1. What do you mean by System Analysis?

Answer: System analysis is studying an existing system to understand how it works, identify problems, and gather requirements for a new or improved system. It's like diagnosing a patient before prescribing treatment—figuring out what's broken and what's needed.
Key Points:

- Involves data collection (interviews, surveys).
- Uses tools like DFDs and ERDs.
- Focuses on user needs and system goals.

Memorize Tip: Think of it as "detective work" for systems—find issues, plan fixes!

---

2. Briefly describe the types of systems with examples.

Answer: Systems are classified by their purpose and complexity:

1. **Open System**: Interacts with the environment, e.g., a bank's online system (takes customer inputs, processes transactions).
2. **Closed System**: Self-contained, no external interaction, e.g., a thermostat controlling room temperature.
3. **Manual System**: Human-driven, e.g., a paper-based library catalog.
4. **Automated System**: Uses computers, e.g., an online shopping website.
5. **Real-Time System**: Instant processing, e.g., air traffic control.
6. **Batch System**: Processes data in groups, e.g., payroll processing.

Memorize Tip: Open/closed = environment interaction; manual/automated = human vs. tech; real-time/batch = speed of processing.

---

3. Why is a database important in MIS? (Page 25, 2nd para)

Answer: A database in MIS (Management Information System) stores, organizes, and retrieves data efficiently to support decision-making. It ensures data is accurate, accessible, and secure, helping managers analyze trends and make informed choices.
Key Points:

- Centralizes data (no duplication).
- Enables fast queries (e.g., sales reports).
- Supports multiple users and applications.

Memorize Tip: Database = the brain of MIS, storing and serving data for smart decisions.

---

4. Define Candidate System. Write the basic steps for prototyping a system.

Answer:

- **Candidate System**: A proposed system designed to replace or improve the existing one, meeting user requirements after feasibility analysis.
- **Prototyping Steps**:
    1. **Identify Requirements**: Gather basic user needs.
    2. **Build Prototype**: Create a simple, working model.
    3. **Test with Users**: Get feedback on functionality.
    4. **Refine Prototype**: Update based on feedback.
    5. **Implement or Iterate**: Finalize or repeat for improvements.

Memorize Tip: Candidate system = "new system plan"; prototyping = build, test, tweak, repeat.

---

5. Reasons a new system does not meet user requirements.

Answer:

1. **Poor Requirements Gathering**: Misunderstood or incomplete user needs.
2. **Lack of User Involvement**: Users not consulted during development.
3. **Inadequate Testing**: Bugs or errors not caught.
4. **Poor Communication**: Misalignment between developers and users.
5. **Scope Creep**: Changing requirements mid-project.

Memorize Tip: Think "GUESS": Gathering (bad), User (no input), Errors (testing), Scope (creep), Speak (poor communication).

---

6. Short note on the multifaceted role of system analysis.

Answer: System analysis is like being a detective, planner, and bridge between users and developers. Roles include:

- **Investigator**: Studies current system issues.
- **Requirement Gatherer**: Collects user needs via interviews, surveys.
- **Modeler**: Creates DFDs, ERDs to visualize systems.
- **Problem Solver**: Designs solutions for inefficiencies.

- **Communicator**: Translates tech to user language.

Memorize Tip: System analyst = detective + planner + translator.

---

7. Differentiate between project-oriented and pool-oriented structure of system analysis.

Answer:

- **Project-Oriented**: Analysts focus on one project at a time, deeply involved from start to finish. *Pro*: Deep expertise, focused work. *Con*: Limited flexibility.
- **Pool-Oriented**: Analysts are shared across multiple projects, assigned as needed. *Pro*: Flexible resource use. *Con*: Less project-specific knowledge.

Memorize Tip: Project = one deep dive; Pool = juggling multiple tasks.

