

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page. The exam has 10 questions with equal weight. Some questions have multiple parts. Point totals are given in the margin.

Student Id: _____ **Answer Key** _____

1. _____ **10**
2. _____ **10**
3. _____ **10**
4. _____ **10**
5. _____ **10**
6. _____ **10**
7. _____ **10**
8. _____ **10**
9. _____ **10**
10. _____ **10**
- Total _____ **100**

No notes or books may be used on the exam. If you have any questions, please raise your hand and I will try to answer them.

1. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

- 2 (a) The majority of errors in software can generally be found during:
- i. requirements reviews
 - ii. design reviews
 - iii. code reviews
 - iv. integration testing
 - v. acceptance testing

Answer: (iii)

- 2 (b) Which of the following make testing object-oriented systems easier?
- i. code reuse
 - ii. encapsulation
 - iii. small methods
 - iv. (ii) and (iii)
 - v. all of the above

Answer: (v)

- 2 (c) Which of the following make testing object-oriented systems harder?
- i. encapsulation
 - ii. polymorphism
 - iii. modularity
 - iv. (i) and (ii)
 - v. all of the above

Answer: (iv)

- 2 (d) Nonfunctional requirements are validated during:
- i. performance testing
 - ii. function testing
 - iii. acceptance testing
 - iv. (i) and (ii)
 - v. all of the above

Answer: (i)

- 2 (e) Acceptance tests are written by:
- i. the customer
 - ii. test engineers
 - iii. developers
 - iv. (i) and (ii)
 - v. none of the above

Answer: (i)

2. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

- 2 (a) A big-bang integration strategy
- i. is appropriate only for small systems
 - ii. is easy to coordinate
 - iii. produces a basic working system early in the testing process
 - iv. (i) and (ii)
 - v. all of the above

Answer: (iv)

- 2 (b) A sandwich integration strategy:
- i. always requires the same number of steps as big-bang integration
 - ii. favors testing of top-level components
 - iii. is easy to coordinate
 - iv. (i) and (ii)
 - v. none of the above

Answer: (v)

- 2 (c) A bottom-up integration strategy:
- i. produces a working system quickly
 - ii. favors testing of lower-level components
 - iii. works well for layered systems
 - iv. (ii) and (iii)
 - v. all of the above

Answer: (iv)

- 2 (d) Which of the following integration strategies require stubs?
- i. bottom-up
 - ii. top-down
 - iii. big-bang
 - iv. (ii) and (iii)
 - v. (i) and (iii)

Answer: (iv)

- 2 (e) Which of the following integration strategies require drivers?
- i. sandwich
 - ii. big-bang
 - iii. modified top-down
 - iv. all of the above
 - v. none of the above

Answer: (iv)

3. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

- 2 (a) Which of the following is *not* a type of maintenance?
- perfective
 - adaptive
 - defective
 - preventive
 - none of the above

Answer: (iii)

- 2 (b) The majority of maintenance effort is spent:
- fixing bugs
 - adding enhancements
 - redesigning software
 - accommodating new platforms and processors
 - none of the above

Answer: (ii)

- 2 (c) We can lower the costs of maintenance by:
- performing preventive maintenance
 - using an evolutionary process model
 - spending more time in requirements gathering
 - (i) and (iii)
 - all of the above

Answer: (v)

- 2 (d) Which of the following is *not* a stage of reengineering?
- reverse engineering
 - forward engineering
 - code restructuring
 - (ii) and (iii)
 - none of the above

Answer: (v)

- 2 (e) Reengineering a piece of legacy software is a good idea if:
- new applications are now available
 - updated documentation is needed
 - the software has many errors
 - (i) and (ii)
 - all of the above

Answer: (v)

4. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

2 (a) Which of the following is *not* a quality failure cost?

- i. testing
- ii. warranty
- iii. help desk
- iv. maintenance
- v. none of the above

Answer: (i)

2 (b) Which of the following are quality appraisal costs?

- i. testing
- ii. code inspections
- iii. training
- iv. (i) and (ii)
- v. all of the above

Answer: (iv)

2 (c) Which of the following affect quality of product operation?

