Florida Institute of Technology

Department of Computer Sciences

Comprehensive Exam (Spring 2007)

SOFTWARE ENGINEERING

Friday, March 16th, 2006; 10:00am – 11:30am

Instructions

- Write the last four digits of your student identification number in the space below.
- This exam consists of 14 pages (including this cover).
- Answer any four (4) of the six (6) questions. Each question has equal value.

  ➔ **Circle the four questions that you want graded:**

  1 2 3 4 5 6

  (If you leave this blank, Questions 1 - 4 will be graded.)
- Use a pen to write your answers in the space provided.
- When a question asks you to "describe," "discuss," or "explain" something, it means you must provide a convincing, clear, and reasonable answer; simply stating a fact without any supporting argument is insufficient.
- No study aids (notes, books, etc.) are permitted during the exam.

Good luck!

ID Number:

For Grading Use Only

	Question	Worth	Grade
1.	Requirements	25	
2.	Design	25	
3.	Construction	25	
4.	Testing	25	
5.	Maintenance & Evolution	25	
6.	Management	25	
	Total	**100**	

1. Requirements (25%)

Requirement specifications are descriptions of a system's functionality, quality attributes, and operational constraints from an engineering perspective. This is in contrast to requirement definitions, which describe the same features of the system but from an end-user point of view. There are a great many requirement specification techniques available, ranging from informal and unstructured (e.g., natural language) to formal and mathematical (e.g., Z). Each one has useful characteristics, but some are more appropriate for a given project than others. Since there is no single technique that is best for all projects, it is important to have a set of criteria to determine, one project at a time, which technique is most suitable.

Problem:

(a) Describe ten (10) criterion for evaluating a requirement specification technique to determine its suitability for a specific project.

(b) Under what circumstances would you select a formal and mathematical specification technique over an informal and unstructured one? Give an example for a modern software system where this would be the case.

Grading: (a) 2% each; (b) 5%.

Note: Use the blank sheet of paper on the next page as needed.

2. Design (25%)

Design is a key component of the overall software lifecycle. Good design contributes to the construction of elegant and bug-free software. During high-level design, control models are created to represent the control flow between sub-systems. To work as a system, sub-systems must be controlled so that their services are delivered to the right place at the right time. Structural design models should not include control information. Instead, the software engineer should organize the sub-systems according to some control model that supplements the structural model used.

Problem:

(a) Describe the *centralized control* design model. Clearly identify the advantages and disadvantages of this approach. Given an example of centralized control for a hypothetical system.

(b) Describe the *event-based control* design model. Clearly identify the advantages and disadvantages of this approach. Given an example of event-based control for a hypothetical system.

(c) Which of these two models would you choose for a batch processing system that takes information about hours worked and pay rates and prints salary slips and bank credit transfer information? Clearly explain why. Be sure to document the assumptions underlying your answer.

Grading: (a) 10%; (b) 10%; (c) 5%.

Note: Use the blank sheet of paper on the next page as needed.

3. Construction (25%)

Linked lists are one of the essential data structures used in software construction.

Problem: Consider a doubly-linked list with alphanumeric data values in its nodes.

(a) Write the C/C++ class/struct(s) that implement(s) such a linked list.

(b) Construct an elegant and efficient **recursive** C/C++ function that takes such a linked list and returns the number of nodes in the list with data values that are lowercase letters.

(c) Redo part (b) but make your C/C++ function **iterative**.

(d) What is the runtime performance of your recursive solution from part (b) in O(*n*) notation? Be sure to explain your answer.

(e) What is the runtime performance of your iterative solution from part (c) in O(*n*) notation? Be sure to explain your answer.

Notes:

- Make sure your solution is constructed clearly and idiomatically, so that it adheres to the commonly accepted definition of good coding style.
- Your solution cannot make use of the Standard Template Library (STL).
- Be sure to properly comment your program; explain how the solution works and why you selected particular algorithm(s) and data structure(s).
- Include defensive programming techniques in your solution where appropriate.
- Use the other side of the paper as needed.

Grading: (a) 5%; (b) 7%; (c) 7%; (d) 3%; (e) 3%.

4. Testing (25%)

There are many types of software testing. For example, functional testing addresses the system's functional requirements. Performance testing addresses the system's non-functional requirements.

Problem:

(a) Describe seven (7) types of performance tests.

(b) Explain why performance tests can be much more difficult to administer than function tests.

Grading: (a) 3% each; (b) 4%.

Note: Use the blank sheet of paper on the next page as needed.

5. Maintenance & Evolution (25%)

Evolution is an important activity of the software lifecycle. In the 1980s, Lehman and Belady examined the growth and evolution of a number of large software systems. Based on their measurements from these studies, they proposed a set of five hypotheses, known as "Lehman's Laws," concerning system change. They claim that these "laws" are invariant and widely applicable.

Problem: Describe all five of Lehman's Laws. Include commentary on the validity of each of these laws as applied to today's software-intensive systems.

Grading: 5% for each description and commentary.

Note: Use the blank sheet of paper on the next page as needed.

6. Management (25%)

Cost & effort estimation is an important part of project management. Numerous models have been created to express the relationship between effort and the factors that influence it. One of the most sophisticated algorithmic estimation techniques is COCOMO II.

Problem:

(a) Describe the COCOMO II estimation technique. Discuss how COCOMO II improves upon the original COCOMO model. Explain the three stages of the COCOMO II model, including model aspects and cost drivers.

(b) Give an example of how COCOMO II could be used by project managers in a modern software system.

Grading: (a) 20%; (b) 5%.

Note: Use the blank sheet of paper on the next page as needed.