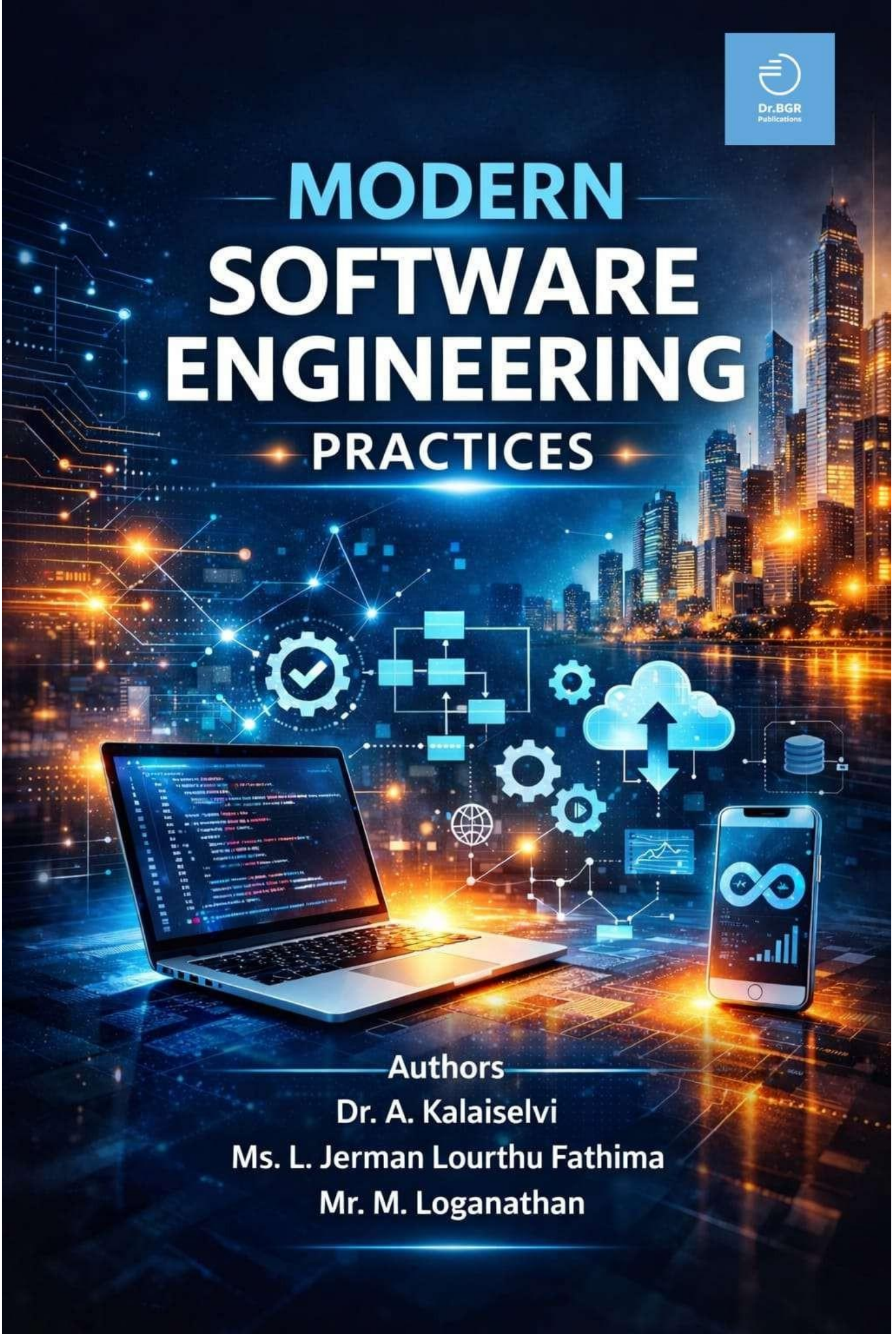
Dr.BGR
Publications
MODERN
SOFTWARE
ENGINEERING
PRACTICES
Authors
Dr. A. Kalaiselvi
Ms. L. Jerman Lourthu Fathima
Mr. M. Loganathan

Dr.BGR Publications

Modern Software Engineering Practices

Dr. A. Kalaiselvi

Assistant Professor

Department of Computer Science and Applications

Arul Anandar College (Autonomous), Karumathur

Ms. L. Jerman Lourthu Fathima

Assistant Professor

Department of Computer Science and Applications

Arul Anandar College (Autonomous), Karumathur

Mr. M. Loganathan

Assistant Professor

Department of Computer Science and Applications

Arul Anandar College (Autonomous), Karumathur

2026

Verso Page

Publisher	Dr. BGR Publications
	India | Tamil Nadu | Tuticorin
	9003494749
	drbgrpublications@gmail.com
	https://drbgrpublications.in/books/
	https://www.instagram.com/drbgrpublications/
Title	Modern Software Engineering Practices
ISBN	978-81-999642-9-7
Book Type	Authored Academic Book
Author	Dr. A. Kalaiselvi
Co-Authors	Ms. L. Jerman Lourthu Fathima
	Mr. M. Loganathan
Acknowledgment	Arul Anandar College, Karumathur, Madurai
Language	English
Product Form	Digital download and online
Date of Publication	25-03-2026
Copyright Holder	© 2026 by respective author
Edited and typeset by	Dr. A. Kalaiselvi
Cover design credit	Dr. A. Kalaiselvi
Title Page by	Dr. B.Govindarajan
Digital Production Line	This book is published in digital format and made available globally through open access platforms.
Disclaimer	The author is fully responsible for the content of this book. The publisher disclaims all liability for errors, omissions, inaccuracies, plagiarism, or interpretations. Unintentional errors may be reported to the author or publisher for correction in future editions.
Copyright Notice	All rights reserved. Reproduction or transmission in any form requires prior written permission. Mail: kalaiselvi@aactni.edu.in
Jurisdiction Clause	Any disputes arising from this publication shall be the sole responsibility of the author(s). The publisher shall not be held liable for any legal claims, disputes, or consequences related to the content of this book.
Barcode	

INDEX

ISBN: 978-81-999642-9-7

ABOUT THE BOOK

Modern Software Engineering Practices (MCQ) is a comprehensive academic resource designed to strengthen students' understanding of fundamental software engineering concepts through a structured multiple-choice question format. The book systematically covers the complete **Software Development Life Cycle (SDLC)**, beginning with an introduction to software engineering principles and progressing through project planning, cost estimation, and requirement specification.

It provides detailed coverage of software design methodologies, implementation strategies, quality assurance practices, and systematic software testing approaches. The final sections focus on software maintenance, highlighting the importance of continuous improvement and long-term system sustainability.

This book is specially designed for **undergraduate students**, **competitive examination aspirants**, and learners preparing for **technical interviews**. By combining theoretical foundations with objective-type questions, the book enables readers to evaluate their knowledge, strengthen conceptual clarity, and prepare effectively for **university examinations and professional certifications**.

ISBN: 978-81-999642-9-7

PREFACE

Software engineering has evolved significantly in response to the increasing complexity of modern software systems. With rapid technological advancements, the demand for reliable, scalable, and high-quality software solutions continues to grow. ***Modern Software Engineering Practices (MCQ)*** is designed to provide learners with a clear and structured understanding of fundamental and contemporary software engineering concepts through an objective and examination-oriented approach.

This book systematically covers the essential stages of the **Software Development Life Cycle (SDLC)**, beginning with an introduction to software engineering and progressing through software project planning, cost estimation, and requirement specification. It further explores key areas such as software design methodologies, implementation strategies, quality assurance practices, testing methodologies, and software maintenance. Each section is supported by carefully prepared multiple-choice questions to reinforce conceptual clarity and facilitate effective self-assessment.

The primary objective of this book is to assist undergraduate students, competitive examination aspirants, and beginners in mastering software engineering principles in a concise and practical manner. By integrating theoretical concepts with MCQ-based evaluation, this work aims to enhance analytical thinking, examination readiness, and professional competence.

We also express our sincere gratitude to **BGR Publications** for providing a valuable platform to publish this scholarly contribution. We hope that this book will serve as a useful academic resource and contribute meaningfully to the learning journey of students pursuing software engineering and related disciplines.

ISBN: 978-81-999642-9-7

ACKNOWLEDGEMENT

The authors express their sincere gratitude to all those who have contributed to the successful completion of ***Modern Software Engineering Practices (MCQ)***. This book is the result of dedicated academic effort, thoughtful preparation, and a shared commitment to enhancing learning in the field of software engineering. We extend our heartfelt thanks to the faculty members and academic mentors whose guidance, insights, and continuous encouragement helped shape the structure and content of this book. Their expertise ensured that the concepts presented are accurate, relevant, and aligned with current academic standards.

We also acknowledge the support of the Head of the Department ***Mr. T. Manoj Prabaharan*** and Institution for providing the necessary academic environment and resources that facilitated the successful development of this work. Special appreciation is extended to the publishers for offering a professional platform to bring this academic resource to students and educators.

Our special appreciation is also extended ***to Dr. BGR Publications*** for offering a professional and esteemed platform to publish this academic contribution, and for their assistance and support during the publication process.

We hope this book serves as a meaningful contribution to learners preparing for university examinations, competitive tests, and professional growth in software engineering.

ISBN: 978-81-999642-9-7

Introduction

Software Engineering has become one of the most essential disciplines in modern computing and information technology. With the rapid growth of software systems across industries such as healthcare, finance, education, and communication, the demand for well-structured, reliable, and efficient software development practices has increased significantly. Software Engineering provides the principles, methodologies, and tools required to develop high-quality software systems that meet user requirements while maintaining reliability, efficiency, and maintainability.

This book has been designed to support undergraduate students in understanding the fundamental concepts of Software Engineering through **Multiple Choice Questions (MCQs)**. MCQs play an important role in reinforcing conceptual clarity, improving analytical thinking, and preparing students for examinations, competitive tests, and professional assessments.

The content of this book covers the core areas of Software Engineering, beginning with an **Introduction to Software Engineering**, which explains the importance, evolution, and basic principles of the discipline. It then moves into **Software Project Planning**, focusing on planning activities, project management concepts, and resource allocation required for successful software development.

Another critical topic addressed in this book is **Software Cost Estimation**, where students are introduced to different techniques used to estimate project effort, cost, and time. Accurate estimation is vital for project success and helps organizations manage resources effectively.

The book also explores **Requirement Specification**, which emphasizes the importance of clearly identifying and documenting user requirements. A well-defined requirement specification ensures that the final software product meets user expectations and business goals.

Following this, the section on **Software Design** introduces students to architectural design, design principles, and structured approaches that guide the development of scalable and maintainable software systems. This is complemented by **Software Implementation**, which focuses on translating design into actual working software using appropriate coding standards and development practices.

To ensure the quality and reliability of software products, the book includes comprehensive coverage of **Software Quality Assurance, Software Testing, and Software Maintenance**. These areas highlight the processes used to verify and validate software, detect defects, and maintain software systems throughout their lifecycle.

Each section in this book includes carefully prepared multiple-choice questions that aim to help students evaluate their understanding of the subject. Additionally, **answer keys** are provided to allow learners to verify their responses and reinforce their knowledge.

Overall, this book aims to serve as a **useful learning and revision resource** for undergraduate students studying Software Engineering. It is also intended to support instructors in assessing students' understanding of key concepts in a structured and efficient manner.

INTRODUCTION TO SOFTWARE ENGINEERING

1. Software is best defined as _____.
   a) Only computer programs
   b) Hardware instructions
   c) Programs, documentation, and operating procedures
   d) Data files only

2. Software Engineering is the application of _____.
   a) Programming skills only
   b) Hardware design techniques
   c) Engineering principles to software development
   d) Computer networking concepts

3. The term *software crisis* refers to _____.
   a) Lack of computers
   b) Problems in managing large software systems
   c) Hardware failures
   d) Internet failure

4. Which organization provided the standard definition of Software Engineering?
   a) ISO
   b) IEEE
   c) ACM
   d) ANSI

5. Software Engineering focuses on _____.
   a) Writing code quickly
   b) Systematic and disciplined development
   c) Hardware maintenance
   d) Network design

6. A program differs from software because software includes _____.
   a) Only source code
   b) Only algorithms
   c) Documentation and procedures
   d) Only users

7. Programs are generally developed for _____.
   a) Widespread use
   b) A specific task
   c) Commercial sale
   d) Multiple platforms

8. Software is expected to be _____.
   a) Complex
   b) User-friendly
   c) Author-dependent
   d) Hardware-dependent

9. Software size is commonly measured using _____.
   a) CPU speed
   b) LOC
   c) RAM
   d) Bandwidth

10. LOC stands for _____.
    a) Level of Complexity
    b) Lines of Code
    c) Language-oriented Commands
    d) Logical Operations Count

11. Size-oriented metrics relate quality or productivity to _____.
    a) Cost
    b) Time
    c) Software size
    d) Hardware

12. Size estimation is important because it helps in _____.
    a) Debugging
    b) Cost and effort estimation
    c) Hardware design
    d) Data entry

13. Software quality refers to _____.
    a) Speed of execution
    b) Meeting user requirements
    c) Number of programmers
    d) Cost of hardware

14. Correctness of software means _____.
    a) Ease of use
    b) Security
    c) Performing required functions
    d) Portability

15. Maintainability refers to _____.
    a) Speed of execution
    b) Ease of modification
    c) Size of software
    d) Cost of development

16. Reliability is concerned with_____.
    a) Ease of learning
    b) Failure-free operation
    c) User interface
    d) Code length

17. Integrity of software is related to _____.
    a) Performance
    b) Security against attacks
    c) Size
    d) Documentation

18. Software productivity is measured as _____.
    a) Lines of code per programmer
    b) Output per unit effort
    c) Cost per module
    d) Time per test case

19. Higher productivity means _____.
    a) More programmers
    b) More documentation
    c) Less effort for more output
    d) More hardware

20. Productivity mainly depends on _____.
    a) Programming language only
    b) Tools, methods, and people
    c) Hardware speed
    d) Internet bandwidth

21. Software project planning begins with _____.
    a) Coding
    b) Testing
    c) Defining project scope
    d) Maintenance

22. Project planning mainly helps in _____.
    a) Writing code
    b) Estimating cost and schedule
    c) Debugging
    d) Installing software

23. Project scope defines _____.
    a) Coding style
    b) Functions, performance, and constraints
    c) Programming language
    d) Hardware type

24. Software project planning is carried out _____.
    a) After testing
    b) At the end of the project
    c) At the beginning of the project
    d) During maintenance

25. A software development process is _____.
    a) Random activities
    b) A structured set of activities
    c) Only coding
    d) Only testing

26. The main objective of process planning is to _____.
    a) Increase code size
    b) Define workflow and responsibilities
    c) Reduce documentation
    d) Avoid testing

27. A software process model describes _____.
    a) Hardware architecture
    b) Development activities and sequence
    c) Programming syntax
    d) Network topology

28. Organizational structure defines _____.
    a) Coding rules
    b) Team roles and responsibilities
    c) Testing tools
    d) Software size

29. A good organizational structure helps in _____.
    a) Increasing confusion
    b) Reducing communication
    c) Effective coordination
    d) Eliminating documentation

30. The project manager is mainly responsible for _____.
    a) Writing code
    b) Managing resources and schedule
    c) Testing modules
    d) Designing algorithms

31. The primary goal of software engineering is to produce _____.
    a) Large software
    b) Low-cost software
    c) High-quality software
    d) Fast software

32. Software engineering uses _____.
    a) Informal methods
    b) Random techniques
    c) Structured and disciplined methods
    d) Trial-and-error approach

33. Software documentation is required for _____.
    a) Increasing execution speed
    b) Maintenance and understanding
    c) Hardware compatibility
    d) Data storage