---

8. Explain the strategies of MIS planning.

Answer: MIS planning aligns IT with business goals. Strategies include:

1. **Top-Down**: Set goals at executive level, then detail plans (e.g., improve sales with CRM).
2. **Bottom-Up**: Identify operational needs, then build systems (e.g., inventory tracking).
3. **Critical Success Factors (CSF)**: Focus on key business needs (e.g., fast data access).
4. **Enterprise Analysis**: Study entire organization to align MIS with strategy.

Memorize Tip: Top-Down (boss leads), Bottom-Up (workers lead), CSF (key goals), Enterprise (big picture).

---

9. Strategies for determining information requirements.

Answer:

1. **Interviews**: Ask users directly about needs.
2. **Surveys/Questionnaires**: Collect data from many users.
3. **Observation**: Watch how users work with the system.
4. **Document Analysis**: Review existing reports, forms.
5. **Prototyping**: Build models to test requirements.

Memorize Tip: ISODP = Interview, Survey, Observe, Document, Prototype.

---

10. What do you mean by interview? (Page 138) Write the stages of the interview. (Page 140)

Answer:

- **Interview**: A face-to-face or virtual conversation to gather user requirements and understand system needs.
- **Stages**:
    1. **Preparation**: Define goals, select interviewees, prepare questions.
    2. **Conducting**: Ask questions, listen actively, take notes.
    3. **Follow-Up**: Clarify doubts, summarize findings, verify accuracy.

Memorize Tip: Interview = talk to learn; Stages = Prep, Talk, Check.

---

11. Advantages and drawbacks of structured and unstructured interview techniques.

Answer:

- **Structured Interview**:
    - *Advantages*: Fixed questions, consistent answers, easy to compare.
    - *Drawbacks*: Rigid, may miss unexpected insights.
- **Unstructured Interview**:
    - *Advantages*: Flexible, uncovers deep insights, conversational.
    - *Drawbacks*: Time-consuming, harder to analyze.

Memorize Tip: Structured = rigid but tidy; Unstructured = free but messy.

---

12. Define structured analysis? (Page 167) List the DFD symbols. (Page 171)

Answer:

- **Structured Analysis**: A method to model systems using structured tools like DFDs, focusing on processes and data flow.
- **DFD Symbols**:
    1. **Process**: Circle/oval (shows data transformation, e.g., “Calculate Salary”).
    2. **Data Flow**: Arrow (shows data movement, e.g., “Order Data”).
    3. **Data Store**: Open-ended rectangle (stores data, e.g., “Customer Database”).

4. **External Entity**: Square/rectangle (outside system, e.g., “Customer”).

Memorize Tip: Structured = organized modeling; DFD = Circle (process), Arrow (flow), Open rectangle (store), Square (entity).

---

13. Create a decision tree for bookstore discounts.

Answer: Scenario: Bookstores get a 25% trade discount; libraries/individuals get 5% for 6–19 copies, 10% for 20–49 copies, 15% for 50+ copies per book title.

Decision Tree:

text

Copy

Is it a Bookstore?

├── Yes: 25% discount

└── No (Library/Individual):

  ├── Copies 6–19: 5% discount

  ├── Copies 20–49: 10% discount

  └── Copies 50+: 15% discount

Memorize Tip: Bookstore = flat 25%; Others = tiered by quantity (5, 10, 15%).

---

14. What is a feasibility study? (Page 200) Write down the steps in feasibility analysis. (Page 202)

Answer:

- **Feasibility Study**: Evaluating if a proposed system is practical and worthwhile (technically, economically, operationally).
- **Steps**:
  1. **Define Problem**: Identify system issues and goals.
  2. **Gather Data**: Collect info on costs, resources, needs.
  3. **Evaluate Alternatives**: Compare possible solutions.
  4. **Assess Feasibility**: Check technical, economic, operational viability.
  5. **Recommend**: Propose best solution or no-go.

Memorize Tip: Feasibility = “can we do it?”; Steps = Problem, Data, Compare, Check, Recommend.

