specific about the error you are trying to detect. For example (from 1,1,2), the risk is that the program might accept a "triangle" that has side 1 + side 2 = side 3.
- Why this test is powerful—A test is powerful, compared to other tests, if it is more likely to expose a failure than they are. If you test for a specific type of failure (the risk), use a powerful test – one that is at least as likely to expose that failure as any other. To fill in this part of the answer, explain why you think the test you chose is powerful. It might be helpful to provide examples of similar tests that are less powerful than this one. If you think that this test is equivalent to many others (no more, no less powerful), say "Equivalent" and list 2 or 3 other tests

that you think are equivalent to this one. This is an easy answer—but if I can easily spot a better test against the same risk, you won't get credit for saying "equivalent".

- Expected result—What should the program do? You might respond, "Error Message."

Assignment Instructions

- Feel free to work in groups
- If you work with others, make sure to name them.
- If you work in a group, it is OK for the group to hand in one collective answer that you all co-sign. (Just provide your name, I don't need your student number.) However, if you co-sign it, you must be able to able to explain EVERY test case that you have on the page.
- I expect better work from a group than from individuals and I will mark accordingly.
- If you work in a group, I expect 5 tests from each of you. So, if there are two of you in the group, the assignment should contain 10 tests.
- Please don't submit 100 tests. Pick 5 good ones per person. Prioritizing among possible tests is one of the important skills of good testers.

Grading Rubric for the Triangle problem

This rubric is given to the students. For each test, the first column indicates the points to be awarded if you give an answer that has the characteristics listed in the second column. Each test can be awarded up to 10 points.

Test Case:

0	Values for the sides of a triangle not provided or incorrect.
1	Values for the sides of the triangle provided and correct.

Risk:

0	Implausible or generic to the point of no value. Not a risk.
1	Generic Risk statement, Vague not related to power.
2	Less clear or less plausible but related to power.
3	Clear statement of plausible risk not related to power or Less clear or less plausible risk but related to power.
4	Clear statement of plausible risk related to power.

Power:

0	No clear linkage between the power discussion and the risk.
1	Shows that the test can detect the error identified in the risk.
2	The test can detect an error, indicate how or why and provide some comparison to other tests. Some indication that this is a good test.
3	Good comparison examples but weak explanation otherwise. A sufficiently strong explanation but without examples. Must indicate this is better than the others.
4	State the principles under which this is more powerful and gives a persuasive example.

Expected Result:

0	Expected result not provided or incorrect.
1	Expected result provided and correct.

Assignment 2: Replicate and Edit Bugs

The purpose of this assignment is to give you experience editing bugs written by other people. This task will give you practice thinking about what a professional report should be, before you start entering your own reports into this public system.

- Work with *OpenOffice* Writer, the word processor.
- Read the instructions at http//qa.*OpenOffice*.org/helping.html, and make sure to use the oooqa keyword appropriately. Read the bug entry guidelines at http://www.*OpenOffice*.org/bugs/bug_writing_guidelines.html.
- Find 5 bug reports in IssueZilla about problems with *OpenOffice* Writer that appear to have not yet been independently verified. These are listed in the database as "unconfirmed" or "new". As of 9/1/2002, there are 927 such reports associated with the "word processor" component. To find lots of bugs, use the search at http://www.*OpenOffice*.org/issues/query.cgi rather than at http://qa.*OpenOffice*.org/issuelinks.html.
- For each report, review and replicate the bug, and add comments as appropriate to the report on issuezilla.
- Send me an email with the bug numbers and for each bug, with comments on what was done well, what was done poorly and what was missing that should have been there in the bug report.