34. Software differs from hardware because software _____.
    a) Wears out
    b) Is intangible
    c) Is physical
    d) Needs electricity

35. The success of a software project mainly depends on_____.
    a) Hardware speed
    b) Good planning and management
    c) Number of computers
    d) Internet connectivity

36. Software Engineering primarily deals with _____.
    a) Writing error-free code
    b) Managing hardware resources
    c) Developing and maintaining software systems
    d) Network security

37. The need for Software Engineering arose mainly due to _____.
    a) Increase in computer speed
    b) Growth in software size and complexity
    c) Reduction in hardware cost
    d) Internet usage

38. Which of the following is NOT a characteristic of software?
    a) It is intangible
    b) It does not wear out
    c) It is manufactured
    d) It can be modified

39. Software engineering emphasizes _____.
    a) Individual programming
    b) Ad-hoc development
    c) Team-based development
    d) Trial-and-error approach

40. The primary output of a software project is _____.
    a) Hardware
    b) Source code only
    c) Software product with documentation
    d) Data files

41. Software crisis mainly refers to _____.
    a) Lack of programmers
    b) Delay and cost overrun of projects
    c) Hardware failures
    d) Virus attacks

42. One major cause of software crisis is _____.
    a) Poor communication
    b) Lack of systematic development approach
    c) Use of high-level languages
    d) Excess memory

43. Software crisis led to the development of _____.
    a) Operating systems
    b) Software Engineering discipline
    c) Computer networks
    d) Artificial intelligence

44. Software size estimation is required during _____.
    a) Maintenance phase
    b) Planning phase
    c) Testing phase
    d) Installation phase

45. Size metrics help in estimating _____.
    a) Hardware cost
    b) Project effort and duration
    c) Network speed
    d) Memory size

46. Which is a disadvantage of LOC?
    a) Easy to calculate
    b) Depends on programming language
    c) Helps cost estimation
    d) Widely used

47. Software quality factors ensure _____.
    a) Fast execution only
    b) Customer satisfaction
    c) More lines of code
    d) Less documentation

48. Usability refers to _____.
    a) Ease of modification
    b) Ease of learning and use
    c) Failure-free operation
    d) Security

49. Portability of software means _____.
    a) Easy installation
    b) Ability to run on different platforms
    c) Easy debugging
    d) Fast execution

50. Which quality factor deals with system efficiency?
    a) Reliability
    b) Performance
    c) Integrity
    d) Usability

51. Software productivity improves with _____.
    a) Poor planning
    b) Skilled manpower
    c) More errors
    d) More documentation only

52. CASE tools help improve productivity by _____.
    a) Increasing hardware usage
    b) Automating development activities
    c) Reducing documentation
    d) Eliminating testing

53. Productivity is negatively affected by _____.
    a) Clear requirements
    b) Skilled developers
    c) Frequent requirement changes
    d) Good tools

54. The first step in project planning is _____.
    a) Coding
    b) Testing
    c) Feasibility study
    d) Maintenance

55. Project planning helps to _____.
    a) Increase confusion
    b) Reduce project risks
    c) Eliminate documentation
    d) Avoid estimation

56. Software cost estimation is mainly used to _____.
    a) Reduce code size
    b) Allocate resources
    c) Improve speed
    d) Design hardware

57. A development process defines _____.
    a) User interface design
    b) Sequence of development activities
    c) Hardware configuration
    d) Coding syntax

58. Organizational structure in software projects helps in_____.
    a) Code reuse
    b) Role clarity
    c) Reducing testing
    d) Eliminating managers

59. Poor organizational planning leads to _____.
    a) Better coordination
    b) Project success
    c) Communication problems
    d) Faster delivery

60. Effective team structure improves _____.
    a) Software complexity
    b) Project coordination and control
    c) Hardware efficiency
    d) Data redundancy

SOFTWARE PROJECT PLANNING

1. **Computers are used in a wide variety of fields such as _____.**

   a) Only education
   b) Only business
   c) Rail ticket reservation and medical diagnosis systems
   d) Only entertainment

2. **Why should a computer work on a wide spectrum of software?**

   a) Software is expensive
   b) Computers are hardware devices
   c) Computers are used in many different application fields
   d) Programs are easy to write

3. **Software is NOT merely a collection of _____.**

   a) Documentation
   b) Users
   c) Computer programs
   d) Procedures

4. **A program is best defined as _____.**

   a) A collection of documents
   b) A set of instructions followed by a computer to perform a task
   c) A user manual
   d) A group of users

5. **A program is generally used by _____.**

   a) End users
   b) Third-party programmers
   c) The author of the program
   d) Software testers

6. **Which of the following is a limitation of a program?**

   a) Easy to debug
   b) Well documented
   c) Difficult for others to modify or debug
   d) Widely used

7. An example of a program mentioned in the text is _____.

a) Railway reservation system
b) Medical diagnosis software
c) A C program to sort names in ascending order
d) Operating system

8. Software is a collection of _____.

a) Hardware components
b) Users
c) Programs developed for widespread use
d) Computer languages

9. Software is expected to be _____.

a) Complex
b) User friendly
c) Author dependent
d) Without documentation

10. Software documentation helps in _____.

a) Increasing hardware speed
b) Preventing errors
c) Customization and ease of use
d) Reducing users

11. Customization of software means _____.

a) Deleting software
b) Changing hardware
c) Modifying software for specific applications
d) Writing programs from scratch

12. In the initial stages, software development was _____.

a) Highly organized
b) Strictly planned
c) Unmanaged
d) Fully automated

13. Rising delays and costs in software development led to _____.

a) Software removal
b) Structured approach to programming
c) Hardware replacement
d) End of software development

14. The rapid growth in software creation is shown by _____.

a) Hardware production
b) Boom in software employment
c) Reduced computer usage
d) Lack of programmers

15. Independent software development raised issues related to _____.

a) Electricity consumption
b) Compatibility, quality, reusability, and maintainability
c) Internet speed
d) Computer size

16. Which of the following is NOT a software development problem?

a) Poorly defined customer requirements
b) Good documentation
c) Schedule and cost increases
d) Poor quality software

17. The term "software crisis" refers to _____.

a) Hardware failure
b) Lack of computers
c) Problems in managing large volumes of software
d) Internet breakdown

18. The exponential growth of software problems made it _____.

a) Easy to manage software
b) Impossible to manage software
c) Easy to debug programs
d) Easy to reuse software

19. The study developed to solve software problems is called _____.

a) Computer Science
b) Information Technology
c) Software Engineering
d) Data Science

20. Software Engineering mainly focuses on _____.

a) Hardware manufacturing
b) Formal and systematic methods for software development
c) Writing single programs
d) Computer networking

21. According to IEEE, Software Engineering is the application of a …. approach to software development.

a) Random
b) Trial and error
c) Systematic, disciplined, and quantifiable
d) Informal

22. IEEE defines Software Engineering as the application of _____to software.

a) Mathematics
b) Science
c) Engineering
d) Programming

23. Software Engineering mainly focuses on the development, operation and _____of software.

a) Execution
b) Installation
c) Maintenance
d) Testing

24. A systematic approach in software engineering includes analysis, design, implementation and _____.

a) Compilation
b) Maintenance
c) Coding only
d) Debugging

25. CASE tools are used in software engineering to provide _____.

a) Manual support
b) Hardware assistance
c) Automated or semi-automated support
d) Internet services

26. Who defined Software Engineering as the practical application of scientific knowledge?

a) Dennis
b) Parnas
c) Boehm
d) Sommerville

27. According to Dennis (1975), Software Engineering is the application of principles, skills and

a) Tools
b) Art
c) Algorithms
d) Machines

28. Parnas states that software engineering exists when _____.
a) Only one program is written
b) Only one person uses the program
c) More than one person or version is involved
d) Program is very small

29. Fairley emphasized that software engineering is both a technological and … ..... discipline.
a) Educational
b) Scientific
c) Managerial
d) Mathematical

30. Fairley's definition highlights production of software within
a) High speed
b) Quality limits
c) Cost and time estimates
d) Memory limits

31. Sommerville states that Software Engineering deals with systems that are ……
a) Very small
b) Written by one person
c) Larger than those handled by a single individual
d) Temporary

32. Sommerville's definition includes both technical and _____aspects.
a) Social
b) Economic
c) Non-technical
d) Educational

33. Pomberger and Blaschek define software engineering as economical production of _ software.
a) Fast
b) Large
c) High-quality
d) Low-level

34. The basic principle of software engineering is to use _____methods.
a) Informal
b) Random
c) Structured, formal and disciplined
d) Unplanned

35. Software Engineering aims to provide _____.
a) Only programs
b) Methods, tools, and procedures
c) Hardware devices
d) Programming languages

36. Methods in software engineering provide _____.
a) Machines
b) Rules and steps to carry out tasks
c) Hardware support
d) Internet access

37. Which of the following is NOT a task supported by methods?
a) Requirements analysis
b) Design
c) Coding
d) Hardware manufacturing

38. Tools in software engineering provide _____support for methods.
a) Manual
b) Automated or semi-automated
c) Theoretical
d) Electrical

39. CASE stands for _____.
a) Computer Assisted Software Engineering
b) Computer Aided Software Engineering
c) Common Application Software Environment
d) Centralized Automated Software Engine

40. Standard procedures mainly define _____.
a) Programming languages
b) Sequence of methods and deliverables
c) Hardware components
d) Coding syntax

41. Standard procedures help in ensuring _____.
a) Speed only
b) Quality and control
c) Memory usage
d) Internet access

42. Software Engineering models represent _____.
a) Hardware structure
b) Programming syntax
c) Software development paradigms
d) Computer networks

43. Theories and methodologies of software development are referred to as _____
a) Programs
b) Algorithms
c) Paradigms
d) Compilers

44. Size-oriented software metrics are derived by normalizing _____.
a) Cost only
b) Hardware resources
c) Quality and/or productivity measures
d) Programming language

45. Size-oriented metrics are based on the _____of the software produced.
a) Complexity
b) Performance
c) Size
d) Quality

46. The term "normalizing" in size-oriented metrics means _____.
a) Increasing values
b) Ignoring measurements
c) Relating measures to software size
d) Removing errors

47. Size-oriented metrics mainly evaluate _____.
a) Hardware efficiency
b) Quality and productivity
c) Network performance
d) User satisfaction

48. Size-oriented software metrics consider the software that has been _____.
a) Designed
b) Planned
c) Produced
d) Estimated

49. Which of the following is essential for deriving size-oriented metrics?
a) Software size
b) Internet speed
c) Operating system
d) Compiler

50. Size-oriented metrics are most useful for measuring _____.
a) Programming syntax
b) Software productivity
c) Computer memory
d) CPU speed

51. Size-oriented metrics relate quality or productivity measures to _____.
a) Time
b) Cost
c) Software size
d) Hardware type

52. The overriding goal of software engineering is to produce a _____.
a) Low-cost system
b) Fast system
c) High-quality system
d) Large system

53. To achieve software quality, engineers must apply effective methods along with _____.

a) Old tools
b) Modern tools
c) Hardware devices
d) Programming languages

54. Measuring software helps to assess the quality of the _____.
a) Process only
b) Product only
c) Both process and product
d) Hardware

55. Which of the following is NOT a software quality indicator mentioned?
a) Correctness
b) Maintainability
c) Integrity
d) Scalability

56. Correctness refers to the degree to which software _____.
a) Is easy to use
b) Performs required functions
c) Is secure
d) Is portable

57. Correctness is commonly measured using _____.
a) Mean Time To Change
b) Defects per KLOC
c) Availability
d) DRE

58. A defect is defined as a verified lack of _____.
a) Performance
b) Speed
c) Conformance to requirements
d) Documentation

59. Maintainability is the ease with which software can be _____.
a) Executed
b) Corrected, adapted, or enhanced
c) Secured
d) Compiled

60. The time-oriented metric used for maintainability is _____.
a) MTBF
b) MTTC
c) KLOC
d) DRE

61. MTTC stands for _____.
a) Mean Time To Compile
b) Mean Time To Correct
c) Mean Time To Change
d) Mean Time To Check

62. Integrity measures a system's ability to withstand _____.
a) Errors
b) Failures
c) Attacks on security
d) Maintenance

63. Integrity depends on which two attributes?
a) Cost and time
b) Threat and security
c) Speed and memory
d) Quality and productivity

64. Threat is defined as the probability that _____.
a) A defect occurs
b) A system crashes
c) An attack will occur
d) Security fails

65. Security is the probability that an attack will be _____.
a) Detected
b) Delayed
c) Repelled
d) Ignored

66. Usability mainly attempts to quantify _____.
a) Efficiency
b) User friendliness
c) Reliability
d) Portability

67. Which is NOT a usability characteristic?
a) Time to learn
b) User productivity
c) User attitude
d) System security

68. DRE measures the effectiveness of _____.
a) Coding
b) Testing
c) Quality assurance activities
d) Documentation

69. The formula for Defect Removal Efficiency is _____.
a) D / (E + D)
b) E / (E + D)
c) E × D
d) E − D

70. In the DRE formula, E represents _____.
a) Errors after delivery
b) Errors before delivery
c) Total defects
d) Estimated defects

71. The ideal value of DRE is _____.
a) 0
b) 0.5
c) 1
d) Greater than 1

72. As E increases, the value of D is likely to _____.
a) Increase
b) Remain same
c) Decrease
d) Become zero always

73. DRE encourages teams to find defects _____.
a) After delivery
b) During maintenance
c) Before delivery
d) After failure

74. DREi measures a team's ability to find errors _____.
a) After project completion
b) Before passing to next phase
c) During maintenance only
d) During testing only

75. A quality objective is to make DREi approach _____.
a) 0
b) 0.5
c) 1
d) Infinity

76. Software quality factors cannot be measured directly because they are _____.
a) Too costly
b) Unclear in description
c) Hardware dependent
d) Optional

77. Reliability can be measured using attributes like _____.
a) MTTC
b) Mean time to failure
c) DRE
d) KLOC

78. Portability can be measured by counting _____.
a) Defects
b) Target-dependent statements
c) Test cases
d) Variables

79. Descriptive variable names relate to _____.
a) Completeness
b) Understandability
c) Portability
d) Efficiency

80. Presence of all required inputs and parameters relates to _____.
a) Conciseness
b) Completeness
c) Consistency
d) Testability

81. Redundant or unreachable code affects_____.
a) Efficiency
b) Conciseness
c) Usability
d) Reliability

82. Machine-dependent statements affect _____.
a) Reliability
b) Maintainability
c) Portability
d) Testability

83. Consistent indentation and naming relate to _____.
a) Consistency
b) Integrity
c) Usability
d) Reliability

84. Ease of future modification relates to _____.
a) Reliability
b) Maintainability
c) Portability
d) Efficiency

85. Availability of test cases and clear pseudo-code relates to _____.
a) Testability
b) Usability
c) Integrity
d) Efficiency

86. Use of GUI and user manuals improves _____.
a) Reliability
b) Usability
c) Integrity
d) Portability

87. Software engineering applies management activities to ensure development is_____.
a) Fast and cheap
b) Random and flexible
c) Systematic, disciplined and measured
d) Informal and adaptive

88. Which of the following is NOT a management activity applied in software engineering?
a) Planning
b) Coordinating
c) Compiling
d) Monitoring

89. Reporting in software engineering helps to _____.
a) Write code
b) Ensure development progress is tracked
c) Execute programs
d) Debug software

90. Measurement goals in software engineering include _____.
a) Organizational objectives
b) Software process improvement goals
c) Both (a) and (b)
d) Hardware improvement goals

91. Goal-driven measurement selection means measurements are chosen based on _____.

a) Available tools
b) Project goals
c) Developer skills
d) Hardware constraints

92. Measurement validity ensures that a metric _____.
a) Is cheap
b) Is easy to calculate
c) Measures what it is intended to measure
d) Is popular