---

15. Considerations in feasibility analysis? Most crucial? Why? (Page 201)

Answer:

- **Considerations**:
    1. **Technical**: Can we build it with available tech?
    2. **Economic**: Is it cost-effective (benefits vs. costs)?
    3. **Operational**: Will users accept and use it?
    4. **Schedule**: Can we finish on time?
- **Most Crucial**: Economic feasibility—because if costs outweigh benefits, the project isn't sustainable.

Memorize Tip: TEOS = Technical, Economic, Operational, Schedule; Economic = money rules.

---

16. Define and explain the procedure for cost/benefit determination.

Answer:

- **Cost/Benefit Determination**: Comparing system costs (development, operation) to benefits (savings, efficiency) to justify investment.
- **Procedure**:
    1. **Identify Costs**: Hardware, software, labor, training.
    2. **Identify Benefits**: Tangible (e.g., cost savings) and intangible (e.g., better decisions).
    3. **Quantify**: Assign dollar values where possible.
    4. **Compare**: Use metrics like ROI or payback period.
    5. **Decide**: Proceed if benefits outweigh costs.

Memorize Tip: Cost/Benefit = weigh money spent vs. gained; Steps = List costs, benefits, quantify, compare, decide.

---

17. Formula for present value, and calculate $1,500 in 4 years at 10% interest.

Answer:

- **Formula**: Present Value (PV) = FV / (1 + r)^n
    - FV = Future Value, r = interest rate, n = years.
- **Calculation**: PV = $1,500 / (1 + 0.10)^4 = $1,500 / 1.4641 = **$1,023.51** (approx).

Memorize Tip: PV = FV divided by (1 + interest)^years; $1,500 becomes ~$1,024 after 4 years at 10%.

---

18. Design methodologies used in system design.

Answer:

1. **Structured Design**: Breaks system into modules (e.g., using DFDs).
2. **Object-Oriented Design**: Uses objects, classes, UML (e.g., use case diagrams).
3. **Prototyping**: Builds iterative models for testing.
4. **Agile Design**: Incremental, user-focused development.

Memorize Tip: S.O.P.A = Structured, Object-Oriented, Prototyping, Agile.

---

19. Distinguish between HIPO and IPO.

Answer:

- **HIPO (Hierarchy plus Input-Process-Output)**: Visualizes system as a hierarchy of modules, with detailed IPO charts for each module. Focus: System structure + process details.
- **IPO (Input-Process-Output)**: Simple chart showing inputs, processes, and outputs for a module. Focus: Single process flow.

Memorize Tip: HIPO = hierarchy + IPO; IPO = just one process's flow.

---

20. What is input and output design? (Pages 283, 296)

Answer:

- **Input Design**: Creating user-friendly ways to enter data (e.g., forms, screens).
- **Output Design**: Designing reports, dashboards to present data clearly. **Key Principles**: Simple, clear, error-free, user-focused.

Memorize Tip: Input = easy data entry; Output = clear data display.

---

21. What is unique about online data entry? Role of CRT terminal? (Pages 289, 292)

Answer:

- **Online Data Entry**: Real-time data input via computer (e.g., web forms), instant validation, no paper.

- **CRT Terminal Role**: Acts as input (keyboard for data entry) and output (screen for feedback) device, enabling interactive, real-time data processing.

Memorize Tip: Online = instant input; CRT = screen + keyboard for real-time work.

---

22. Advantages and disadvantages of file organization methods.

Answer:

- **Sequential**:
  - *Adv*: Simple, fast for batch processing.
  - *Dis*: Slow for random access.
- **Indexed Sequential**:
  - *Adv*: Faster access with index, supports sequential and random.
  - *Dis*: Extra storage for index.
- **Direct (Random)**:
  - *Adv*: Fast access for specific records.
  - *Dis*: Complex, needs hashing.

Memorize Tip: Sequential = simple but slow; Indexed = faster but bulky; Direct = quick but tricky.

---

23. Short note on the role of the Database Administrator.

Answer: The Database Administrator (DBA) manages the database system:

- Designs and maintains database structure.
- Ensures data security and backups.
- Optimizes performance and access.
- Supports users and applications.

