

1. True\_\_False\_\_ (Brooks Chap 9): Representation (i.e., data and its structure) is the essence of programming!
2. True\_\_False\_\_ (Brooks Chap 13): In good top-down design (TDD), clarity of structure and representation makes the precise statement of requirements and functions of the modules more difficult.
3. True\_\_False\_\_ (Brooks Chap 13): In good TDD, suppression of detail makes flaws in the structure more difficult to detect and locate.
4. True\_\_False\_\_ (Brooks Chap 13): In good TDD, if the design can be tested at each level of refinement (i.e., steps), then testing can start earlier and focus on the lowest level of detail before the step is complete.
5. True\_\_False\_\_ (Brooks Chap 13): In good TDD, partitioning and independence of modules avoids system bugs.
6. True\_\_False\_\_ (Brooks Chap 14): In reducing role conflicts, when the manager knows his boss will accept status reports without panic or preemption, he comes to give honest appraisal.
7. True\_\_False\_\_ (Brooks Chap 14): Brooks states that schedule slippage in a project is a disaster due to termites [not tornadoes]. Yet day-by-day slippage is easy to recognize, prevent and makeup.
8. True\_\_False\_\_ (Brooks Chap 14): Brooks also states that estimates never really change significantly as the start time draws near no matter how wrong they turn out to be.
9. True\_\_False\_\_ (Brooks Chap 14): Schedule milestones should be flexible, and define points in time that may not necessarily correspond to measurable events.
10. True\_\_False\_\_ (Brooks Chap 14): Critical path schedules tell which slips matter, show who waits for what, and are most valuable to their users while in preparation.
11. True\_\_False\_\_ (Brooks Chap 14): Every boss needs to know 2 kinds of information a) exceptions to plan that require action, and b) status picture for education.
12. True\_\_False\_\_ (Brooks Chap 15): In a useful prose description of a program, the following elements are important: a) purpose, b) environment, c) domain and range, d) functions realized and algorithms used, and e) I/O formats, as well as others.

13. True\_\_False\_\_ (Brooks Chap 15): In a useful prose description of a program, the following elements are also important: a) operating instructions, b) options, c) exception handling, d) running time, and e) accuracy and checking, as well as others.
14. True\_\_False\_\_ (Brooks Chap 15): There are two faces of a computer program a) the face of the programmer and b) the face of the user.
15. True\_\_False\_\_ (Brooks Chap 15): 3 levels of documentation are required for 3 different types of software users as follows a) Casual user, b) Military user and c) Commercial user.
16. True\_\_False\_\_ (Brooks Chap 15): Self documenting programs are created while the code is being composed and minimizes extra work.
17. True\_\_False\_\_ (Brooks Chap 16): Brooks describes two definitions of AI as (1) The use of computers to solve problems that previously could only be solved by applying human intelligence, and (2) The use of a specific set of programming techniques known as heuristic or rule based programming.
18. True\_\_False\_\_ (Summerville): The principle value of using formal specification techniques in the software process is that it forces an analysis of the system requirements at an early stage. Correcting errors at this stage is cheaper than modifying a delivered system.
19. True\_\_False\_\_ (Brooks Chap 16): One of the most touted recent developments is the programming language Ada, a special-purpose, high-level language of the 1980s. Ada reflects evolutionary improvements in language concepts and embodies features to encourage modern design and modularization concepts.
20. True\_\_False\_\_ (Brooks Chap 17): Harel offers a silver bullet called the Vanilla Framework and Brooks argues that software structure is embedded in 3-space, so there is at least one natural mapping from a conceptual design to a 3-dimensional diagram.
21. True\_\_False\_\_ (Brooks Chap 17): NSB asserts and argues that software engineering developments will produce an order-of-magnitude improvement in programming productivity within ten years.
22. True\_\_False\_\_ (Brooks Chap 18): All repairs (corrective maintenance) tend to destroy structure, to increase entropy and disorder of the system.
23. True\_\_False\_\_ (Brooks Chap 18): One of Brooks propositions is that more programming projects have gone awry for lack of calendar time than for all other causes.

- 24. True\_\_False\_\_ (Brooks Chap 19):Brooks claims that software robustness and software productivity have been hurt by the *buy and build shrink-wrapped packages as components* trends of recent years.
  
- 25. True\_\_False\_\_ (Brooks Chap 19):WIMP stands for Why Implement Mega Programs?
  
- 26. True\_\_False\_\_ (Brooks Chap 19): Brooks claims that Apple capitalized on the notion of incremental transition from a novice to a power user.
  
- 27. True\_\_False\_\_ (Brooks Chap 19): One of the central issues (concerns) of software engineering today is how to maintain intellectual control over complexity in large doses.