93. Which of the following is NOT a type of software measurement?
a) Size measurement
b) Structure measurement
c) Resource measurement
d) Hardware measurement

94. Quality measurement focuses on measuring _____.
a) Hardware speed
b) Software characteristics
c) Network performance
d) User hardware

95. Data collection in software measurement can be done using _____.
a) Only automated techniques
b) Only manual techniques
c) Survey techniques and automated/manual methods
d) Interviews only

96. Software measurement models involve _____.
a) Only implementation
b) Model building, calibration and evaluation
c) Coding and testing
d) Hardware setup

97. Interpretation and refinement of models are part of _____.
a) Data collection
b) Measurement models
c) Testing
d) Coding

98. Rationale management includes all EXCEPT _____.
a) Problem
b) Alternatives considered
c) Criteria used
d) Source code

99. The final step in rationale management is _____.
a) Debate
b) Decision
c) Criteria selection
d) Problem identification

100. Unit testing compares _____.
a) Subsystems with requirements
b) Objects with design models
c) System with requirements
d) Software with hardware

101. Integration testing focuses on _____.
a) Individual modules
b) User acceptance
c) Combination of subsystems
d) Exceptional cases

102. System testing compares the system with _____.
a) Code
b) Design
c) Requirements model
d) Test cases

103. Acceptance testing is carried out to _____.
a) Improve code quality
b) Validate customer requirements
c) Check syntax errors
d) Improve performance

104. Software configuration management is responsible for _____.
a) Coding standards
b) Establishing baselines and controlling changes
c) Debugging
d) Testing

105. Project management ensures delivery of software _____.
a) Early
b) Late
c) On time and within budget
d) With more features

106. Which is NOT a project management activity?
a) Planning and budgeting
b) Hiring developers
c) Monitoring project status
d) Writing test cases

107. Software metrics are _____.
a) Qualitative data
b) Numerical data related to software development
c) Hardware measurements
d) Programming rules

108. Software metrics strongly support _____.
a) Coding
b) Testing only
c) Software project management
d) Hardware selection

109. Metrics support cost estimation and scheduling in which management function?
a) Organizing
b) Controlling
c) Planning
d) Improving

110. Size and schedule metrics mainly influence _____.
a) Planning
b) Organizing
c) Controlling
d) Improving

111. Tracking project progress using metrics relates to _____.
a) Planning
b) Organizing
c) Controlling
d) Improving

112. Metrics used to identify process improvement areas relate to _____.
a) Planning
b) Organizing
c) Controlling
d) Improving

113. A metric quantifies a characteristic of a _____.
a) Person
b) Computer
c) Process or product
d) Network

114. Which of the following is a raw metric?
a) Defects per KLOC
b) LOC per staff-hour
c) Number of source lines of code
d) Cost performance index

115. Which is an example of a derived metric?
a) Number of requirements
b) Number of test cases
c) Defects per thousand lines of code
d) Number of staff-hours

116. Derived metrics are obtained from _____.
a) Hardware tools
b) One or more raw metrics
c) Interviews
d) Surveys only

117. A project is defined as _____.
a) An unplanned activity
b) A planned undertaking to present results at a specified time
c) A routine operation
d) A continuous process

118. In the project definition, the word "undertaking" refers to _____.
a) Writing documentation
b) Executing a program
c) Making a new product or changing an old product
d) Testing software

119. Software project management begins with _____.
a) Testing
b) Coding
c) Project planning
d) Maintenance

120. Project planning consists of a set of activities that begin with_____.
a) Scheduling
b) Design
c) Estimation
d) Risk analysis

121. Estimation in software projects includes estimating _____.
a) Hardware and tools only
b) Resources, cost, and schedule
c) Programming languages
d) User requirements

122. Accurate estimation requires all EXCEPT _____.
a) Experience
b) Historical information
c) Courage to commit to quantitative measures
d) Elimination of risk

123. Estimation often involves committing to quantitative measures when _____.
a) Only quantitative data exists
b) Only qualitative data exists
c) No data exists
d) Exact data exists

124. Estimation in software projects involves inherent _____.
a) Accuracy
b) Simplicity
c) Risk
d) Certainty

125. Risk in estimation leads to _____.
a) Cost reduction
b) Uncertainty
c) Perfection
d) Stability

126. Software project estimation should be viewed as _____.
a) Only science
b) Only art
c) Both art and science
d) Guesswork

127. The primary goal of estimation is to be _____.
a) Fast
b) Approximate
c) Accurate
d) Flexible

128. Estimation must be done carefully because _____.
a) It eliminates all risks
b) It involves uncertainty and risk
c) It is easy
d) It requires no experience

129. Historical information is useful in estimation because it _____.
a) Removes uncertainty
b) Helps improve accuracy
c) Eliminates qualitative data
d) Guarantees success

130. The main objective of software project planning is to provide a framework to estimate

a) Coding standards
b) Resources, cost, and schedule
c) Testing techniques
d) User satisfaction

131. Software project planning estimates are initially made _____.
a) At the end of the project
b) During testing
c) At the beginning of the project
d) During maintenance

132. Project estimates should be updated _____.
a) Only once
b) Only at the end
c) Regularly as the project progresses
d) Never

133. Reasonable estimates are achieved through a process of _____.
a) Coding
b) Information discovery
c) Testing
d) Debugging

134. The first activity in software project planning is _____.
a) Resource allocation
b) Software scope determination
c) Scheduling
d) Cost estimation

135. Software scope describes all EXCEPT _____.
a) Function
b) Performance
c) Constraints
d) Programming language

136. Performance considerations include _____.
a) Memory only
b) Processing and response time requirements
c) Hardware price
d) Staff experience

137. Constraints identify limits imposed by _____.
a) Users only
b) External hardware, memory, or existing systems
c) Programmers
d) Documentation

138. To define scope, communication should begin between _____.
a) Manager and tester
b) Customer and developer
c) Analyst and programmer
d) User and system

139. Context-free questions are used to _____.
a) Test software
b) Understand the basic problem and solution
c) Design algorithms
d) Write code

140. Meta questions mainly focus on the _____.
a) Technical solution
b) Cost estimation
c) Effectiveness of the meeting
d) System design

141. FAST stands for _____.
a) Fast Application Software Technique
b) Facilitated Application Specification Techniques
c) Formal Application System Tool
d) Functional Analysis System Technique

142. Which of the following is NOT a development resource?
a) Development environment
b) Reusable software components
c) Human resources
d) Network bandwidth

143. Each resource is specified by how many characteristics?
a) Two
b) Three
c) Four
d) Five

144. The last two resource characteristics together represent a _____.
a) Cost estimate
b) Time window
c) Schedule buffer
d) Risk factor

145. Human resources include specification of _____.
a) Hardware type
b) Organizational position and specialty
c) Programming language
d) Network type

146. The number of people required for a project is determined after _____.
a) Coding
b) Testing
c) Estimation of development effort
d) Deployment

147. Off-the-shelf components are _____.
a) Newly developed
b) Ready for use and fully validated
c) High risk
d) Incomplete

148. Full-experience components involve _____.
a) No past experience
b) Team with full experience in application area
c) Completely new design
d) High modification risk

149. Partial-experience components require _____.
a) No modification
b) Minor modification
c) Substantial modification
d) No analysis

150. New components are _____.
a) Purchased from vendors
b) Built specifically for the current project
c) Reused without changes
d) Already tested

151. Software Engineering Environment (SEE) includes
a) Only software
b) Only hardware
c) Both hardware and software
d) Network only

152. A project planner must verify availability of environmental resources for a required

a) Budget
b) Time window
c) Team size
d) Phase

153. The chronological sequence of software project phases is called _____.
a) Software process
b) Software life cycle
c) Software model
d) Project plan

154. Software life cycle extends from development to _____.
a) Testing
b) Deployment
c) Retirement
d) Maintenance only

155. The cyclical nature of the life cycle indicates that phases can be _____.
a) Skipped
b) Repeated
c) Removed
d) Ignored

156. Which is NOT a goal of requirements analysis and planning?
a) Determining steps
b) Identifying resources
c) Coding modules
d) Defining automation

157. One product of the requirements analysis phase is _____.
a) Program code
b) User requirements
c) Test cases
d) Physical data model

158. System specification acts as a _____.
a) Design document
b) Contract between client and producer
c) Test report
d) Code manual

159. An exact project schedule is produced during _____.
a) Requirements analysis
b) System specification
c) Design
d) Implementation

160. System and component design focuses on _____.
a) Coding
b) Mapping requirements to components
c) Testing
d) Deployment

161. Which is NOT a product of design phase?
a) Logical data model
b) System architecture
c) Algorithmic structure
d) Program code

162. Coding is performed during _____.
a) Design phase
b) Implementation phase
c) System testing
d) Maintenance

163. Translation of logical data model into physical data model occurs during _____
a) Design
b) Implementation
c) Testing
d) Maintenance

164. System testing ensures that the system fulfils the _____.
a) Design
b) Code
c) System specification
d) User manual

165. The longest phase of the software life cycle is _____.
a) Design
b) Testing
c) Implementation
d) Operation and maintenance

166. Correcting errors during actual operation is part of _____.
a) Testing
b) Design
c) Maintenance
d) Planning

167. During development, documentation mainly supports _____.
a) Coding speed
b) Communication
c) Cost estimation
d) Hardware usage

168. Quality assurance focuses on fulfilling quality criteria such as_____.
a) Speed and size
b) Correctness, reliability, maintainability
c) Cost and schedule
d) Hardware performance

169. Planning the software development process primarily involves_____ .
a) Writing code
b) Testing software
c) Several important considerations
d) Installing software

170. The first consideration in planning the software development process is to ……
a) Select tools
b) Hire programmers
c) Define a product life-cycle model
d) Prepare documentation

171. A software project passes through several phases before it is _____.
a) Designed
b) Tested
c) Ready for practical use
d) Installed

172. A framework in software projects is used to define the _____.
a) Cost structure
b) Flow of activities
c) Programming language
d) Hardware platform

173. Which of the following is NOT a software development activity mentioned?
a) Define
b) Develop
c) Manufacture
d) Test

174. The correct sequence of activities includes _____.
a) Deliver → Develop → Test
b) Define → Develop → Test → Deliver
c) Operate → Define → Test
d) Maintain → Test → Develop

175. Maintenance of software occurs _____.
a) Before delivery
b) During testing
c) After delivery
d) Only during development

176. A software life-cycle model defines _____.
a) Coding style
b) Flow of development activities
c) Hardware requirements
d) Programming syntax

177. Different software projects may use variations of life-cycle models based on _____.
a) Programmer skills
b) Deliverables and milestones
c) Operating systems
d) Compiler type

178. Before starting a project, a life-cycle model must be _____.
a) Designed
b) Tested
c) Selected and finalized
d) Modified

179. Which of the following is a well-defined software life-cycle model?
a) Linear model
b) Waterfall model
c) Random model
d) Trial model

180. The Waterfall model is best described as _____.
a) Iterative
b) Random
c) Sequential
d) Incremental

181. In the Waterfall model, each phase _____.
a) Overlaps with others
b) Is completed before the next begins
c) Is optional
d) Is repeated continuously

182. The Prototype model is mainly used when _____.
a) Requirements are well defined
b) Requirements are unclear
c) Project is very small
d) No user interaction is needed

183. A prototype is _____.
a) Final software
b) Partial or preliminary version of software
c) Test document
d) Design specification

184. The main advantage of the Prototype model is _____.
a) Reduced documentation
b) Better requirement understanding
c) Faster coding
d) Lower maintenance

185. The Spiral model primarily focuses on _____.
a) Coding
b) Testing
c) Risk analysis
d) Documentation

186. The Spiral model combines features of _____.
a) Waterfall and Prototype models
b) Prototype and Agile models
c) Waterfall and Incremental models
d) Agile and RAD models

187. Each loop of the Spiral model represents a _____.
a) Testing phase
b) Risk analysis
c) Development phase
d) Complete development cycle

188. The Object-oriented life-cycle model is based on _____.
a) Functions
b) Procedures
c) Objects
d) Algorithms

189. In object-oriented life-cycle models, the system is viewed as a collection of _____.

a) Modules
b) Objects
c) Programs
d) Processes

190. Object-oriented models support _____.
a) Reusability
b) Modularity
c) Maintainability
d) All of the above

191. Selection of a software life-cycle model depends on _____.
a) Hardware only
b) Project deliverables and milestones
c) Compiler used
d) Coding speed

192. A life-cycle model provides a structured way to _____.
a) Write programs
b) Manage software development
c) Increase hardware speed
d) Reduce memory usage

193. Which model is most suitable for large, high-risk projects?
a) Waterfall
b) Prototype
c) Spiral
d) Object-oriented

194. Software life-cycle models define the sequence of development activities because

a) Coding must start early
b) Projects need structured flow
c) Testing is optional
d) Maintenance is avoided

195. Different projects may require different life-cycle models because _____.
a) All software is same
b) Deliverables and milestones vary
c) Programming languages differ
d) Developers change

196. Completing a software project is mainly a _____.

a) Individual effort
b) Machine-driven process
c) Team effort
d) Documentation task

197. If 'n' individuals are organized into 't' teams and each team is assigned one or more functional tasks, the approach is considered _____.

a) Least productive
b) Moderately productive
c) Most productive
d) Unstructured

198. Which option for applying human resources is considered the most productive?

a) Individuals working independently
b) Informal teams coordinated loosely
c) Teams assigned functional tasks
d) Random allocation of people

199. Who initiates, plans, tracks, and manages resources of the entire software project?

a) Analyst
b) Module Leader
c) Project Manager
d) Architect

200. A Module Leader is mainly responsible for _____.

a) Customer communication
b) Managing entire project
c) Ensuring proper functionality of a module
d) Writing user manuals

201. Requirement analysis is primarily carried out by the _____.

a) Developer
b) Tester
c) Analyst
d) Reviewer

202. (i) A Domain Consultant mainly provides _____.

a) Coding support
b) Testing strategies
c) Domain-specific technical knowledge
d) Project scheduling

(ii) Who reviews project documents and code for technical correctness & standard compliance?

a) Tester
b) Architect
c) Reviewer
d) Analyst

203. Designing the solution after requirements are specified is the role of the _____.

a) Analyst
b) Developer
c) Architect
d) Tester

204. Who writes code, tests it, and delivers it error-free?

a) Tester
b) Architect
c) Developer
d) Reviewer

205. Defects found during testing are logged and reported by the _____.

a) Developer
b) Module Leader
c) Tester
d) Project Manager

206. In which team structure do all members participate in decision-making?

a) Hierarchical team
b) Chief programmer team
c) Democratic team
d) Centralized team

207. The Democratic team is also known as the _____.

a) Chief team
b) Egoless team
c) Hierarchical team
d) Structured team

208. In a Democratic team, leadership _____.

a) Is fixed
b) Is decided by seniority
c) Rotates among team members
d) Is assigned by management

209. Which is NOT an advantage of the Democratic team?

a) High job satisfaction
b) Open communication
c) Low communication overhead
d) Learning opportunity

210. One major disadvantage of the Democratic team is _____.

a) Poor documentation
b) Communication overhead
c) No learning opportunity
d) Poor quality software

211. The Chief Programmer Team model was proposed by _____.

a) Weinberg
b) Royce
c) Baker
d) Boehm

212. Which of the following is NOT a member of the Chief Programmer Team?

a) Chief Programmer
b) Project Assistant
c) Project Secretary
d) Quality Manager

213. Who controls project progress and assumes overall responsibility in the Chief Programmer Team?

a) Project Assistant
b) Project Secretary
c) Specialist
d) Chief Programmer

214. The main task of the project secretary is _____.

a) Coding
b) Testing
c) Administration of project library
d) Design validation

215. Which is an advantage of the Chief Programmer Team?

a) Suitable for very large teams
b) Less responsibility on one person
c) Better control due to direct involvement
d) No need for specialists