- i. correctness
- ii. reliability
- iii. reusability
- iv. (i) and (ii)
- v. all of the above

Answer: (iv)

2 (d) Which of the following affect quality of product revision?

- i. portability
- ii. usability
- iii. interoperability
- iv. (i) and (ii)
- v. none of the above

Answer: (v)

2 (e) Which of the following affect quality of product transition?

- i. integrity
- ii. testability
- iii. maintainability
- iv. (i) and (ii)
- v. none of the above

Answer: (v)

5. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

- 2 (a) Which of the following are evolutionary process models?
- i. prototyping
 - ii. spiral
 - iii. extreme programming
 - iv. (i) and (iii)
 - v. none of the above

Answer: ERROR: should have been (ii) and (iii)

- 2 (b) Extreme programming:
- i. has people program in pairs
 - ii. has high communication with the customer
 - iii. is appropriate for large teams
 - iv. (i) and (ii)
 - v. all of the above

Answer: (iv)

- 2 (c) The “V” process model:
- i. illustrates the importance of testing
 - ii. includes more testing than other models
 - iii. eliminates testing completely
 - iv. uses black box testing exclusively
 - v. none of the above

Answer: (i)

- 2 (d) Which choice has the models in order of increasing need for customer communication?
- i. waterfall, extreme programming, spiral
 - ii. waterfall, spiral, extreme programming
 - iii. prototyping, waterfall, spiral
 - iv. incremental, extreme programming, spiral
 - v. none of the above

Answer: (ii)

- 2 (e) The most important phase of the spiral model is:
- i. development
 - ii. customer evaluation
 - iii. risk analysis
 - iv. engineering
 - v. none of the above

Answer: (iii)

6. *Note:* Each incorrect answer results in a penalty of $\frac{1}{2}$ point:

- 2 (a) Intuitive people are preferred over rational people for:
- i. requirements
 - ii. design
 - iii. implementation
 - iv. (i) and (ii)
 - v. (ii) and (iii)

Answer: (iv)

- 2 (b) Which of the following is *not* a formal, impersonal method of communication?
- i. source code
 - ii. technical notes
 - iii. code inspections
 - iv. (i) and (ii)
 - v. none of the above

Answer: (iii)

- 2 (c) Loosely structured teams:
- i. require a lot of communication
 - ii. are good for generating new ideas
 - iii. are best suited to small teams
 - iv. (i) and (ii)
 - v. all of the above

Answer: (v)

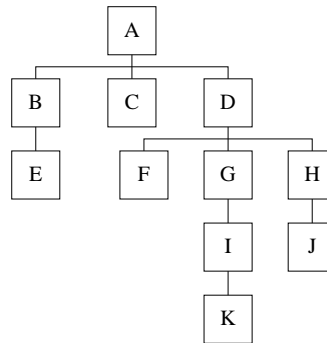
- 2 (d) The best conceptual integrity can be achieved using:
- i. a democratic decentralized team
 - ii. a control centralized team
 - iii. a control decentralized team
 - iv. a loosely structured team
 - v. none of the above

Answer: (ii)

- 2 (e) Which of the following statements are *not* true?
- i. A democratic decentralized team cannot work for large projects.
 - ii. A controlled centralized team has only vertical communication.
 - iii. A controlled decentralized team has only horizontal communication.
 - iv. (i) and (iii)
 - v. all of the above

Answer: (iv)

7. Consider a system with the following modular structure:



Using a big-bang integration strategy, the following modules would be tested at each step:

1. A, B, C, D, E, F, G, H, I, J, K
2. (A..K)

Show the steps and modules tested using the following integration strategies:

4

(a) bottom-up

Answer:

1. K
2. (I,K), J
3. E, F, (G, I, K), (H, J)
4. (B, E), C, (D, F, G, H, I, J, K)
5. (A..K)

6

(b) modified top-down

Answer:

1. A
2. B, C, D
3. (A..D)
4. E, F, G, H
5. (A..H)
6. I, J
7. (A..J)
8. K
9. (A..K)

8. Answer the following questions related to *The Mythical Man Month*:

5

- (a) What is the second system effect that Brooks describes? How would *you* try to avoid this if you were a software developer or manager?