Memorize Tip: DBA = database's guardian—designs, secures, tunes.

---

24. What is system testing? Why is it tested?

Answer:

- **System Testing**: Checking if the entire system works as designed (functionality, performance).

- **Why**: To catch errors, ensure user requirements are met, and verify reliability before deployment.

Memorize Tip: System testing = full system checkup to avoid failures.

---

25. Types of system tests.

Answer:

1. **Unit Testing**: Tests individual modules.
2. **Integration Testing**: Tests module interactions.
3. **System Testing**: Tests entire system functionality.
4. **Acceptance Testing**: Verifies user requirements.

Memorize Tip: UISA = Unit, Integration, System, Acceptance.

---

26. What is implementation? How does it differ from conversion?

Answer:

- **Implementation**: Building, testing, and deploying the system (coding, training, installation).
- **Conversion**: Switching from old system to new (e.g., parallel, direct). **Difference**: Implementation = full process; Conversion = just the switch.

Memorize Tip: Implementation = build + launch; Conversion = just the swap.

---

27. Distinguish between software modification and software system audit. (Pages 404, 405)

Answer:

- **Software Modification**: Changing code to fix bugs or add features.
- **Software System Audit**: Reviewing system to ensure it meets standards, security, and performance.
  Difference: Modification = code changes; Audit = system checkup.

Memorize Tip: Modify = tweak code; Audit = inspect system.

---

28. Factors considered prior to system selection.

Answer:

1. **Functionality**: Meets user needs.

2. **Cost**: Fits budget.
3. **Scalability**: Handles future growth.
4. **Compatibility**: Works with existing systems.
5. **Support**: Vendor reliability, maintenance.

Memorize Tip: FCSCS = Functionality, Cost, Scalability, Compatibility, Support.

---

29. Software criteria for selection.

Answer:

1. **Usability**: Easy to use.
2. **Reliability**: Stable, low errors.
3. **Performance**: Fast and efficient.
4. **Security**: Protects data.
5. **Maintainability**: Easy to update.

Memorize Tip: URPSM = Usability, Reliability, Performance, Security, Maintainability.

---

30. What is project management? (Page 447) Steps for establishing a system project. (Page 448)

Answer:

- **Project Management**: Planning, organizing, and controlling resources to complete a system project.
- **Steps**:
    1. **Define Objectives**: Set project goals.
    2. **Plan Tasks**: Break into activities.
    3. **Assign Resources**: Allocate team, budget.
    4. **Schedule**: Create timeline (e.g., Gantt chart).
    5. **Monitor & Control**: Track progress, fix issues.

Memorize Tip: PM = plan, do, check; Steps = Goals, Tasks, Resources, Schedule, Monitor.

---

31. What is a Gantt chart? (Page 450) How to develop one? How does it differ from a PERT chart? (Page 453)

Answer:

- **Gantt Chart**: A bar chart showing project tasks and timelines.
- **Development**: List tasks, estimate durations, draw bars on a timeline.
- **Difference from PERT**:
    - **Gantt**: Simple, shows tasks vs. time.
    - **PERT**: Network diagram, shows task dependencies and critical path.

Memorize Tip: Gantt = bars for time; PERT = arrows for dependencies.

---

32. Major threats to system security? Most serious? Why?

Answer:

- **Threats**:
    1. **Hacking**: Unauthorized access.
    2. **Malware**: Viruses, ransomware.
    3. **Phishing**: Tricking users for data.
    4. **Data Breaches**: Leaked sensitive info.
- **Most Serious**: Data breaches—expose sensitive data, harm reputation, and cost millions.

Memorize Tip: Threats = HMPD (Hacking, Malware, Phishing, Data breach); Breaches = worst due to data loss.

---

33. What is encryption? How does it work? What systems use it?

Answer:

- **Encryption**: Converting data into unreadable code to protect it.
- **How It Works**: Uses algorithms (e.g., AES) and keys to scramble/unscramble data.
- **Systems**: Banking, e-commerce, healthcare, any system with sensitive data (e.g., credit card systems).