216. A major disadvantage of the Chief Programmer Team is _____.

a) Poor communication
b) Limited to small teams
c) Low productivity
d) No documentation

217. Hierarchical team structure is based on _____.

a) Group consensus
b) Egoless approach
c) Phase-wise decomposition
d) Informal coordination

218. In a hierarchical organization, the number of hierarchy levels increases with _____.

a) Team skill
b) Project complexity
c) Project size
d) Documentation level

219. Which of the following is NOT a responsibility of the project manager?

a) Personnel selection
b) Project progress checks
c) Coding modules
d) Managing cost overruns

220. Middle-level managers are mainly responsible for _____.

a) Requirement gathering
b) Phase-related planning and control
c) Testing only
d) Documentation only

221. Project management is primarily concerned with _____.

a) Writing program code
b) Defining and achieving targets
c) System testing only
d) Database management

222. Project management optimizes the use of resources such as _____.

a) Time and people only
b) Time, money, people, materials, energy and space
c) Hardware and software only
d) Documents and reports

223. A project is best described as _____.

a) A permanent organizational function
b) An ongoing functional activity
c) A temporary endeavor creating a unique product or service
d) A repetitive process

224. The duration of a project is the time between its _____.

a) Planning and testing
b) Design and implementation
c) Start and completion
d) Approval and delivery

225. Which of the following is NOT a typical project?

a) Construction of a building
b) Development of computer software
c) Daily payroll processing
d) Manufacturing a vehicle prototype

226. The project manager usually _____.

a) Writes program code
b) Tests the software
c) Maintains progress and coordination
d) Designs system architecture

227. The primary goal of a project manager is to _____.

a) Produce the end product directly
b) Reduce overall risk of failure
c) Increase documentation
d) Perform coding activities

228. Which of the following is a project management activity?

a) Coding modules
b) Planning the work
c) Writing user manuals
d) Debugging programs

229. Estimating resources in project management refers to _____.

a) Selecting tools
b) Predicting human and material needs
c) Writing test cases
d) Creating documents

230. Assigning tasks is part of which project management function?

a) Organizing the work
b) Reporting progress
c) Analyzing results
d) Controlling execution

231. Monitoring and regulating project activities is known as _____.

a) Directing activities
b) Project execution
c) Controlling project execution
d) Risk assessment

232. Reporting progress mainly helps in _____.

a) Coding
b) Tracking project status
c) Designing systems
d) Testing software

233. How many project control variables are managed in project management?

a) Three
b) Four
c) Five
d) Six

234. Which of the following is NOT a project control variable?

a) Time
b) Cost
c) Quality
d) Technology

235. The amount of time required to complete a project is referred to as _____.

a) Cost
b) Scope
c) Time
d) Quality

236. Project cost is generally calculated based on _____.

a) Hardware price
b) Team size only
c) Time multiplied by resource cost
d) Quality standards

237. Quality in project management is influenced by _____.

a) Number of people
b) Time spent on tasks
c) Project scope only
d) Project duration

238. Scope of a project refers to _____.

a) Budget estimation
b) Time schedule
c) Requirements and objectives
d) Risk assessment

239. Potential points of failure in a project are called _____.
a) Constraints
b) Errors
c) Risks
d) Defects

240. Most project risks can be overcome by _____.

a) Ignoring them
b) Increasing documentation
c) Providing enough time and resources
d) Reducing scope

241. Which project variables are usually contracted?

a) Time, cost, quality, scope
b) Risk only
c) Resources and tools
d) People and materials

242. Final project values are decided through _____.

a) System testing
b) Customer negotiation
c) Coding phase
d) Maintenance phase

243. When some variables are fixed by the customer, the remaining ones are set by ____.

a) Developers
b) Project management
c) Test engineers
d) System analysts

244. Project management decisions should ideally be based on _____.

a) Assumptions
b) Experience only
c) Solid estimation techniques
d) Trial and error

SOFTWARE COST ESTIMATION

1. Software is considered the most expensive element in computer-based systems because _____.
a) Hardware is cheaper
b) Software development involves many variables
c) Software is reusable
d) Software is easy to modify

2. A large error in software cost estimation may result in _____.
a) Better performance
b) Reduced effort
c) Profit or loss difference
d) Faster delivery

3. Which of the following variables affect software cost estimation?
a) Human
b) Technical
c) Environmental
d) All of the above

4. Software cost estimation is necessary mainly for _____.
a) Coding
b) Testing
c) Scheduling and cost planning
d) Debugging

5. Software cost estimation is usually based on _____.
a) Mathematical proofs
b) Current project size only
c) Comparison with previous projects
d) Customer demand

6. Empirical data from previous projects is often unreliable because _____.
a) Data is always incorrect
b) Projects are identical
c) Technical and organizational conditions change
d) Tools never change

7. The uniqueness of software systems leads to _____.
a) Large empirical datasets
b) Few comparable projects
c) Accurate estimates
d) Standardized cost

8. According to Brooks, project time consists of _____.
a) Coding and testing
b) Productive work and communication
c) Design and implementation
d) Planning and control

9. If no communication were required, project time (t) would vary with team size (n) as
a) $t \approx n$
b) $t \approx 1/n$
c) $t \approx n^2$
d) $t \approx \log n$

10. The formula considering communication overhead is _____.
a) $t \approx 1/n$
b) $t \approx n^2$
c) $t \approx 1/n + k \cdot n^2/2$
d) $t \approx k/n$

11. "Adding manpower to a late software project makes it later" was stated by _____.

a) Boehm
b) Putnam
c) Brooks
d) Albrecht

12. In the classical sequential model, the highest effort is spent on _____.
a) Design
b) Implementation
c) Testing
d) Analysis

13. In a prototyping-oriented model, maximum effort is spent on _____.
a) Design
b) Implementation
c) Testing
d) Problem analysis & specification

14. Object-oriented and prototyping-oriented models spend maximum effort on _____.

a) Implementation
b) Testing
c) Analysis and specification
d) Maintenance

15. Which option suggests postponing estimation to reduce errors?
a) Early estimation
b) Delay estimation
c) Decomposition

d) Empirical modeling

16. "Divide and conquer" approach in estimation refers to _____.
a) Empirical estimation
b) Delay estimation
c) Decomposition techniques
d) LOC estimation

17. Empirical models are based on _____.
a) Guesswork
b) Historical data
c) Algorithms only
d) Current cost

18. Which is NOT a factor affecting software cost estimation?
a) Estimator's experience
b) Number of staff
c) Color of software interface
d) Complexity of software

19. Expected size of a program is usually measured in terms of _____.
a) CPU speed
b) LOC
c) Memory
d) Time

20. Knowledge of the problem domain affects _____.
a) Testing only
b) Personnel efficiency
c) Hardware cost
d) Compilation

21. LOC stands for
a) Level of Complexity
b) Lines of Code
c) Language Oriented Code
d) Logical Operation Count

22. LOC-based estimation uses which type of values?
a) Random
b) Optimistic, Most likely, Pessimistic
c) Fixed
d) Absolute

23. In the given CAD example, the expected LOC value is _____.
a) 4600
b) 6900
c) 8600
d) 6800

24. The total LOC estimate of a system is obtained by _____.
a) Averaging all modules
b) Selecting the highest value
c) Summing expected values of all modules
d) Ignoring pessimistic values

25. Process-based estimation decomposes the project into _____.
a) Modules
b) Activities and tasks
c) Code lines
d) Classes only

26. Effort in process-based estimation is usually expressed in _____.
a) LOC
b) Function Points
c) Person-months
d) CPU cycles

27. Average labour rates are applied to estimate _____.
a) Quality
b) Size
c) Cost
d) Complexity

28. (i) The final step in process-based estimation is _____.
a) Estimating scope
b) Computing cost and effort
c) Selecting tools
d) Risk analysis

(ii) Which of the following are software cost estimation models?
a) Waterfall, Spiral, Agile
b) COCOMO, Putnam, Function Point
c) Black box, White box, Grey box
d) RAD, Incremental, Prototype

29. COCOMO stands for: _____.

a) Constructive Cost Model
b) Continuous Cost Model
c) Controlled Cost Model
d) Computation Cost Model

30. COCOMO is an example of a: _____.

a) Top-down estimation technique
b) Empirical estimation technique
c) Algorithmic cost estimation model
d) Heuristic estimation model

31. COCOMO uses which estimation approach?

a) Top-down
b) Bottom-up
c) Expert judgment
d) Delphi method

32. In COCOMO, estimation begins at: _____.

a) System level
b) Project level
c) Module/subsystem level
d) Organization level

33. Which of the following is NOT **a module-level effort multiplier?**

a) Product complexity
b) Programmer capability
c) Virtual machine experience
d) Hardware cost

34. Which is the first step **in COCOMO cost estimation?**

a) Compute total effort
b) Perform sensitivity analysis
c) Identify subsystems and modules
d) Add development costs

35. Sensitivity analysis in COCOMO is used to: _____.

a) Reduce development time
b) Improve product quality
c) Establish tradeoff benefits
d) Increase team size

36. COCOMO estimates are finally compared with: _____.

a) Function Point estimates
b) Putnam estimates
c) Top-down Delphi estimation
d) Expert opinion

37. How many project modes are defined in the basic COCOMO model?

a) Two
b) Three
c) Four
d) Five

38. Which COCOMO mode is suitable for small, simple projects?

a) Embedded
b) Semi-detached
c) Organic
d) Prototype

39. Organic mode projects typically have: _____.

a) Tight hardware constraints
b) Mixed experience teams
c) Less rigid requirements
d) High system complexity

40. Semi-detached mode projects are characterized by: _____.

a) Very small teams
b) Completely rigid requirements
c) Mixed experience levels
d) No hardware constraints

41. Embedded mode software is developed under: _____.

a) Flexible constraints
b) No hardware dependency
c) Tight hardware and operational constraints

42. Less documentation Flight control software is an example of: ____.

a) Organic mode
b) Semi-detached mode
c) Embedded mode
d) Prototype model

43. In the COCOMO effort equation, PM stands for: _____.

a) Project months
b) Program modules
c) Programmer months
d) Product metrics

44. TDEV in COCOMO refers to: _____.

a) Total development effort
b) Product development time
c) Testing development effort
d) Technical development value

45. KLOC represents: _____.

a) Known lines of code
b) Thousands of logical operations
c) Number of delivered lines of code (in thousands)
d) Kernel lines of code

46. Which parameters are coefficients in the COCOMO equations?

a) bi and di
b) PM and TDEV
c) ai and ci
d) KLOC and PM

47. Which mode has the highest exponent bi**?**

a) Organic
b) Semi-detached
c) Embedded
d) All are same

48. Staffing level estimation can be done only after estimating: _____.

a) Project risk
b) Development effort
c) Software size
d) Maintenance cost

49. Staffing requirement is highest **during: _____.**

a) Requirement analysis
b) High-level design
c) Build phase
d) Maintenance phase

50. Adequate staffing should have: _____.

a) Only experienced people
b) Only freshers
c) A blend of experienced and inexperienced people
d) Contract staff only

51. Software maintenance typically consumes: _____.

a) 10–20% of funds
b) 20–30% of funds
c) 40–70% of funds
d) 80–90% of funds

52. Which is a non-technical factor **affecting maintenance cost?**

a) Code complexity
b) Application domain
c) Algorithm design
d) Programming language

53. Maintenance costs increase when: _____.

a) Program is new
b) Staff is stable
c) Program age increases
d) Hardware is stable

54. Maintenance costs are reduced when: _____.

a) Documentation is missing
b) Original developer maintains the system
c) Hardware changes frequently
d) Staff turnover is high

55. Programs become obsolete when: _____.

a) Code size increases
b) Hardware becomes cheaper
c) Application becomes obsolete
d) Documentation improves

56. Payroll software maintenance cost increases mainly due to: _____.

a) Mathematical changes
b) Taxation system changes
c) Compiler upgrades
d) Algorithm optimization

57. Boehm's maintenance cost estimation uses: _____.

a) KLOC
b) EAF
c) Annual Change Traffic (ACT)
d) Function points

58. ACT is defined as: _____.

a) Total lines of code
b) Fraction of code changed per year
c) Effort multiplier
d) Maintenance manpower

59. ACT formula is: _____.

a) DSItotal / DSIadded
b) (DSIadded + DSImodified) / DSItotal
c) DSImodified × DSItotal
d) DSItotal − DSIadded

60. PM in maintenance estimation refers to: _____.

a) Project management
b) Programmer months
c) Product metrics
d) Program modules

61. The enhanced Boehm maintenance equation includes: _____.

a) KLOC
b) TDEV
c) EAF
d) bi

62. EAF is used because: _____.

a) Development and maintenance efforts are identical
b) Maintenance effort multipliers differ from development
c) It reduces ACT
d) It reduces cost

REQUIREMENT SPECIFICATION

1. A complete understanding of software requirements is essential for: _____.

a) Coding efficiency
b) Software testing
c) Success of software development
d) System maintenance

2. Requirements analysis is a process of: _____.

a) Coding and testing
b) Discovery, refinement, modeling, and specification
c) Debugging and validation
d) Implementation and maintenance

3. The specification produced during requirements analysis becomes the: _____.

a) Final software product
b) Basis for all software engineering activities
c) Testing document
d) Coding guideline

4. Requirements analysis bridges the gap between: _____.

a) Design and coding
b) Testing and maintenance
c) System-level software allocation and software design
d) Planning and implementation

5. Requirements analysis enables the system engineer **to: _____.**
a) Write program code
b) Specify software function and performance
c) Perform unit testing
d) Maintain software

6. Which of the following is specified during requirements analysis?
a) Coding standards
b) User manuals
c) Software interfaces with other system elements
d) Compiler options

7. Requirements analysis helps establish: _____.
a) Test cases
b) Design constraints
c) Debugging strategy
d) Deployment plan

8. Requirements analysis allows the software engineer **to: _____.**

a) Refine software allocation
b) Generate object code
c) Prepare test scripts
d) Install the system

9. Which models are built during requirements analysis?

a) Hardware models only
b) Process, data, and domain models
c) Source code models
d) Network models

10. Requirement analysis provides information for: _____.

a) Data, architectural, and procedural design
b) Coding and debugging
c) Hardware procurement
d) Software installation

11. How many areas of effort are involved in requirements analysis?

a) Three
b) Four
c) Five
d) Six

12. Which of the following is NOT **an area of requirements analysis?**

a) Problem recognition
b) Modeling
c) Coding
d) Review

13. The first step in requirements analysis is: _____.

a) Modeling
b) Evaluation and synthesis
c) Problem recognition
d) Specification

14. Problem recognition focuses on: _____.

a) Designing solutions
b) Identifying basic problem elements perceived by users
c) Coding modules
d) Testing software

15. During evaluation and synthesis, the analyst: _____.

a) Writes test cases
b) Defines data objects and software functions
c) Implements algorithms
d) Debugs errors

16. Evaluation and synthesis continues until: _____.

a) Coding starts
b) Customer and analyst are confident about specifications
c) Testing is completed
d) Documentation is finalized

17. Modelling in requirements analysis is compared to: _____.

a) Writing source code
b) Creating a blueprint for a building
c) Debugging programs
d) Testing software

18. The main purpose of modeling is to: _____.

a) Reduce coding time
b) Improve communication and ensure architectural soundness
c) Generate executable code
d) Reduce maintenance cost

19. Models help understand: _____.

a) Compiler behaviour
b) Data and control flow, functional processing, and behaviour
c) Programming language syntax
d) Hardware configurations