Assignment Procedure

For each bug:

- Review the report for clarity and tone (see "first impressions", next slide).
    - Send comments on clarity and tone in the notes you send me (but don't make these comments on the bug report itself)
- Attempt to replicate the bug.
    - Send comments to me on the replication steps (were the ones in the report clear and accurate), your overall impressions of the bug report as a procedure description, and describe any follow-up tests that you would recommend.
- You may edit the bug report yourself, primarily in the following ways.
    - Add a comment indicating that you successfully replicated the bug on XXX configuration in YYY build.
    - Add a comment describing a simpler set of replication steps (if you have a simpler set). Make sure these are clear and accurate.
    - Add a comment describing why this bug would be important to customers (this is only needed if the bug looks minor or like it won't be fixed. It is only useful if you clearly know what you are talking about, your tone is respectful).
    - Your comments should NEVER appear critical or disrespectful of the original report or of the person who wrote it. You are adding information, not criticizing what was there.
- If you edit the report in the database, **never change what the reporter has actually written**. You are not changing his work, you are adding comments to it at the end of the report
- Your comments should have your name and the comment date, usually at the start of the comment, for example: "(Cem Kaner, 12/14/01) Here is an alternative set of replication steps:")
- Send me an email, telling me that you have reviewed the report and made changes.

A Checklist for Editing Bugs

The bug editor should check the bug report for the following characteristics:

A. First impressions—when you first read the report:

1. Is the summary short (about 50-70 characters) and descriptive? (see the slide: *Important Parts of the Report: Problem Summaries*)
2. Can you understand the report? As you read the description, do you understand what the reporter did? Can you envision what the program did in response? Do you understand what the failure was?
3. Is it obvious where to start (what state to bring the program to, to replicate the bug)?
4. Is it obvious what files to use (if any)? Is it obvious what you would type?
5. Is the replication sequence provided as a numbered set of steps, which tell you exactly what to do and, when useful, what you will see?
6. Does the report include unnecessary information, personal opinions or anecdotes that seem out of place?
7. Is the tone of the report insulting? Are any words in the report potentially insulting?
8. Does the report seem too long? Too short? Does it seem to have a lot of unnecessary steps? (This is your first impression—you might be mistaken. After all, you haven't replicated it yet. But does it LOOK like there's a lot of excess in the report?)
9. Does the report seem overly general ("Insert a file and you will see" – what file? What kind of file? Is there an example, like "Insert a file like blah.foo or blah2.fee"?)

B. When you replicate the report:

10. Can you replicate the bug?
11. Did you need additional information or steps?
12. Did you get lost or wonder whether you had done a step correctly? Would additional feedback (like, "the program will respond like this...") have helped?
13. Did you have to guess about what to do next?
14. Did you have to change your configuration or environment in any way that wasn't specified in the report?
15. Did some steps appear unnecessary? Were they unnecessary?
16. Did the description accurately describe the failure?
17. Did the summary accurate describe the failure?
18. Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not?

C. Closing impressions:

19. Does the description include non-factual information (such as the tester's guesses about the underlying fault) and if so, does this information seem credible and useful or not? (The report need not include information like this. But it should not include non-credible or non-useful speculation.)
20. Does the description include statements about why this bug would be important to the customer or to someone else? (The report need not include such information, but if it does, it should be credible, accurate, and useful.)

D. Follow-up tests:

21. Are there follow-up tests that you would run on this report if you had the time? (There are notes on follow-up testing in the course slides 105-117)?
22. What would you hope to learn from these tests?
23. How important would these tests be?
24. You will probably NOT have time to run many follow-up tests yourself. Don't take the time to run more than 1 or 3 such tests.
25. Are some tests so obvious that you feel the reporter should run them before resubmitting the bug? Can you briefly describe them to the reporter?
26. Some obvious style issues that call for follow-up tests—if the report describes a corner case without apparently having checked non-extreme values. Or the report relies on other specific values, with no indication about whether the program just fails on those or on anything in the same class (what is the class?) Or the report is so general that you doubt that it is accurate ("Insert any file at this point" – really? Any file? Any type of file? Any size? Maybe this is accurate, but are there examples or other reasons for you to believe this generalization is credible?)

GRADING NOTES FOR THE BUG EDITING ASSIGNMENT

Two components for grading the papers –

1) Comments at Issuezilla (the *OpenOffice* database)
2) Editor's report submitted to us.

I allocated 14 points possible for each bug, but totalled out of 10. That is, if you got a 3/14 for the bug, your score was changed to 3/10. Similarly, 14/14 became 10/10. There were 10 points available for each bug.

COMMENTS ON THE BUG REPORTS THEMSELVES, FILED IN ISSUEZILLA

[NOTE: This was prepared as feedback for students but can be easily turned into a rubric.]

