Florida Institute of Technology

Department of Computer Sciences

Comprehensive Exam (Spring 2004)

SOFTWARE ENGINEERING

Friday, March 12th, 2004; 10:00am – 11:30am

Instructions

- Write the last four digits of your student identification number in the space below.
- This exam consists of 10 pages (including this cover).
- Answer any four (4) of the following six (6) questions. Each question is of equal value (25%). Circle the questions that you want graded:

  1 2 3 4 5 6

  (If you leave this blank, questions 1 through 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, lucid, 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.	The Software Lifecycle	25	
2.	Software Requirements Analysis	25	
3.	Software Testing	25	
4.	Design & Implementation of Software	25	
5.	Software Project Estimation & Planning	25	
6.	Software Quality Assurance	25	
	Total	**100**	

1. The Software Lifecycle (25%)

A basic software engineering lifecycle model consists of the phases Requirements, Design, Construction (Implementation), Testing, and Maintenance.

Problem: Which of these phases is the most EXPENSIVE phase – justify your answer.

Grading: 5% for identifying correct phase; 20% for justification.

2. Software Requirements Analysis (25%)

Getting software requirements right is notoriously difficult. One of the main problems is getting everyone to agree to the same thing. In other words, developing a common understanding of the problem, from both a user (requirements definition) and an engineering (requirements specification) perspective.

Problem: Describe three techniques for representing requirements specifications. Give an example of two of the three techniques for the same hypothetical system.

Grading: 5% for each clear explanation (15%); 5% for each example (10%).

3. Software Testing (25%)

Software is often made from independent modules of code that have to be integrated together to form a complete application.

Problem: Describe “Top-Down Software Integration and Testing” and “Bottom-Up Software Integration and Testing.” Outline the principals of each approach (including any additional requirements to assist with testing) as well some strengths and weakness of each approach.

Grading: 10% for Top-Down description; 10% for Bottom-Up description; 5% for strengths and weaknesses.

4. Design & Implementation of Software (25%)

Imagine that for some reason that the **`<stack>`** container adapter was not available in the Standard Template Library (STL). You has been tasked with creating a simplified version of **`<stack>`** matching the following C++ interface:

```
template <class Item>
class Stack {

  private:
      // implementation-dependent code

  public:
      Stack(int);
      int empty() const;      // 1 = empty, 0 otherwise
      void push(Item);
      void pop();
      Item top() const;
};
```

<u>Problem</u>: Implement a version of **`<stack>`** matching this interface using native C++ *arrays* as the underlying data structure.

<u>Notes</u>:

- *Your code must not rely on any aspect of the STL.*
- The argument to the **`Stack`** constructor specifies the maximum size of the stack.
- If an error is detected, call the predefined routine **`error(char *)`**. This routine will print the error message to standard error and terminate the program.
- Make sure your solution is constructed clearly and idiomatically, so that it adheres to the commonly accepted definition of good coding style.
- Use the other side of the paper as needed.

<u>Grading</u>: Correctness: 20%; Style: 5%

5. Software Project Estimation & Planning (25%)

Project management and scheduling is an important part of delivering software in a timely manner.

Problem: Describe the following terms: Milestone, Dependencies, Critical path, Slack (or slippage) time. Discuss one potential problem of project scheduling.

Grading: Description of each term: 5% each; Discussion of problem: 5%.

6. Software Quality Assurance (25%)

Of the various software process models that have appeared in the literature, it can be argued that the Software Engineering Institute's "Capability Maturity Model for Software" (SEI SW-CMM®) has had the most impact for large organizations.

Problem: Describe the SW-CMM. Explain each level. Discuss the advantages and disadvantages of this software process improvement model.

Grading: Description (15%); Advantages & Disadvantages (10%).

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