20. The final output of requirements analysis is: _____.

a) Test plan
b) Software Requirements Specification (SRS)
c) Design document
d) User manual

21. A good SRS should be: _____.

a) Informal and flexible
b) Understandable, complete, consistent, and unambiguous
c) Code-oriented
d) Implementation-dependent

22. Requirements reviews are mainly used for: _____.

a) Coding verification
b) Validation of requirements
c) System installation
d) User training

23. One basic specification principle is to: _____.

a) Combine design and implementation
b) Separate functionality from implementation
c) Focus on coding details
d) Ignore environment

24. Cognitive models describe a system as: _____.

a) Designed by engineers
b) Implemented by programmers
c) Perceived by users
d) Tested by QA teams

25. Specifications must be tolerant of: _____.

a) Errors
b) Incompleteness
c) Implementation details
d) Hardware changes

26. A specification is always: _____.

a) Complete
b) Executable
c) An abstraction of reality
d) Error-free

27. Which section of SRS describes system interfaces?

a) Functional requirements
b) External interface and data flow
c) Glossary
d) Acceptance criteria

28. Which is a desirable property of SRS?

a) Ambiguous
b) Traceable
c) Informal
d) Implementation-oriented

29. Which property ensures requirements can be tested?

a) Functional
b) Verifiable
c) Easily changed
d) Correct

30. Which SRS property supports requirement change?

a) Traceable
b) Easily changed
c) Verifiable
d) Functional

31. Formal specifications are mainly used to: _____.

a) Write source code
b) Specify functional characteristics clearly
c) Perform system testing
d) Debug errors

32. Formal specifications support: _____.

a) Informal reasoning
b) Formal reasoning and verification
c) Rapid prototyping
d) Coding automation

33. There are how many types of formal specifications?

a) One
b) Two
c) Three
d) Four

34. Relational notations are based on: _____.

a) States and transitions
b) Entities and attributes
c) Events and tables
d) Inputs and outputs

35. Which is an example of relational notation?

a) Petri nets
b) Decision tables
c) Algebraic axioms
d) State transition diagrams

36. State-oriented notations focus on: _____.

a) Entities only
b) Current state and stimuli
c) Past execution history
d) Program structure

37. Which is an example of state-oriented notation?

a) Regular expressions
b) Recurrence relations
c) Finite-state mechanisms
d) Algebraic equations

38. PSL stands for: _____.

a) Problem Specification Language
b) Program Statement Language
c) Problem Statement Language
d) Project Software Language

39. PSL was developed by: _____.

a) Boehm
b) Daniel Teichrow
c) R. Balzar
d) D.T. Ross

40. PSA is: _____.

a) A modeling technique
b) A testing tool
c) An automated analyzer for PSL
d) A design language

41. Which language was developed for real-time systems?

a) PSL
b) RSL
c) GIST
d) SADT

42. SADT was developed by: _____.

a) Daniel Teichrow
b) R. Balzar
c) D.T. Ross
d) Barry Boehm

43. SSA mainly uses: _____.

a) Textual descriptions
b) Object-oriented models
c) Data flow diagrams
d) State machines

44. GIST is based on: _____.

a) State-oriented model
b) Relational model of objects and attributes
c) Procedural model
d) Data flow model

45. GIST specification consists of: _____.

a) One part
b) Two parts
c) Three parts
d) Four parts

SOFTWARE DESIGN

1. **Design is the first step in which phase of software development?**

a) Maintenance phase
b) Testing phase
c) Development phase
d) Deployment phase

2. **When the term *design* is used as a verb, it represents: _____.**
a) The final output
b) The documentation
c) The process of design
d) The system model

3. **When the term *design* is used as a noun, it represents: _____.**

a) The design process
b) The design result or model
c) The coding standard
d) The test plan

4. **The goal of the design process is to: _____.**

a) Write program code
b) Test the system
c) Produce a model or representation of the system
d) Debug errors

5. **Software design translates requirements into: _____.**

a) Source code
b) Test cases
c) A blueprint for constructing software
d) User manuals

6. **Software design is described as: _____.**

a) A linear process
b) A one-time activity
c) An iterative process
d) A testing activity

7. A good software design should exhibit: _____.

a) Flat structure
b) Hierarchical representation
c) No modularity
d) Single module

8. Which is a characteristic of good design?

a) High interface complexity
b) Monolithic structure
c) Modular structure
d) Code duplication

9. A good design should have: _____.

a) Mixed data and procedure
b) Distinct representation of data and procedure
c) No data abstraction
d) No procedure

10. Independent functional characteristics should be exhibited by: _____.

a) Variables
b) Modules
c) Interfaces
d) Algorithms

11. A good design should minimize: _____.

a) Modularity
b) Documentation
c) Interface complexity
d) Hierarchy

12. Fundamental design concepts help in: _____.

a) Writing code faster
b) Getting a program right
c) Reducing documentation
d) Eliminating testing

13. Abstraction allows designers to: _____.

a) Focus on low-level details
b) Ignore problem definition
c) Focus on problem solving without irrelevant details
d) Write detailed code

14. Procedural abstraction refers to: _____.

a) Named collection of data objects
b) Named sequence of events
c) Data hiding
d) Module hierarchy

15. Data abstraction refers to: _____.

a) Named sequence of operations
b) Named collection of data objects
c) Control structure
d) Program logic

16. Refinement is the process of: _____.

a) Removing modules
b) Testing software
c) Providing successively more detail
d) Debugging programs

17. Modularity means: _____.

a) Single large program
b) Dividing software into modules
c) Ignoring interfaces
d) Writing repeated code

18. Excessive modularization may: _____.

a) Reduce effort
b) Increase total effort
c) Improve performance
d) Eliminate errors

19. Software architecture describes: _____.

a) Coding techniques
b) Overall structure of software components
c) Test case design
d) Compiler behaviour

20. Control hierarchy represents: _____.

a) Data flow
b) Algorithm steps
c) Module organization and control
d) Code execution

21. Horizontal partitioning divides software into: _____.

a) Input, processing, storage
b) Input, data transformation, output
c) Design, coding, testing
d) User, system, hardware

22. A benefit of horizontal partitioning is: _____.

a) Increased side effects
b) Difficult maintenance
c) Easier testing
d) Higher coupling

23. A disadvantage of horizontal partitioning is: _____.

a) Reduced testing
b) More data passed across interfaces
c) Reduced extensibility
d) Less modularity

24. Vertical partitioning is also known as: _____.

a) Layering
b) Factoring
c) Abstraction
d) Refinement

25. In vertical partitioning, top-level modules perform: _____.

a) Input/output tasks
b) Computational work
c) Control functions
d) Storage operations

26. Low-level modules in vertical partitioning perform: _____.

a) Control decisions
b) Processing and I/O tasks
c) Interface management
d) Design validation

27. A software module is a: _____.

a) Data item
b) Named entity
c) Compiler unit
d) Test case

28. Modules can be: _____.

a) Executed only once
b) Separately compiled and stored
c) Used only internally
d) Hardware dependent

29. Fan-out refers to: _____.

a) Number of modules controlling a module
b) Number of modules controlled by another module
c) Depth of hierarchy
d) Width of hierarchy

30. Fan-in indicates: _____.

a) Width of program structure
b) Depth of control
c) Number of modules controlling a given module
d) Number of parameters

31. A controlling module is called: _____.

a) Subordinate
b) Child
c) Superordinate
d) Independent

32. Visibility in control hierarchy indicates: _____.

a) Direct invocation only
b) Indirect and direct access to components
c) Program execution speed
d) Memory usage

33. Connectivity refers to: _____.

a) Indirect usage
b) Direct invocation of components
c) Program size
d) Interface complexity

34. Data structure represents: _____.

a) Algorithm flow
b) Logical relationship among data elements
c) Program control
d) Module hierarchy

35. A scalar item represents: _____.

a) Multiple data elements
b) A single data element
c) Linked data
d) Multidimensional data

36. An array is: _____.

a) A scalar item
b) A linked list
c) An n-dimensional data structure
d) A single vector

37. A linked list organizes data: _____.

a) Contiguously
b) Sequentially only
c) Non-contiguously
d) Randomly

38. Software procedure focuses on: _____.

a) Data modeling
b) Processing details of each module
c) Architectural layout
d) User interface

39. Software procedure must specify: _____.

a) Only algorithms
b) Only data structures
c) Sequence, decisions, repetitions, and data organization
d) Only control flow

40. Procedural design occurs after: _____.

a) Coding
b) Testing
c) Data, architectural, and interface design
d) Maintenance

41. The fundamental goal of software design is to _____.
a) Increase module size
b) Increase interconnections
c) Minimize complexity and interconnections
d) Maximize control flow

42. **Which concept measures the** functional strength of a module**?**
a) Coupling
b) Cohesion
c) Fan-out
d) Abstraction

43. **A module that performs a** single well-defined task **exhibits**
a) Logical cohesion
b) Temporal cohesion
c) Functional cohesion
d) Procedural cohesion

44. **Cohesion is considered best when it is _____.**
a) Low
b) Medium
c) Moderate
d) High

45. **Which of the following represents the** lowest level of cohesion**?**
a) Functional
b) Sequential
c) Communicational
d) Coincidental

46. **A module where tasks are related by** time of execution **exhibits _____.**
a) Logical cohesion
b) Temporal cohesion
c) Procedural cohesion
d) Sequential cohesion

47. **When module elements must be executed in a** specific order**, it is _____.**
a) Procedural cohesion
b) Logical cohesion
c) Sequential cohesion
d) Communicational cohesion

48. **Coupling refers to _____.**
a) Functional strength of a module
b) Complexity of algorithms
c) Interconnection between modules
d) Program size

49. **The best form of coupling is _____.**
a) Content coupling
b) Control coupling
c) Data coupling
d) No direct coupling

50. Which coupling should be strictly avoided**?**
a) Data coupling
b) Stamp coupling
c) Content coupling
d) External coupling

51. Data design is conducted during _____.
a) Testing phase
b) Coding phase
c) Design phase
d) Maintenance phase

52. Data design has a profound influence **on _____.**
a) Hardware cost
b) Software quality
c) User interface
d) Network speed

53. The foundation of data design is _____.
a) Modular programming
b) Coupling and cohesion
c) Information hiding and data abstraction
d) Flow control

54. According to Wasserman, low-level data decisions should be _____.
a) Made early
b) Avoided
c) Deferred
d) Ignored

55. Representation of data structures should be known only to _____.
a) All modules
b) Testing modules
c) Related modules
d) Modules that directly use the data

56. Functional independence is achieved by _____.
a) High coupling and low cohesion
b) Low cohesion and low coupling
c) High cohesion and low coupling
d) High coupling and high cohesion

57. Fan-out refers to _____.
a) Number of modules calling a module
b) Depth of program
c) Number of modules controlled by a module
d) Interface size

58. A structure with high fan-out **indicates _____.**
a) Good factoring
b) Poor modularity
c) High cohesion
d) High fan-in

59. A predictable module can be treated as _____.
a) White box
b) Black box
c) Gray box
d) Control box

60. Branching into the middle of a module is called _____.
a) Modular entry
b) Controlled entry
c) Pathological connection
d) Fan-in

61. Which is NOT a design notation?
a) GDN
b) TDN
c) PDL
d) DDL

62. Structured programming consists of _____.
a) Sequence, Selection, Loop
b) Sequence, Condition, Repetition
c) Condition, Loop, Jump
d) Input, Process, Output

63. Flowcharts use which symbol for decision making?
a) Circle
b) Rectangle
c) Diamond
d) Oval

64. Which diagram shows information flow and transformations**?**
a) Flowchart
b) Box diagram
c) Data Flow Diagram
d) Decision table

65. Decision tables are mainly used for _____.
a) Algorithm design
b) Condition–action representation
c) Code generation
d) Data storage

66. Top-down design begins with _____.
a) Coding
b) Implementation details
c) Requirement analysis
d) Testing

67. Bottom-up design starts with _____.
a) Abstract model
b) Concrete machine
c) User interface
d) Testing

68. Object-oriented decomposition focuses on _____.
a) Functions
b) Data only
c) Objects with data and operations
d) Algorithms

69. Information hiding principle was proposed by _____.
a) Booch
b) Dijkstra
c) Parnas
d) Rumbaugh

70. Design documentation ensures _____.
a) Faster execution
b) Requirement traceability
c) Reduced memory
d) Compiler optimization

SOFTWARE IMPLEMENTATION

1. **The implementation phase of software development mainly focuses on _____.**
a) Requirement analysis
b) Translating design into source code
c) System testing
d) Software maintenance

2. **The primary goal of implementation is to _____.**
a) Reduce execution time
b) Increase program size
c) Write clear source code and documentation
d) Eliminate all errors

3. **Which quality makes debugging, testing, and modification easier?**
a) Clever coding
b) Compact coding
c) Clear and straightforward code
d) Complex algorithms

4. **Which of the following are hallmarks of good programs?**
a) Complexity and speed
b) Simplicity, clarity, and elegance
c) Cleverness and compactness
d) Extensive branching

5. **Unexplained cleverness in code indicates _____.**

a) Good optimization
b) Adequate design
c) Misguided thinking
d) High performance

6. **The goal of structured coding is to _____.**
a) Improve compiler speed
b) Achieve linear flow of control
c) Reduce memory usage
d) Increase program length

7. **Structured coding follows which control flow principle?**
a) Multiple entry, multiple exit
b) Single entry, multiple exit
c) Single entry, single exit
d) Random entry, random exit

8. Excessive use of which statement violates structured coding?
a) IF
b) FOR
c) SWITCH
d) GOTO

9. Strict adherence to single-entry single-exit constructs may raise concerns about ……
a) Readability
b) Efficiency
c) Maintainability
d) Documentation

10. Programming languages are vehicles for communication between _____.
a) Systems and networks
b) Software and hardware
c) Humans and computers
d) Compilers and interpreters

11. Psychological characteristics of a programming language mainly affect _____.
a) Execution speed
b) Communication quality
c) Memory usage
d) Compiler efficiency

12. Which characteristic indicates consistent notation and lack of arbitrary rules?
a) Ambiguity
b) Compactness
c) Uniformity
d) Tradition

13. Ambiguity in a programming language means _____.
a) Compiler confusion
b) Multiple compiler interpretations
c) Human misinterpretation of statements
d) Syntax errors

14. Compactness refers to _____.
a) Program execution speed
b) Amount of code to remember
c) Memory consumption
d) Code portability

15. Locality is related to _____.
a) Sequential memory
b) Synthetic memory
c) External memory
d) Cache memory

16. Extensive branching violates _____.
a) Uniformity
b) Locality
c) Linearity
d) Compactness

17. The concept of tradition helps programmers Software Implementation

a) Avoid errors
b) Learn new languages faster
c) Improve efficiency
d) Reduce memory

18. Semantic knowledge is _____.
a) Language dependent
b) Easier to acquire
c) Conceptual and design oriented
d) Instructional

19. Syntactic knowledge is applied mainly during _____.
a) Requirements analysis
b) Design
c) Coding
d) Testing

20. Confusion may arise when _____.
a) Semantic rules differ
b) Syntax is similar but not equivalent
c) Language is compact
d) Language is uniform

21. Which is NOT an engineering characteristic of programming languages?
a) Compiler efficiency
b) Source code portability
c) Ambiguity
d) Maintainability

22. Ease of design-to-code translation depends on _____.
a) Processor speed
b) Language support for structured constructs
c) Memory size
d) Operating system

23. A good compiler should provide _____.
a) Only fast execution
b) Only syntax checking
c) Efficient code and debugging support
d) Only memory optimization

24. Source code portability means _____.
a) Code runs faster on all machines
b) Code needs no compiler
c) Code can move across platforms with little change
d) Code size is small