The content of your comments has to vary depending on the problem. The key thing is that the follow-up report has to be useful to the reader.

For example, a simple failure to replicate might be sufficient (though it is rarely useful unless it includes a discussion of what was attempted.) Sometimes, detailed follow-up steps that simplify or extend the report are valuable.

This is worth up to 7 points out of 10

	Subcomponents of the comments at Issuezilla	**Points possible**
1	Report states configuration and build	+ Up to 1
2	If the report is disrespectful in tone, zero the grade for the report.	0 for the report
3	If you clearly report a simpler set of replication steps	+ Up to 5
4	If you clearly report a good follow-up test	+ Up to 5

5	A follow-up test or discussion that indicates that you don't understand the bug is not worth much.	+ Up to 1
6	If there is enough effort and enough usable information in the follow up test.	+ Up to 3
7	If you make a good argument regarding importance (pro or con)	+ Up to 5
8	If the bug is in fact not reproducible, and the report demonstrates that you credibly tested for reproducibility	+ Up to 5
9	Nonreproducible bug on alternate configuration without discussion	- 1
10	Nonreproducible bug on alternate configuration that was already dismissed	- 2

REPORT TO US

This is worth up to 7 points out of 10

Here, you evaluated the report rather than trying to improve it. I wanted to see details that suggested that you had insight into what makes bug reports good or bad, effective or ineffective. I did not expect you to walk through every item in the checklist and tell me something for each item (too much work, most of it would have wasted your time). Instead, I expected that you would raise a few issues of interest and handle them reasonably well. For different reports, you might raise very different issues.

1) I was interested in comments on:

a) What was done well.
b) What was done poorly.
c) What was missing that should have been there.

2) In the assignment, the checklist suggested a wide range of possible comments, on

d) First impressions
e) Replication
f) Closing impressions
g) Follow-up tests

The comments did not have to be ordered in any particular way but they should have addressed the issues raised in the assignment checklist in a sensible order. We credited them as follows:

Individual issue discussions are worth up to 3 points, but are normally worth 0.5 or 1 point (typically 1 point if well done). An exceptional discussion that goes to the heart of the quality of the report or suggests what should have been done in a clear and accurate way is worth 2 points. An exceptional and extended (long) discussion that goes to the heart of the quality of the report AND includes follow-up test discussion or suggests (well) what should should have been done is worth 3 points.

3) The primary basis of the evaluation in this section is insight into the quality of the bug report. If the student has mechanically gone through the list of questions, without showing any insight, the max point count is 5. If we see insight, the max point count is 7.

A discussion that shows that the tester did not understand the bug under consideration is worth at most 5, and was often worth less.

Bug number	**Comments at issuzilla**	**Editor's report**	**Total points**
1	**7**	**7**	**14 _ 10**
2	**7**	**7**	**14 _ 10**
3	**7**	**7**	**14 _ 10**
4	**7**	**7**	**14 _ 10**
5	**7**	**7**	**14 _ 10**
GRAND TOTAL			**50**

Assignment 3: Domain Testing & Bug Reporting

1 Create between 10 and 20 domain tests. You can stop at 10 if you find (and write up) 2 bugs. You can stop at 15 tests if you find (and write up) 1 bug.

2 Work in the *Word Processing* part of *OpenOffice*.

3 Pick a function associated with Word Processing. Please run all of your tests on the same function. (If several students are working together, you can pick one function per student.)

4 Pick one (1) input, output, or intermediate result variable

- Identify the variable. Stick with that one variable throughout testing.
- Run a mainstream test (a test that is designed to exercise the function without stressing it). You do tests like this first in order to learn more about the function and the variable's role in that function.

5 Identify risks associated with that variable

6 For each risk, design a test specifically for that risk that is designed to maximize the chance of finding a defect associated with this risk.

7 Explain what makes this a powerful test. It might help you to compare it to a less powerful alternative.

8 What is the expected result for the high-power test?

9 What result did you get?

10 Report your results in a table format that has the following columns:

1 Feature or function
2 Variable name or description
3 Risk
4 Test
5 What makes this test powerful
6 Expected result
7 Obtained result

11 If you find bugs, write up bug reports and enter them into Issuezilla.

12 I strongly recommend that you pair up with someone and have them replicate your bug and evaluate a draft version of your report before you submit it to Issuezilla. I will evaluate your report against a professional standard of quality (essentially, the same evaluation that you just did in Assignment 2).