Memorize Tip: Encryption = data lock; Algorithm + key; Used in banks, shops, hospitals.

---

34. Define a system. Characteristics of a good system.

Answer:

- **System**: A set of components (people, hardware, software) working together for a goal.

- **Characteristics**:
    1. **Efficient**: Fast, low resource use.
    2. **Reliable**: Consistent, error-free.
    3. **User-Friendly**: Easy to use.
    4. **Scalable**: Grows with needs.
    5. **Secure**: Protects data.

Memorize Tip: System = teamwork for goal; Good = ERUSS (Efficient, Reliable, User-friendly, Scalable, Secure).

---

35. Role and responsibilities of a System Analyst.

Answer:

- **Role**: Bridge between users and developers, designing effective systems.
- **Responsibilities**:
    1. Analyze current systems.
    2. Gather requirements.
    3. Design solutions (DFDs, ERDs).
    4. Coordinate with developers.
    5. Test and implement systems.

Memorize Tip: Analyst = user-tech bridge; Duties = Analyze, Gather, Design, Coordinate, Test.

---

36. Compare and contrast SDLC with the Agile model.

Answer:

- **SDLC**: Structured, sequential phases (plan, analyze, design, implement, maintain). *Pro*: Clear plan, good for large projects. *Con*: Rigid, slow changes.
- **Agile**: Iterative, flexible, delivers small increments. *Pro*: Fast, adaptive. *Con*: Less predictable.
- **Similarities**: Both aim for quality systems, involve testing.

Memorize Tip: SDLC = step-by-step, rigid; Agile = quick loops, flexible.

---

37. Steps involved in initial investigation of a system.

Answer:

1. **Identify Problem**: Understand system issues.
2. **Gather Data**: Interview users, review documents.
3. **Analyze Current System**: Find inefficiencies.
4. **Define Objectives**: Set goals for new system.
5. **Report Findings**: Propose next steps.

Memorize Tip: IPADO = Identify, Probe, Analyze, Define, Outline.

---

38. How can feasibility studies help in planning a successful system?

Answer: Feasibility studies ensure a system is practical by checking:

- Technical: Can we build it?
- Economic: Is it worth the cost?
- Operational: Will users adopt it?
- Helps avoid wasted resources, aligns system with goals.

Memorize Tip: Feasibility = reality check for tech, money, users; saves time and cash.

---

39. Distinguish between tangible and intangible benefits with examples.

Answer:

- **Tangible Benefits**: Measurable, e.g., cost savings ($10,000/year), faster processing (50% time reduction).
- **Intangible Benefits**: Non-measurable, e.g., better customer satisfaction, improved decision-making.

Memorize Tip: Tangible = count it (money, time); Intangible = feel it (satisfaction, decisions).

---

40. Key principles of good input/output design.

Answer:

1. **Simplicity**: Easy to use, minimal steps.
2. **Clarity**: Clear labels, instructions.
3. **Accuracy**: Validates data, reduces errors.
4. **User-Centric**: Matches user needs, habits.

5. **Consistency**: Uniform design across system.

Memorize Tip: SCACS = Simple, Clear, Accurate, User-Centric, Consistent.

---

41. File organization techniques and relevance in database design.

Answer:

- **Techniques**:
    1. **Sequential**: Records in order, good for batch (e.g., payroll).
    2. **Indexed Sequential**: Index for faster access, suits mixed use.
    3. **Direct**: Hash-based, fast for random access (e.g., customer lookup).
- **Relevance**: Affects database speed, storage, and query efficiency; chosen based on access needs.

Memorize Tip: Sequential = line; Indexed = line + shortcuts; Direct = instant; Pick for speed/storage.

---

42. Importance of Quality Assurance (QA) in system development.

Answer: QA ensures the system meets standards, works reliably, and satisfies users by:

- Testing for bugs.
- Verifying requirements.
- Improving user trust and system success.

Memorize Tip: QA = quality checkpoint; Tests, verifies, builds trust.