25. Availability of development tools helps in _____.
a) Increasing complexity
b) Reducing code quality
c) Shortening development time
d) Increasing memory usage

26. Company policy may influence language choice mainly due to _____.
a) Syntax rules
b) Uniformity and maintenance
c) Compiler errors
d) Programmer preference

27. Maintainability of software largely depends on _____.
a) Hardware
b) Documentation only
c) Readable source code
d) Compiler speed

28. Readability of a program depends more on _____.
a) Programming language
b) Compiler
c) Programming style
d) Hardware

29. Coding style emphasizes _____.
a) Speed and memory
b) Complexity
c) Simplicity and clarity
d) Clever tricks

30. Which is NOT an element of coding style?
a) Structuredness
b) Expressiveness
c) Data declaration
d) Compilation

31. Structuring in the large deals with _____.
a) Algorithms
b) Classes, modules, and methods
c) Statements
d) Expressions

32. Structuring in the small focuses on _____.
a) System architecture
b) Algorithm readability
c) Compiler design
d) Database design

33. Structured programming uses which control elements?
a) GOTO, CALL, RETURN
b) Sequence, branch, loop
c) IF, GOTO, EXIT
d) SWITCH, BREAK, CONTINUE

34. Expressive identifiers should be _____.
a) Short and cryptic
b) Long but meaningful
c) Numeric
d) Abbreviated

35. Which improves program readability the most?
a) Removing comments
b) Using clever logic
c) Proper naming and comments
d) Compact code

36. Incorrect comments are _____.
a) Acceptable
b) Better than no comments
c) Worse than no comments
d) Ignorable

37. Alphabetical ordering of variables helps in _____.

a) Compilation
b) Debugging and maintenance
c) Execution speed
d) Memory allocation

38. Which should be avoided in statement construction?
a) Parentheses
b) Simple statements
c) Heavy nesting
d) Clear logic

39. I/O style is established during _____.
a) Coding
b) Testing
c) Design
d) Maintenance

40. Efficiency must be considered only after _____.
a) Optimization
b) Understanding the problem and solution
c) Coding
d) Testing

41. Which is NOT an efficiency type?
a) Code efficiency
b) Memory efficiency
c) Algorithm efficiency
d) I/O efficiency

42. Buffering I/O improves _____.
a) Code readability
b) Communication overhead
c) I/O efficiency
d) Maintainability

43. Coding standards mainly specify _____.
A. Algorithms to be used
B. Hardware requirements
C. Preferred coding style
D. Testing techniques

44. Coding standards are often viewed by programmers as _____.
A. Enhancing creativity
B. Increasing execution speed
C. Constraining creative problem solving
D. Eliminating errors completely

45. The main purpose of adopting common coding styles is to_____.
A. Reduce program size
B. Improve execution time
C. Produce code of uniform quality
D. Avoid documentation

46. Uniform coding style means_____.
A. Programmers must think alike
B. Algorithms must be identical
C. Individual creativity is eliminated
D. Code quality remains consistent

47. Which of the following is a typical programming standard?
A. Use goto statements freely
B. Nesting depth should not exceed five levels
C. Subprograms may exceed any length
D. Avoid documentation

48. A guideline differs from a standard because it _____.
A. Is mandatory
B. Allows flexibility
C. Is enforced by compiler
D. Is language dependent

49. "Goto statements should be avoided in normal circumstances" is an example of .
A. Coding standard
B. Algorithm rule
C. Programming guideline
D. Design constraint

50. Departure from normal coding circumstances requires approval from
A. Compiler
B. Tester
C. System analyst
D. Project leader

51. Computer software includes
A. Only source code
B. Only user manuals
C. Source code and supporting documents
D. Only executable files

52. Internal documentation includes _____.
A. User manuals
B. Test plans
C. Source code comments
D. Installation guides

53. Supporting documents are generated during _____.
A. Only implementation
B. Only testing
C. Development and maintenance
D. Only design

54. Which of the following is NOT a supporting document?
A. Requirements specification
B. Design document
C. Test plan
D. Compiler error list

55. A program unit is developed and maintained by _____.
A. Multiple programmers
B. System analyst
C. One responsible programmer
D. Project manager

56. A well-designed program unit should be _____.
A. Large and complex
B. Difficult to test
C. Modular and testable
D. Dependent on other units

57. Program unit notebooks are used to _____.
A. Execute programs
B. Compile source code
C. Organize work and documentation
D. Replace user manuals

58. Internal documentation consists of _____.
A. Only external manuals
B. Comments and standard introductions
C. Only test cases
D. Installation notes

59. Implementation phase is concerned with translating _____.
A. Requirements into design
B. Design into code
C. Code into test cases
D. Test cases into product

60. Programming languages act as a vehicle for communication between _____.
A. User and system
B. Hardware and software
C. Programmer and computer
D. Manager and developer

61. _____ of source code is important for all nontrivial software development.
A. Execution
B. Compilation
C. Documentation
D. Optimization

62. Coding style refers to _____.
A. Only design
B. Only implementation
C. Both design and implementation
D. Only testing

63. The choice of programming language affects _____.
A. Only testing
B. Only design
C. Implementation of a project
D. Hardware selection

64. Ideally, design should be carried out _____.
A. After choosing language
B. With knowledge of language
C. Independent of implementation language
D. After coding

65. Which of the following is NOT a quality criterion of programming languages?
A. Modularity
B. Portability
C. Profitability
D. Efficiency

66. Modularity supports _____.
A. Monolithic programs
B. Division of labour
C. Increased compile time
D. Poor maintenance

67. Independent compilation is supported by languages like _____.
A. Ada
B. Modula-2
C. C
D. Eiffel

68. Separate compilation allows checking at _____.
A. Run time
B. Design time
C. Compile time
D. Maintenance time

69. High documentation value improves _____.
A. Execution speed
B. Readability and maintainability
C. Memory usage
D. Hardware dependency

70. Recursive data structures are difficult to implement in _____.
A. C
B. Object-oriented languages
C. FORTRAN and COBOL
D. Eiffel

71. Object-oriented languages support _____.
A. Only simple data
B. Extensible abstract data types
C. No modularity
D. Poor reusability

72. Elimination of GOTO statements forces _____.
A. Complex control flow
B. Better structured programs
C. Longer code
D. Slower execution

73. Static typing improves _____.
A. Execution time
B. Integrity
C. Portability
D. Dialog support

74. Portability mainly depends on _____.
A. Compiler availability
B. Programmer skill
C. Documentation style
D. Control flow

75. Languages designed for dialog support include _____.
A. COBOL
B. FORTRAN
C. BASIC
D. Assembly

76. Specialized language elements help in _____.
A. Reducing hardware cost
B. Solving domain-specific problems
C. Eliminating documentation
D. Increasing program size

77. A data type specifies _____.
A. Only a set of operations
B. Only a set of data values
C. A set of data objects and permitted operations
D. Only memory allocation

78. Objects of integer type include _____.
A. Only arithmetic operators
B. Only relational operators
C. A range of integer values and arithmetic/relational operators
D. Only literals

79. The range of integer values is _____.
A. Fixed for all languages
B. User defined
C. Implementation dependent
D. Unlimited

80. The main purpose of data typing is to _____.
A. Increase execution speed
B. Classify objects by intended usage
C. Reduce program length
D. Avoid documentation

81. Data typing helps the language translator to _____.
A. Select algorithms
B. Generate comments
C. Select storage representations
D. Optimize loops

82. Strongly typed languages aim to _____.
A. Allow all operations
B. Detect operations among different types
C. Prevent operations among different types
D. Eliminate type declarations

83. Type checking refers to _____.
A. Storage allocation
B. Syntax verification
C. Restrictions on data manipulation
D. Code optimization

84. Different languages impose different type restrictions due to _____.
A. Hardware limitations
B. Compiler speed
C. Language designer philosophies
D. User preferences

85. How many levels of type checking are distinguished?
A. Three
B. Four
C. Five
D. Six

86. Level 0 type checking is known as _____.
A. Strong typing
B. Mixed mode
C. Automatic coercion
D. Type less

87. Automatic type coercion belongs to _____.
A. Level 0
B. Level 1
C. Level 2
D. Level 3

88. Mixed mode type checking corresponds to _____.
A. Level 1
B. Level 2
C. Level 3
D. Level 4

89. The highest level of type checking is _____.
A. Automatic coercion
B. Mixed mode
C. Pseudo-strong
D. Strong type checking

90. A declaration associates an identifier with _____.
A. Only a variable
B. Only a type
C. A program entity
D. Only a subprogram

91. The region of source code where a declaration is effective is called _____.
A. Lifetime
B. Scope
C. Domain
D. Binding

92. Scoping rules determine _____.
A. Program execution speed
B. How identifiers are defined and used
C. Memory size
D. Control flow

93. Which of the following is NOT a scoping rule?
A. Global scope
B. FORTRAN scope
C. Nested scope
D. Dynamic scope (not mentioned)

94. Global scope is provided in _____.
A. C and Ada
B. BASIC and COBOL
C. Pascal and Modula-2
D. Eiffel and Smalltalk

95. In global scope _____.
A. Identifiers are local to modules
B. Identifiers are visible only in subprograms
C. Identifiers are known in all program regions
D. Identifiers must be redeclared

96. In FORTRAN scope, identifiers are known _____.
A. Only inside loops
B. Only inside functions
C. Throughout the containing program unit
D. Everywhere in the system

97. Identifiers in FORTRAN become visible outside the unit if they appear in _____.
A. DATA statements
B. COMMON statements
C. DO loops
D. FORMAT statements

98. Program segments executing independently are called _____.
A. Modules
B. Threads only
C. Tasks or processes
D. Subprograms

99. Concurrent execution means the result is independent of _____.
A. Input data
B. Execution order
C. Hardware
D. Programming language

100. On multiple-processor machines, concurrency achieves _____.
A. Reduced memory usage
B. Increased processing efficiency
C. Simplified coding
D. Better documentation

101. On a single-processor system, concurrency is achieved by _____.
A. Parallel execution
B. Interleaving execution
C. Multiple compilers
D. Multiple languages

102. A task may execute while another task is _____.
A. Terminated
B. Waiting for I/O
C. Deleted
D. Compiled

103 The trend toward multiprocessor systems has led to _____.
A. Lower-level languages
B. Removal of concurrency
C. Higher-level concurrency constructs
D. Elimination of tasks

104. One major problem in concurrent programming is _____.
A. Compilation
B. Debugging only
C. Synchronization
D. Documentation

105. Preventing simultaneous updating of shared data is called _____.
A. Synchronization problem
B. Deadlock problem
C. Mutual exclusion problem
D. Scheduling problem

106. Synchronization ensures _____.
A. Faster execution
B. Proper information transfer between tasks
C. Reduced memory usage
D. Compiler optimization

107. Which of the following is NOT a fundamental approach to concurrency?
A. Shared variables
B. Asynchronous message passing
C. Synchronous message passing
D. Recursive function calls

108. Communication using shared memory is called _____.
A. Message passing
B. Shared variables
C. Signal handling
D. Task scheduling

109. In synchronous message passing _____.
A. Sender and receiver do not wait
B. Communication occurs independently
C. Sender and receiver synchronize
D. No data is exchanged

110. Asynchronous message passing means _____.
A. Sender waits for receiver
B. Receiver waits for sender
C. Communication without waiting
D. No concurrency

SOFTWARE QUALITY ASSURANCE
SOFTWARE TESTING
SOFTWARE MAINTENANCE

1. According to the American Heritage Dictionary, quality is defined as _____.
A. Fitness for use
B. Degree of excellence
C. A characteristic or attribute of something
D. Conformance to standards

2. Quality based on measurable characteristics is classified into _____.
A. Functional and non-functional quality
B. Internal and external quality
C. Quality of design and quality of conformance
D. Product and process quality

3. Quality of design in software development includes _____.
A. Coding and testing
B. Requirements, specifications and system design
C. Implementation and maintenance
D. Debugging and documentation

4. Quality of conformance mainly focuses on _____.
A. Requirement analysis
B. Design documentation
C. Implementation following design
D. User training

5. Conformance quality is considered high when _____.
A. Design is innovative
B. Code is complex
C. Implementation meets requirements and performance goals
D. Software is released early

6. Quality Assurance is defined as _____.
A. Testing the final product
B. Debugging errors
C. Planned and systematic actions to ensure conformance
D. Improving program efficiency

7. QA also includes the _____functions of management.
A. Development
B. Auditing and reporting
C. Coding
D. Debugging

8. Which of the following is NOT a need for Quality Assurance?
A. Humans are error-free
B. Cost of fixing errors increases with time
C. Customers should not find bugs
D. Systems software is not bug free

9. Post-release debugging is _____.
A. Cheaper
B. Optional
C. More expensive
D. Unnecessary

10. The primary goal of QA is to _____.
A. Detect errors only
B. Provide confidence about product quality
C. Replace testing
D. Reduce documentation

11. If QA data identifies problems, responsibility to resolve lies with _____.
A. Programmer
B. Tester
C. Management
D. Customer

12. Cost of quality refers to _____.
A. Cost of testing alone
B. Cost of fixing bugs
C. Total cost of achieving and not achieving quality
D. Cost of documentation

13. Quality costs are divided into _____.
A. Development and maintenance
B. Prevention, appraisal and failure
C. Internal and external
D. Testing and debugging

14. Quality planning and training belong to _____.
A. Appraisal cost
B. Failure cost
C. Prevention cost
D. External cost

15. In-process inspection and testing are examples of
A. Prevention cost
B. Appraisal cost
C. Internal failure cost
D. External failure cost

16. Costs that disappear if no defects exist are called _____.
A. Prevention costs
B. Appraisal costs
C. Failure costs
D. Maintenance costs

17. Rework and repair are classified as _____.
A. External failure cost
B. Internal failure cost
C. Prevention cost
D. Appraisal cost

18. Warranty work and complaint resolution are _____.
A. Internal failure costs
B. Appraisal costs
C. External failure costs
D. Prevention costs

19. Competitive pressure forces companies to _____.
A. Reduce staff
B. Ignore quality
C. Improve cost and quality
D. Focus only on speed

20. Retaining customers is _____than acquiring new customers.
A. More expensive
B. Equally expensive
C. Less expensive
D. Impossible

21. Defects in mission-critical systems may cause _____.
A. Minor inconvenience
B. Documentation errors
C. Serious harm to users
D. No impact

22. Software quality attributes are standardized by _____.
A. Microsoft only
B. ISO, ANSI, IEEE
C. Google
D. Universities

23. Correctness refers to _____.
A. Code efficiency
B. Meeting user mission and specifications
C. Ease of use
D. Portability

24. Reliability of software depends on
A. Performance only
B. Correctness and availability
C. Efficiency and portability
D. Maintainability

25. Reliability is defined as the probability that software _____.
A. Runs fast
B. Is bug-free
C. Performs specified functions over time
D. Is easy to modify

26. A low error rate indicates _____.
A. Low efficiency
B. High reliability
C. Poor design
D. High complexity

27. User friendliness includes _____.
A. Adequacy, learnability, robustness
B. Efficiency, portability, reliability
C. Modularity, testability
D. Performance only

28. Adequacy ensures that _____.
A. Input is complex
B. Only necessary input is requested
C. More features are added
D. Output is hidden

29. Error messages should be _____.
A. Technical
B. Lengthy
C. User-comprehensible
D. Ignored

30. Learnability mainly depends on
A. Hardware
B. User interface and manuals
C. Compiler
D. Memory

31. Robustness deals with _____.
A. Speed
B. Handling errors and failures
C. Porting
D. Code size

32. Maintainability refers to effort required to _____.
A. Compile code
B. Fix errors or add features
C. Execute program
D. Test hardware

33. Maintainability depends on _____.
A. Readability, extensibility, testability
B. Speed and memory
C. Portability only
D. Reliability only