Answer: Essentially, Brooks argues that if your first system is successful, then you have a tendency to try to do too much in your second system. You unknowingly add too many features ("feature creep"), use too little personnel, etc. As a result, your second system will fail.

You could try to avoid the second system effect by using an evolutionary process model, particularly an incremental model, which forces you to deliver the system in smaller pieces. If you fall behind schedule, you could always eliminate some features. You might also decide to use a highly structured team to ensure that deadlines are met. Or, you might use object-oriented technologies in an attempt to reuse code and encourage modularity.

5

- (b) What does Brooks mean by the "accidents" and "essence" of software engineering? Where have we been spending the majority of our time and effort and where should be spending it?

Answer: An "accident" is something that is not essential or inherent to software engineering, whereas an "essence" is something that is inherent to software engineering. For example, programming languages and editors are accidents, and failing to capture requirements is an essence. In the past, we have spent the majority of our time on accidents (e.g., inventing new programming languages). However, Brooks argues that 90% of all effort is really spent dealing with the essence, and that therefore we need to spend more of our time there if we hope to improve the software engineering discipline.

- 5 9. (a) A few years ago, Sun Microsystems decided to develop and market StarOffice, a set of desktop tools that would be comparable to Microsoft's Office suite of tools but would be targeted for UNIX rather than Windows. At that time, no other major UNIX vendor had developed or was planning to develop such a product.

What process model would you use? What team organization would you use? Justify your answers by briefly explaining not only your choice but also why you rejected another choice.

Answer: Given the well-defined requirements and lack of constraints, the spiral model would be the best choice. You could argue for the waterfall model, although the project is very large and an evolutionary model is probably needed. The incremental model is not needed since there is plenty of time, personnel, and money for the project.

A controlled team organization would be appropriate. The project is quite large and does not require innovation, since Microsoft has already done most of that, so a democratic team organization is not suitable. In contrast, a controlled team organization is probably necessary given the scale of the project.

- 5 (b) You are currently working for a networking company that is working on optical networks, which is the hottest, fastest growing area. You are a software engineering hired to write the software to configure and test the hardware (i.e., set up and configure the new optical routers). The software must ship with the hardware, otherwise the hardware is useless. Unfortunately, the hardware is constantly being changed and will not be finalized until a few weeks before shipping.

What process model would you use? What team organization would you use? Justify your answers by briefly explaining not only your choice but also why you rejected another choice.

Answer: The incremental model is the definite choice here since you need to produce a working product in a very short amount of time. The basic features could be included in the first release and more advanced features added in later releases. Prototyping is not appropriate because you need to construct a real deliverable. The waterfall model is not appropriate as well given the constantly changing environment.

Given the strict deadline, a controlled team organization is needed. A controlled-decentralized organization is probably best, since the technology is new and therefore problem solutions could require some originality. A democratic team organization is definitely not a good idea if you want to ship the product on time.

- 5 10. (a) Which is more important: the product or the process used to build the product? Briefly argue both cases.

Answer: The product is more important than the process since the product is ultimately what is delivered to the customer. The customer pays you for the product and, for the most part, does not care how you go about building it.

However, certain process models are better for certain products given completeness of requirements, time constraints, etc. Therefore, the process might be viewed as being as important as the product since the right process can help ensure that a robust, correct product is delivered on time and within budget.

- 5 (b) If a product passes the test suites, is it of high quality? Briefly justify your answer by explaining what quality is and the relationship between testing and quality of software.

Answer: If the product passes the test suites, it is not necessarily of high quality. Quality encompasses many things, only one of which is correctness, which testing only tries to ensure (but generally cannot guarantee). Quality software should also be maintainable, portable, reusable, and flexible. Most of these properties testing cannot ensure. If a product fails the test suites, then it certainly is not of high quality regardless of how reusable or maintainable it is. If a product fails the test suites, it certainly is not correct, which is arguably the most important of all product characteristics.