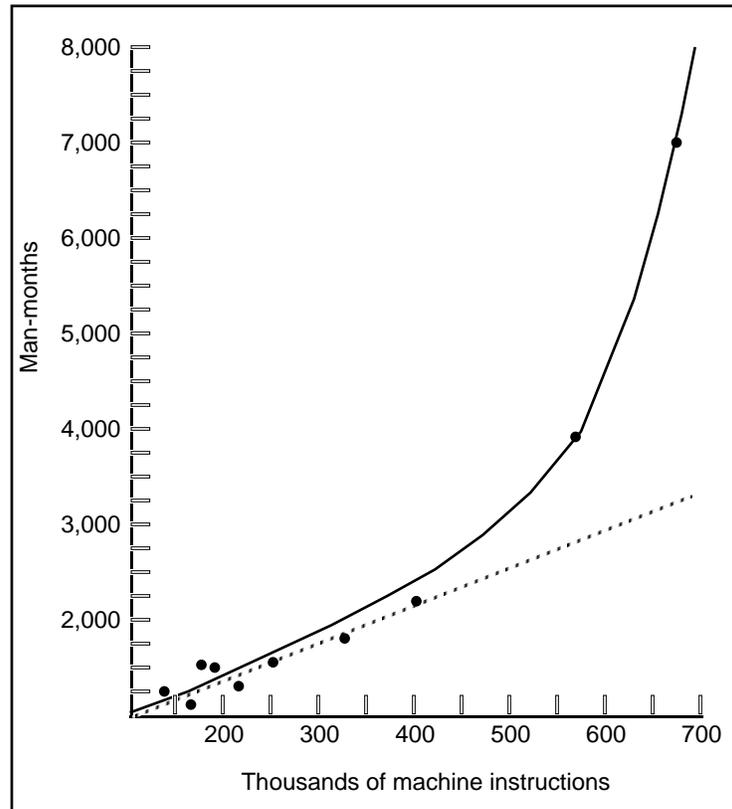
---

These questions are important but are not necessary to prepare for Quiz 2. They may be useful in preparing for the exam:

- 28. (Brooks Chap 4): Conceptual integrity does require that a system reflect a single philosophy and that the specification as seen by the user flow from a few minds. Because of the real division of labor into \_\_\_\_\_, \_\_\_\_\_, and realization, however, this does not imply that a system so designed will take longer to build. Experience shows the opposite, that the integral system goes together \_\_\_\_\_ and takes \_\_\_\_\_ time to test.
  
- 29. (Brooks Chap 3 and 4): In these articles the author comments on the notion of conceptual integrity. How is conceptual integrity achieved? There are three basic ideas (BE BRIEF).
  - (a) \_\_\_\_\_  
\_\_\_\_\_.
  - (b) \_\_\_\_\_  
\_\_\_\_\_.
  - (c) \_\_\_\_\_  
\_\_\_\_\_.
  
- 30. (Brooks Chap 6): The manual is the external \_\_\_\_\_ of the product. (Hint, the style must be precise, full, and accurately detailed.)



34. (Brooks Chap 8): The following graph (Figure 8.1) is related to an equation that is used to determine the effort needed to complete a given software project. This graph show the general relationship described by the data of Portman, Aron, Harr, and OS/360.



(Fig. 8.1 in Brooks) Programming effort as a function of program size.

The following questions are generally based on this relationship.

a. Figure 8.1 shows the equation:

$$\text{Effort} = (\text{constant}) * (\text{_____})^{1.5} \text{ \{fill in the blank\}.}$$

b. The 4 data sets which are discussed (i.e., presented by Portman, Aron, Harr, OS/360) all confirm striking differences in \_\_\_\_\_ related to the complexity and difficulty of the task itself.

c. Corbato of MIT's Project MAC reported a mean productivity of 1200 lines of debugged PL/1 statements per man-year on the MULTICS system. That data seem to be comparable in terms of kind of effort included. However, Corbato's numbers are in *lines* per man-year, not *words*! Each statement in his system corresponds to about 3-5 words of handwritten code! This suggests two conclusions:

- Programming seems \_\_\_\_\_ in terms of elementary statements, a conclusion that is reasonable in terms of the thought a statement requires and the errors it may include.
- Programming productivity may be increased as much as five times when a suitable \_\_\_\_\_ language is used.

35. (Brooks Chap 9): Since size is such a large part of the user cost of a programming system product, the builder must set size targets, \_\_\_\_\_ size, and devise size \_\_\_\_\_ techniques, just as the hardware builder sets component count targets, \_\_\_\_\_ component

count, and devises count \_\_\_\_\_ techniques. Brooks continues to relate a OS/360 experience and concludes the following moral: Set total size budgets as well as \_\_\_\_\_ - \_\_\_\_\_ budgets; set budgets on backing-store accesses as well as on size. The second moral is subsequently described which calls for defining exactly what a module \_\_\_\_\_ (two words) when you specify how big it is.

36. (Brooks Chap 15), Test cases fall into 3 parts of the input data domain. *The first type are the mainline cases that exercise chief functions.* The other 2 types of test cases, by their nature, are *barely Legitimate.* Name & describe the other two types in terms of their purpose (or intent).

(Second Type) \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

(Third Type) \_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

37. (Brooks Chap 16): What is the silver bullet that Brooks is talking about in the article "No Silver Bullet: Essence and Accidents of Software Engineering."

\_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

38. (Brooks Chap 16): Based on the article "No Silver Bullet: Essence and Accidents of Software Engineering," describe the meaning of the following items:

(a) Expert Systems \_\_\_\_\_

(b) Automatic Programming \_\_\_\_\_  
 \_\_\_\_\_

(c) Graphical Programming \_\_\_\_\_  
 \_\_\_\_\_

(d) Program Verification \_\_\_\_\_  
 \_\_\_\_\_

38. (Brooks Chap 11): The fundamental problem with program maintenance is that fixing the defect has a substantial (20-50%) chance of introducing another. So the whole process is two steps forward and one step back. How can we begin to avoid such side affects?

---

---

---

---

39. (Brooks Chap 11): Systems program building is entropy-decreasing process, hence metastable. What is program maintenance in contrast to this thesis?

---

---

---

---

40. (Brooks Chap 20): What are the three main questions that Brooks addresses here that have been given to him by his readers 20 years after?

---

---

---

---

41. (Brooks Chap 20): What example does Brooks give about enforcing architecture to provide conceptual integrity across an application?

---

---

---

---

42. (Brooks Chap 20): What is the problem with Microsoft's "Build Every Night" approach?

---

---

---

---

43. (Brooks Chap 20): What are the distinctive concerns of software engineering today (as set forth in Chapter 1) but clearly enumerated in Chapter 20?

---

---

---

---

44. How true is Brooks law? \_\_\_\_\_.