34. Readability is influenced by _____.
A. Programming style and documentation
B. CPU speed
C. Compiler version
D. Network speed

35. Extensibility allows _____.
A. Easy execution
B. Changes without side effects
C. Faster compilation
D. Reduced documentation

36. Testability is improved by _____.
A. Monolithic design
B. Modularity and structure
C. Removing documentation
D. Using GOTO

37. Efficiency refers to optimal use of _____.
A. Time only
B. Storage only
C. All system resources
D. Documentation

38. Portability is the ease of _____.
A. Execution
B. Modification
C. Running on different systems
D. Debugging

39. A system is portable if _____.
A. It uses machine-specific features
B. Porting cost is high
C. Porting effort is less than new development
D. It uses assembly language

40. SQA refers to _____.
A. Testing only
B. Umbrella activities for ensuring quality
C. Coding standards
D. Debugging tools

41. SQA focuses on _____.
A. Fault detection only
B. Prevention and excellence
C. Correction only
D. Speed

42. Software requirements are the foundation of _____.
A. Coding
B. Quality measurement
C. Testing tools
D. Documentation

43. Failure to meet implicit requirements makes quality _____.
A. Perfect
B. Acceptable
C. Suspect
D. Excellent

44. The SQA group is responsible for _____.
A. Coding
B. Planning, analysis and reporting
C. Debugging
D. Marketing

45. SQA Plan identifies _____.
A. Coding syntax
B. Audits, reviews and standards
C. Compiler settings
D. Hardware

46. Recording non-compliance is reported to _____.
A. Testers
B. Programmers
C. Senior management
D. Customers

47. Walkthroughs are used to check _____.
A. Execution speed
B. Correctness of models
C. Memory usage
D. Compiler errors

48. Walkthrough team usually consists of _____.
A. One person
B. Three to five reviewers
C. Ten members
D. Only manager

49. Software reviews act as a _____.
A. Debugger
B. Filter for the software process
C. Compiler
D. Scheduler

50. FTR is a _____activity.
A. Testing
B. Management
C. Quality assurance
D. Coding

51. FTR aims to uncover errors in _____.
A. Logic, function and implementation
B. Hardware
C. Documentation only
D. Network

52. FTR meeting duration should be _____.
A. Unlimited
B. Less than two hours
C. One full day
D. Five hours

53. The developer of the work product is called _____.
A. Reviewer
B. Recorder
C. Producer
D. Moderator

54. Inspections are more _____than walkthroughs.
A. Informal
B. Unstructured
C. Formal
D. Casual

55. Fagan inspection consists of how many steps?
A. Three
B. Four
C. Five
D. Six

56. Static analysis is performed _____.
A. By executing code
B. Without executing code
C. After deployment
D. By users

57. Code inspection aims to _____.
A. Correct errors
B. Detect errors
C. Optimize speed
D. Add features

58. Complexity analysis helps to identify _____.
A. Syntax errors
B. Error-prone areas
C. Hardware faults
D. Network issues

59. Data-flow analysis checks _____.
A. Code execution speed
B. Use of data before assignment
C. UI design
D. Compilation time

60. Static analysis focuses on _____.
A. Structural characteristics
B. User feedback
C. Runtime performance
D. Hardware testing

61. According to Glen Meyers, testing is the process of: _____.
A. Debugging a program
B. Executing a program to find errors
C. Reviewing the code
D. Improving performance

62. A successful test case is one that: _____.
A. Produces correct output
B. Finds no errors
C. Finds previously undiscovered errors
D. Runs faster

63. Which of the following testing objectives is correct?
A. Testing proves absence of defects
B. Testing guarantees correctness
C. Testing uncovers errors
D. Testing replaces debugging

64. Testing cannot**: _____.**
A. Uncover defects
B. Demonstrate conformance to specifications
C. Prove absence of errors
D. Indicate software reliability

65. According to Davis, tests should be: _____.
A. Designed after coding
B. Planned before coding
C. Created during debugging
D. Written by customers

66. The Pareto principle in software testing states that: _____.
A. 80% testing is enough
B. 20% modules contain 80% defects
C. All modules have equal defects
D. Errors are random

67. "Testing should begin in the small and progress towards testing in the large" means:
A. Test simple inputs first
B. Start with unit testing and move to system testing
C. Test small projects only
D. Use few test cases

68. Exhaustive testing is not possible because: _____.
A. Test cases are costly
B. Programs are complex
C. Number of paths is very large
D. Test tools are limited

69. Testing is most effective when performed by: _____.
A. Programmer
B. Project manager
C. Independent third party
D. Customer

70. A good test case should be: _____.
A. Redundant
B. Very complex
C. Non-redundant
D. Random

71. Which is NOT a characteristic of a good test?
A. High probability of finding errors
B. Redundancy
C. Best of its kind
D. Balanced complexity

72 Which of the following is tested in software testing?
A. Only code
B. Only design
C. Modules and their integration
D. Hardware only

73. Dynamic testing involves: _____.
A. Code inspection
B. Executing the program
C. Design review
D. Requirement analysis

74. Dynamic testing is performed: _____.
A. Only after deployment
B. Only during design
C. Throughout the software life cycle
D. Only during coding

75. Which is NOT an activity of dynamic testing?
A. Test case selection
B. Test execution
C. Test evaluation
D. Requirement specification

76. Reliability, portability, maintainability, and usability are: _____.
A. Functional requirements
B. Quality attributes
C. Testing tools
D. Programming constructs

77. Unit testing is mainly done to validate: _____.
A. Requirements
B. Design
C. Code
D. Customer acceptance

78. Integration testing validates: _____.
A. Coding errors
B. Design strategies
C. User interface
D. Documentation

79. System testing ensures that: _____.
A. Individual modules work
B. Design is correct
C. Functional and non-functional requirements are met
D. Code is optimized

80. Acceptance testing is performed by: _____.
A. Programmer
B. Tester
C. Project manager
D. Customer

81. Unit testing is usually: _____.
A. Black box oriented
B. White box oriented
C. User oriented
D. Requirement oriented

82. Unit testing is simplified when modules have: _____.
A. Low cohesion
B. High coupling
C. High cohesion
D. No interfaces

83. Boundary condition testing checks: _____.
A. Error messages
B. Performance
C. Extreme input values
D. Output formatting

84. Independent path testing ensures that: _____.
A. All modules are integrated
B. All statements execute at least once
C. All errors are fixed
D. All data is validated

85. A driver in unit testing is: _____.
A. Subordinate module
B. Main program that calls the module
C. Error handler
D. Test environment

86. A stub is used to: _____.
A. Replace higher-level modules
B. Replace lower-level modules
C. Replace drivers
D. Replace test data

87. Bottom-up integration testing starts from: _____.
A. Control module
B. Top level
C. Lowest level modules
D. User interface

88. In top-down integration testing, unavailable modules are replaced by: _____.
A. Drivers
B. Compilers
C. Stubs (surrogates)
D. Test cases

89. An advantage of top-down testing is: _____.
A. Late detection of design errors
B. Early detection of design errors
C. Easy error localization
D. No need for stubs

90. A disadvantage of bottom-up testing is: _____.
A. Need for stubs
B. Design errors detected late
C. Difficult test case design
D. Lack of detail

91. Sandwich testing combines: _____.
A. Unit and system testing
B. Black box and white box testing
C. Top-down and bottom-up testing
D. Manual and automated testing

92. Big-Bang integration testing is: _____.
A. Highly recommended
B. Easy to debug
C. Risky and not feasible
D. Cost effective

93. Big-Bang testing is difficult because: _____.
A. Modules are small
B. Interfaces are clear
C. Error localization is difficult
D. Testing is slow

94. System testing is conducted on: _____.
A. Individual modules
B. Integrated subsystems
C. Complete integrated system
D. Source code

95. System testing mainly evaluates: _____.
A. Coding efficiency
B. Compliance with specified requirements
C. Algorithm correctness
D. Program syntax

96. System testing belongs to: _____.
A. White box testing
B. Black box testing
C. Gray box testing
D. Structural testing

97. System testing requires knowledge of: _____.
A. Internal logic
B. Source code
C. System specifications
D. Data structures

98. System testing is performed: _____.
A. Only by developers
B. Only by testers
C. By multiple system element developers
D. Only by customers

99. Alpha testing is conducted: _____.
A. In real user environment
B. In developer's environment
C. Without developer involvement
D. After release

100. Alpha testing is carried out by: _____.
A. Developers
B. End users
C. Customers
D. Project managers

101. Beta testing is conducted: _____.
A. In a controlled environment
B. In developer's lab
C. In end-user environment
D. Before alpha testing

102. Beta testing environment is: _____.
A. Simulated
B. Controlled
C. Real-world
D. Virtual

103. The main purpose of beta testing is to: _____.
A. Detect syntax errors
B. Improve code structure
C. Identify defects in real usage
D. Test algorithms

104. "Finger pointing" refers to: _____.
A. User blaming developer
B. Tester blaming user
C. Developers blaming each other
D. System crash

105. Finger pointing occurs mainly during: _____.
A. Unit testing
B. Integration testing
C. System testing
D. Debugging

106. To avoid finger pointing, software engineers should: _____.
A. Ignore interface issues
B. Design error-handling paths
C. Skip system testing
D. Delay testing

107. Recovery testing checks: _____.
A. Speed of execution
B. Data encryption
C. Ability to recover from failures
D. User interface

108. Security testing verifies: _____.
A. Data accuracy
B. Protection against unauthorized access
C. Response time
D. System recovery

109. Stress testing evaluates system behaviour under: _____.
A. Normal load
B. Minimum load
C. Abnormal resource demand
D. Idle condition

110. Sensitivity testing focuses on: _____.
A. Invalid data only
B. Small variations in valid data
C. Maximum memory usage
D. Hardware failures

111. Performance testing measures: _____.
A. Code readability
B. Run-time speed and response
C. Logic correctness
D. Data accuracy

112. Load testing determines: _____.
A. Maximum failure point
B. System reliability
C. Ability to handle expected load
D. Security level

113. Endurance testing checks: _____.
A. Peak load
B. Short-term stress
C. Long-term stability
D. Interface correctness

114. Spike testing evaluates: _____.
A. Gradual load increase
B. Sudden load increase
C. Continuous load
D. Minimal load

115. Regression testing ensures that: _____.
A. New features work
B. Old functionality remains unaffected
C. Performance improves
D. Code is optimized

116. Regression testing is especially important in: _____.
A. New projects
B. Maintenance projects
C. Prototype development
D. Requirement analysis

117. Acceptance testing is conducted by: _____.
A. Developers
B. Testers
C. Customers
D. System analysts

118. Acceptance testing validates: _____.
A. Internal logic
B. Coding standards
C. Customer requirements
D. Design patterns

119. White box testing is also called: _____.
A. Functional testing
B. Structural testing
C. Acceptance testing
D. Performance testing

120. White box testing requires knowledge of : _____.
A. User requirements
B. Program structure
C. Test data
D. User interface

121. White box testing ensures: _____.
A. Only output correctness
B. Only input correctness
C. Internal logic correctness
D. UI consistency

122. Cyclomatic complexity is used in: _____.
A. Boundary testing
B. Basis path testing
C. Load testing
D. Regression testing

123. Cyclomatic complexity formula is: _____.
A. N – E + 2
B. E + N – 2
C. E – N + 2
D. N + E + 2

124. Basis path testing guarantees: _____.
A. All paths executed
B. All statements executed at least once
C. No errors
D. Maximum performance

125. Condition testing focuses on: _____.
A. Loops
B. Logical expressions
C. Data flow
D. Interfaces

126. Data flow testing is based on: _____.
A. Branch coverage
B. Definition-use chains
C. Input range
D. Output comparison

127. Loop testing focuses on: _____.
A. Decision statements
B. Function calls
C. Loop constructs
D. Data types

128. Black box testing is also known as: _____.
A. Structural testing
B. Functional testing
C. Unit testing
D. Integration testing

129. Black box testing is based on: _____.
A. Code
B. Control flow
C. Requirements specification
D. Algorithms

130. Which error is detected by black box testing?
A. Logic error
B. Interface error
C. Typographical error
D. Syntax error

131. Equivalence partitioning aims to: _____.
A. Test all inputs
B. Minimize test cases
C. Maximize errors
D. Test boundaries only

132. Boundary Value Analysis tests: _____.
A. Random inputs
B. Middle values
C. Boundary values
D. Invalid values only

133. Smoke testing is: _____.
A. Exhaustive testing
B. Detailed testing
C. Preliminary testing
D. Performance testing

134. Debugging is primarily done to: _____.
A. Improve performance
B. Find and fix defects
C. Test requirements
D. Validate design

135. Debugging is mainly performed by: _____.
A. Tester
B. Customer
C. Programmer
D. Manager

136. First step in debugging is: _____.
A. Fix the bug
B. Recognize bug existence
C. Run compiler
D. Rewrite code

137. Which debugging approach is least efficient?
A. Backtracking
B. Cause elimination
C. Brute force
D. Logical analysis

138. Backtracking approach is suitable for: _____.
A. Large programs
B. Distributed systems
C. Small programs
D. Real-time systems

139. Cause elimination approach uses: _____.
A. Trial and error
B. Induction and deduction
C. Memory dumps
D. Random testing

140. A bug symptom may: _____.
A. Always reveal cause
B. Never change
C. Disappear temporarily
D. Be easily traceable

141. Debugging outcome can be: _____.

A. Bug always fixed
B. Cause may or may not be found
C. Program rewritten
D. Testing replaced

142. Software maintenance accounts for approximately: _____.
A. 10–20% of total development effort
B. 20–40% of total development effort
C. 50–80% of total development effort
D. 90–100% of total development effort

143. Software maintenance mainly involves: _____.
A. Writing new programs
B. Fixing only errors
C. Post-release modifications
D. Requirement analysis

144. Only about ___of maintenance effort is spent fixing mistakes.
A. 10%
B. 20%
C. 50%
D. 80%

145. The remaining 80% of maintenance effort is spent on: _____.
A. Debugging only
B. Code rewriting
C. Adaptation, enhancement, and reengineering
D. Testing new software

146. Corrective maintenance refers to: _____.
A. Enhancing performance
B. Correcting undiscovered errors
C. Adapting to new environments
D. Preventing future failures

147. Adaptive maintenance is performed due to: _____.
A. User dissatisfaction
B. Software failure
C. Environmental changes
D. Poor documentation

148. Perfective maintenance focuses on: _____.
A. Bug fixing
B. Environment changes
C. Improving the system without changing functionality
D. Code refactoring only

149. Preventive maintenance is also known as: _____.
A. Perfective maintenance
B. Corrective maintenance
C. Adaptive maintenance
D. Reengineering

150. Structured maintenance is possible when: _____.
A. Only source code exists
B. Documentation is missing
C. Complete software configuration exists
D. No test records exist

151. Unstructured maintenance usually starts with: _____.
A. Design evaluation
B. Requirement analysis
C. Source code evaluation
D. Test execution

152. Regression testing is difficult in unstructured maintenance because: _____.
A. Code is complex
B. No testing records exist
C. Users are unavailable
D. Tools are expensive

153. One intangible cost of software maintenance is: _____.
A. Increased revenue
B. Customer satisfaction
C. Lost development opportunities
D. Faster delivery

154. Which is NOT an intangible cost of maintenance?
A. Customer dissatisfaction
B. Reduction in software quality
C. Staff disruption
D. Compiler errors

155. The maintenance effort model is: _____.
A. M = p + K(c + d)
B. M = p – K(c – d)
C. M = p + Ke(c – d)
D. M = p – Ke(c + d)

156. In the maintenance effort equation, 'C' represents: _____.
A. Code size
B. Complexity due to poor design
C. Cost factor
D. Control flow

157. Difficulty in tracing software evolution is mainly due to: _____.
A. Hardware changes
B. Poor documentation
C. User errors
D. Compiler limitations

158. Maintenance becomes difficult when: _____.
A. Software is modular
B. Developers are available
C. Software is not designed for change
D. SCM tools are used

159. One major reason maintenance has a poor image is: _____.
A. High salary
B. Low workload
C. High frustration level
D. Automation tools

160. Maintainability is defined as: _____.
A. Ease of testing software
B. Ease of modifying software
C. Ease of understanding, correcting, adapting, and enhancing software
D. Ease of coding

161. Which activity improves maintainability during analysis?
A. Code indentation
B. Developing standards and guidelines
C. Loop optimization
D. Debugging

162. Emphasizing clarity and modularity belongs to: _____.
A. Analysis activities
B. Architectural design activities
C. Implementation activities
D. Testing activities

163. Use of symbolic constants is recommended during: _____.
A. Analysis
B. Design
C. Implementation
D. Maintenance

164. Maintenance activity usually begins with: _____.
A. Design review
B. Code inspection
C. Change request
D. Testing

165. The biggest challenge in software maintenance is: _____.
A. Coding
B. Testing
C. Managing maintenance staff
D. Documentation

166. Involving maintenance staff early helps to: _____.
A. Increase workload
B. Improve motivation
C. Reduce documentation
D. Delay development

167. SCM is mainly concerned with: _____.
A. Code generation
B. Change control
C. Performance testing
D. Debugging

168. SCM activities start when: _____.
A. Testing begins
B. Coding begins
C. Project begins
D. Product is released

169. Which is NOT a goal of SCM?
A. Configuration identification
B. Status accounting
C. Code optimization
D. Build management

170. Configuration identification answers: _____.
A. Who made the change?
B. Why was the change made?
C. What code are we working with?
D. When was the change made?