13 Write a summary report that explains what you believe you now know and don't know about the function, based on your testing. (If your group tested several functions, write up a summary report for each.)

Notes (that I'll use for grading) on Exercise 3

- It's important to answer every section:
  - The table needs 7 columns
  - There should be 10-20 tests and 0-2 bug reports
  - There should be a summary report that explains what you know about the function under test.
- It's important to show the domain analysis (or its results)
  - Use boundary values
  - Identify them as bounds and equivalence classes or identify the different sections of the space as you partition it. You might find it useful to start with a boundary analysis (and table).
  - Be specific about risk
  - Be specific about power (compare to others of the same equivalence class)

Assignment 4 Exploratory Attacks

- This is the fourth of five assignments.
- This is a good assignment for discovering bugs. Remember that if you want bonus points, the bugs MUST be in the database (and I must be notified of it) by December 6 (CSE 4431) or December 8 (SWE 5410). Please feel free to take a bug to a replicator as soon as you have written it up. Enter the bug into IssueZilla when you feel it is good enough to enter (whether you have taken it to a replicator or not). Then send me a note with the bug report or with a pointer to it.
- Conduct at least 4 tests of the *OpenOffice* Word Processing feature, involving one attack from each class of attack:
  - Input constraints
  - Output constraints
  - Storage constraints
  - Computation
- Do NOT do these tests on an embedded spreadsheet in *OpenOffice*.
- For each test,
  - explain why your test is a particularly powerful example of that kind of attack. (That is, explain why this test is better than other, similar tests that you could derive from the same type of attack.)
  - Explain why your attack is a member of the class (input / output / storage / computation) that claim for it.
  - If you find a bug, please report it in the bug tracking database.
- I encourage you to do this assignment with a (one) partner. Creative testing works better in pairs.
- The usual collaboration rules apply--if two of you work together, you should hand in eight tests, two from each category.

Assignment 5 Test Automation Requirements

- Do this question in collaboration with one other student.
- Imagine that you are on the *OpenOffice* testing team as a full-time staff member. You are asked to use a GUI automation tool, such as WinRunner, Silk, or QA Robot, to automate some or all of the testing of *OpenOffice* Word.
- Read "Avoiding Shelfware", "Architectures of Test Automation" and the other papers on test automation on the Blackboard site.
- Consider any ten of the "Twenty-Seven Questions About Requirements" discussed in Avoiding Shelfware. To the best of your ability (sometimes you will make and state reasonable assumptions rather than doing extensive research), answer those ten questions and for each one, explain how that answer would influence your decisions as to what to automate, how to automate, and when in the project to automate it. (Note: you should consider 10 questions whether you work alone or with one other student.)

---

[1] There are plenty of online resources for law students who are learning how to write essay exams, such as

- Martha Peters, "A General Plan for Exams," University of Iowa College of Law Academic Achievement Program, http://www.uiowa.edu/~aap001/examwrite.html, viewed 2/3/03.
- Carolyn Nygren, "Legal Learning for Bar Candidates -- Bar Exam", http://www.findlaw.com/studyskills/3_bar_candidates.html, viewed 2/3/03.
- Gregory Berry, "Rules of Effective Examsmanship for Law Students," School of Law, Howard University, http://www.law.howard.edu/faculty/pages/berry/advice/examtips.htm, viewed 2/3/03.

In addition, many books coach law students on exam preparation and writing skills.

[2] Two examples of University website guides to essay questions are:

- Writing@CSU Writing Guide: "Answering Exam Questions," http://writing.colostate.edu/references/processes/exams, viewed 2/3/03.
- University of Durham Undergraduate Information Site, "Advice on Answering Exam Questions," http://www.dur.ac.uk/biological.sciences/Undergraduate/ugexampage2.htm

These are tip-of-the-iceberg examples. Searches on www.dogpile.com or www.google.com on phrases like "essay exam strategy" and "call of the question" yield thousands of links.

[3] See Cem Kaner, James Bach & Bret Pettichord, *Lessons Learned in Software Testing*, Wiley, 2002, chapter 6.