171. A configuration item is: _____.
A. A test case
B. A unit that can be versioned
C. A user requirement
D. A design document only

172. Maintainability metrics assume that maintainability is related to: _____.
A. Cost
B. Size
C. Complexity
D. Speed

173. Halstead's effort metric considers: _____.
A. Control flow only
B. Number of operators and operands
C. Lines of code
D. Execution time

174. McCabe's cyclomatic complexity is based on: _____.
A. Program size
B. Decision structure
C. Number of variables
D. Memory usage

175. Cyclomatic complexity is computed using: _____.
A. LOC
B. Operators and operands
C. Graph nodes and edges
D. CPU cycles

176. The law of continuing change states that: _____.
A. Software never changes
B. Software must change or become less useful
C. Software complexity decreases
D. Maintenance is optional

177. Increasing complexity occurs unless: _____.
A. More programmers are added
B. Active restructuring is done
C. More testing is done
D. SCM tools are used

178. The law of organizational stability states that: _____.
A. Development rate is variable
B. Resources control evolution
C. Development rate remains constant
D. Errors decrease over time

179. CASE stands for: _____.
A. Computer Aided Software Engineering
B. Control and Software Environment
C. Computer Assisted System Evaluation
D. Coding and Software Execution

180. CASE tools mainly help to: _____.
A. Replace engineers
B. Automate manual activities
C. Eliminate testing
D. Reduce documentation

181. Which is NOT a building block of CASE?
A. Environment architecture
B. Hardware platform
C. CASE tools
D. Compiler optimization

182. Risk analysis tools help in: _____.
A. Coding
B. Risk identification and analysis
C. Testing
D. Debugging

183. Requirements tracing tools help to: _____.
A. Generate code
B. Track requirements to implementation
C. Optimize performance
D. Debug errors

184. SCM tools support: _____.
A. Only version control
B. Identification and change control
C. Only testing
D. Only coding

185. Static analysis tools: _____.
A. Execute the program
B. Analyse code without execution
C. Require hardware probes
D. Modify source code

186. Dynamic analysis tools: _____.
A. Work without program execution
B. Interact with executing programs
C. Replace test cases
D. Eliminate bugs automatically

187. Intrusive testing tools: _____.
A. Do not modify code
B. Insert probes into software
C. Use external processors
D. Avoid execution

188. Reengineering tools perform: _____.
A. Forward engineering
B. Reverse engineering and restructuring
C. Debugging only
D. Testing only

189. Reverse engineering tools: _____.
A. Generate source code
B. Generate design models from code
C. Optimize performance
D. Compile programs

Conclusion

Software Engineering plays a crucial role in the development of reliable, efficient, and high-quality software systems that support modern technological advancements. Understanding its core principles is essential for students who aspire to build successful careers in the software and information technology industries.

This book has presented key concepts of Software Engineering through a structured collection of multiple-choice questions covering important topics such as Software Project Planning, Cost Estimation, Requirement Specification, Software Design, Software Implementation, and Software Quality Assurance. By engaging with these questions, students can strengthen their conceptual understanding and improve their ability to analyse and apply Software Engineering principles.

The MCQs included in this book are intended to support **learning, self-assessment, and examination preparation**. They provide students with opportunities to test their knowledge, identify areas that require further study, and build confidence in their understanding of the subject.

It is hoped that this book will serve as a valuable academic resource for undergraduate students and educators alike. By practicing the questions and reviewing the answer keys, learners can deepen their knowledge and develop a solid foundation in Software Engineering.

As the field of software development continues to evolve, the importance of structured engineering practices will only grow. Therefore, building a strong understanding of these fundamental concepts will help students become capable professionals who contribute effectively to the design, development, and maintenance of high-quality software systems.

ANSWER KEY

❖ Introduction to Software Engineering

01. c	02. c	03. b	04. b	05. b	06. c	07. b	08. b	09. b	10. b
11. c	12. b	13. b	14. c	15. b	16. b	17. b	18. b	19. c	20. b
21. c	22. b	23. b	24. c	25. b	26. b	27.b	28. b	29. c	30. b
31. c	32. c	33. b	34. b	35. b	36. c	37. b	38. c	39. c	40. c
41. b	42. b	43. b	44. b	45. b	46. b	47. b	48. b	49. b	50. b
51. b	52. b	53. c	54. c	55. b	56. b	57. b	58. b	59. c	60. b

❖ Software Project Planning

01. c	02. c	03. c	04. b	05. c	06. c	07. c	08. c	09. b	10. c
11. c	12. c	13. b	14. b	15. b	16. b	17. c	18. b	19. c	20. b
21. c	22. c	23. c	24. b	25. c	26. c	27. b	28. c	29. c	30. c
31. c	32. c	33. c	34. c	35. b	36. b	37. d	38. b	39.b	40. b
41. b	42. c	43. c	44. c	45. c	46. c	47. b	48. c	49.a	50. b
51. c	52. c	53. b	54. c	55. d	56. b	57. b	58. c	59.b	60. b
61. c	62. c	63. b	64. c	65. c	66. b	67. d	68. c	69. b	70. b
71. c	72. c	73. c	74. b	75. c	76. b	77. b	78. b	79.b	80. b
81. b	82. c	83. a	84. b	85. a	86. b	87. c	88. c	89. b	90. c
91. b	92. c	93. d	94. b	95. c	96. b	97. b	98. d	99. b	100. b
101. c	102. c	103. b	104. b	105. c	106. d	107. b	108. c	109. c	110. b
111. c	112. d	113. c	114. c	115. c	116. b	117. b	118. c	119.c	120. c
121. b	122. d	123. b	124. c	125. b	126. c	127. c	128. b	129.b	130. b
131. c	132. c	133. b	134. b	135. d	136. b	137. b	138. b	139.b	140. c
141. b	142. d	143. c	144. b	145. b	146. c	147. b	148. b	149.c	150. b

151. c	152. b	153. b	154. c	155. b	156. c	157. b	158. b	159. b	160. b
161. d	162. b	163. b	164. c	165. d	166. c	167. b	168. b	169. c	170. c
171. c	172. b	173. c	174. b	175. c	176. b	177. b	178. c	179. b	180. c
181. b	182. b	183. b	184. b	185. c	186. a	187. d	188. c	189. b	190. d
191. b	192. b	193. c	194. b	195. b	196. c	197. c	198. c	199. c	200. c
201. c	202. c & c	203. c	204. c	205. c	206. c	207. b	208. c	209. c	210. b
211. c	212. d	213. d	214. c	215. c	216. b	217. c	218. c	219. c	220. b
221. b	222. b	223. c	224. c	225. c	226. c	227.b	228. b	229. b	230. a
231. c	232. b	233. c	234. d	235. c	236. c	237. b	238. c	239. c	240. c
241. a	242. b	243. b	244. c						

❖ Software Cost Estimation

01. b	02. c	03. d	04. c	05. c	06. c	07. b	08. b	09. b	10. c
11. c	12. c	13. d	14. c	15. b	16. c	17. b	18. c	19. b	20. b
21. b	22. b	23. d	24. c	25. b	26. c	27. c	28. b & b	29. a	30. c
31. b	32. c	33. d	34. c	35. c	36. c	37. b	38. c	39. c	40. c
41. c	42. c	43. c	44. b	45. c	46. c	47. c	48. b	49. c	50. c
51. c	52. b	53. c	54. b	55. c	56. b	57. c	58. b	59. b	60. b
61. c	62. b								

❖ Requirement Specification

01. c	02. b	03. b	04. c	05. b	06. c	07. b	08. a	09. b	10. a
11. c	12. c	13. c	14. b	15. b	16. b	17. b	18. b	19. b	20. b
21. b	22. b	23. b	24. c	25. b	26. c	27. b	28. b	29. b	30. b
31. b	32. b	33. c	34. b	35. b	36. b	37. c	38. c	39. b	40. c
41. c	42. c	43. c	44. b	45. b					

❖ Software Design

01. c	02. c	03. b	04. c	05. c	06. c	07. b	08. c	09. b	10. b
11. c	12. b	13. c	14. b	15. b	16. c	17. b	18. b	19.b	20. c
21. b	22. c	23. b	24. b	25. c	26. b	27. b	28. b	29. b	30. c
31. c	32. b	33. b	34. b	35. b	36. c	37. c	38. b	39.c	40. c
41. c	42. b	43. c	44. d	45. d	46. b	47. a	48. c	49.d	50. c
51. c	52. b	53. c	54. c	55. d	56. c	57. c	58. b	59.b	60.c
61. d	62. b	63. c	64. c	65. b	66. c	67. b	68. c	69. c	70. b

❖ Software Implementation

01. b	02. c	03. c	04. b	05. c	06. b	07. c	08. d	09. b	10. c
11. b	12. c	13. c	14. b	15. d	16. c	17. b	18. c	19. c	20. b
21. c	22. b	23. c	24. c	25. c	26. b	27. c	28. c	29. c	30. d
31. b	32. b	33. b	34. b	35. c	36. c	37. b	38. c	39. c	40. b
41. a	42. c	43. c	44. c	45. c	46. d	47.b	48. b	49. c	50. d
51. c	52. c	53. c	54. d	55. c	56. c	57. c	58. b	59. b	60. c
61. c	62. c	63. c	64. c	65. c	66. b	67. c	68. c	69. b	70. c
71. b	72. b	73. b	74. a	75. c	76. b	77. c	78. c	79.c	80. b

81. c	82. c	83. c	84. c	85. b	86. d	87. b	88. b	89.d	90.c
91. b	92. b	93. d	94. b	95. c	96. c	97. b	98. c	99.b	100. b
101. b	102. b	103. c	104. c	105. c	106. b	107. d	108. b	109.c	110. c

- **Software Quality Assurance**
- **Software Testing**
- **Software Maintenance**

01. c	02. c	03. b	04. c	05. c	06. c	07. b	08. a	09.c	10. b
11. c	12. c	13. b	14. c	15. b	16. c	17. b	18. c	19.c	20. c
21. c	22. b	23. b	24. b	25. c	26. b	27. a	28. b	29.c	30. b
31. b	32. b	33. a	34. a	35. b	36. b	37. c	38. c	39.c	40. b
41. b	42. b	43. c	44. b	45. b	46. c	47. b	48. b	49.b	50. c
51. a	52. b	53. c	54. c	55. c	56. b	57. b	58. b	59.b	60. a
61. b	62. c	63. c	64. c	65. b	66. b	67. b	68. c	69. c	70. c
71. b	72. c	73. b	74. c	75. d	76. b	77. c	78. b	79.c	80. d
81. b	82. c	83. c	84. b	85. b	86. b	87. c	88. c	89. b	90. b
91. c	92. c	93. c	94. c	95. b	96. b	97. c	98. c	99. b	100. b
101. c	102. c	103. c	104. c	105. c	106. b	107. c	108. b	109. c	110. b
111. b	112. c	113. c	114. b	115. b	116. b	117. c	118. c	119. b	120. b
121. c	122. b	123. c	124. b	125. b	126. b	127. c	128. b	129. c	130. b
131. b	132. c	133. c	134. b	135. c	136. b	137. c	138. c	139. b	140. c
141. b	142. c	143. c	144. b	145. c	146. b	147. c	148. c	89.b	150. c

151. c	152. b	153. c	154. d	155. c	156. b	157. b	158. c	159. c	160. c
161. b	162. b	163. c	164. c	165. c	166. b	167. b	168. c	169. c	170. c
171. b	172. c	173. b	174. b	175. c	176. b	177. b	178. c	179. a	180. b
181. d	182. b	183. b	184. b	185. b	186. b	187. b	188. b	189. b	

About the Authors

Dr. A. Kalaiselvi serves as an Assistant Professor in the Department of Computer Science and Applications at Arul Anandar College (Autonomous), Karumathur. She holds an M.Sc. in Computer Science (First Rank Holder), B.Ed., M.Phil. in Cloud Computing, and a Ph.D. in Cloud Computing from Bharathidasan University, Tiruchirappalli. She has also qualified the State Eligibility Test (SET). She completed her higher studies at Madurai Kamaraj University and Bharathidasan University. With over seven years of teaching experience in higher education, she is actively involved in teaching undergraduate and postgraduate students in the field of Computer Science. Her research interests include Cloud Computing, Internet of Things (IoT), Artificial Intelligence, and Cybersecurity. She has published several research papers indexed in Web of Science (WoS) and Scopus, and has presented her research at various national and international conferences. She also contributes as an editor and author for academic books and conference proceedings, reflecting her dedication to scholarly advancement and knowledge dissemination. She is committed to promoting a strong research culture among students by encouraging participation in academic activities such as research publications, conferences, and scholarly projects. Through her teaching, mentoring, and guidance, she continuously strives to enhance students' academic excellence, research aptitude, and professional development

Ms. L. Jerman Lourthu Fathima, MCA, works as an Assistant Professor in the Department of Computer Science and Applications. She has several years of experience in IT field and interested in teaching undergraduate and postgraduate students in computer science–related disciplines. Her academic interests include Software Engineering, programming concepts, and emerging trends in software development. She is actively involved in academic instruction, curriculum delivery, and student mentoring, with a strong focus on conceptual clarity and examination-oriented learning. Her teaching experience has significantly contributed to the development of this book, making it a useful resource for students pursuing studies in Software Engineering

Mr. M. Loganathan, MCA, serves as an Assistant Professor in the Department of Computer Science and Applications at Arul Anandar College (Autonomous). He brings several years of professional experience in the IT industry, which enriches his academic contributions and classroom engagement. He is passionate about teaching both undergraduate and postgraduate students in Computer Science and related disciplines. His academic interests include Software Engineering, Programming Languages, and application-oriented computing. He is actively involved in teaching, academic mentoring, and curriculum development, with a strong commitment to enhancing students' conceptual clarity and practical skills. Through his learner- centred approach, he strives to bridge the gap between theoretical knowledge and real-world applications

ISBN 978-81-999642-9-7
9 788199 964297

Dr.BGR Publications