---

43. Importance of security measures and disaster recovery planning.

Answer:

- **Security Measures**: Protect data from hacks, breaches (e.g., encryption, firewalls).
- **Disaster Recovery**: Plans Sagittarius Plans for data loss, system crashes (e.g., backups, alternate sites).
- **Importance**: Ensures data safety, system uptime, business continuity.

Memorize Tip: Security = data shield; Disaster Recovery = system comeback plan.

---

44. Key phases of the Software Development Life Cycle (SDLC).

Answer:

1. **Planning**: Define goals, scope, resources.
2. **Analysis**: Gather requirements, study system.
3. **Design**: Create system blueprints (DFDs, ERDs).
4. **Implementation**: Code, test, deploy system.
5. **Maintenance**: Fix bugs, update system.

Memorize Tip: PADIM = Plan, Analyze, Design, Implement, Maintain.

---

45. Project management spectrum and its components.

Answer:

- **Spectrum**: People, process, product, technology.
- **Components**:
    1. **People**: Team roles, skills.
    2. **Process**: Steps like SDLC or Agile.
    3. **Product**: System features, quality.
    4. **Technology**: Tools, platforms used.

Memorize Tip: PPPT = People, Process, Product, Tech.

---

46. Steps in the Change Control process in software configuration management.

Answer:

1. **Request Change**: Identify needed change.
2. **Evaluate Impact**: Assess effects on system.
3. **Approve/Reject**: Decision by change board.
4. **Implement Change**: Update system.
5. **Verify**: Test and document changes.

Memorize Tip: REAIV = Request, Evaluate, Approve, Implement, Verify.

---

47. Short note on project management tools.

Answer: Tools help plan, track, and manage projects:

- **Gantt Chart**: Visualizes task timelines.
- **PERT Chart**: Shows task dependencies.

- **Software**: MS Project, Jira, Trello for scheduling, tracking.

Memorize Tip: Tools = Gantt, PERT, software for project control.

---

48. What is Requirement Elicitation? Describe the process.

Answer:

- **Requirement Elicitation**: Gathering user needs for the system.
- **Process**:
    1. **Plan**: Define goals, select methods.
    2. **Collect Data**: Use interviews, surveys, observation.
    3. **Analyze**: Identify key requirements.
    4. **Document**: Create requirement specs.
    5. **Validate**: Confirm with users.

Memorize Tip: Elicitation = need-finding; Steps = Plan, Collect, Analyze, Document, Validate.

---

49. Differences and similarities between Coupling and Cohesion? Advantages of modularization.

Answer:

- **Coupling**: Degree of dependency between modules (low is better).
- **Cohesion**: How well a module's tasks work together (high is better).
- **Similarities**: Both affect system quality, modularity.
- **Differences**: Coupling = inter-module links; Cohesion = intra-module focus.
- **Advantages of Modularization**: Easier maintenance, testing, reuse.

Memorize Tip: Coupling = module connections (keep low); Cohesion = module focus (keep high); Modularization = easy upkeep.

---

50. Role of DFD and Data Dictionary in system analysis.

Answer:

- **DFD**: Shows data flow, processes, stores, entities; visualizes system logic.
- **Data Dictionary**: Defines data elements (e.g., fields, formats); clarifies data meaning.
  Role: DFD maps system flow; Data Dictionary explains data details.

Memorize Tip: DFD = system map; Data Dictionary = data guidebook.

---

51. Steps of design process. Best software design approach.

Answer:

- **Steps**:
    1. Define requirements.
    2. Create system architecture.
    3. Design modules, interfaces.
    4. Test design.
    5. Refine and document.
- **Best Approach**: Agile—iterative, user-focused, adapts to changes.

Memorize Tip: Design = Define, Architect, Detail, Test, Refine; Agile = flexible winner.

---

52. Guidelines for designing user-friendly interfaces.

Answer:

1. **Simple**: Minimal, clear design.
2. **Consistent**: Uniform look and feel.
3. **Intuitive**: Easy to navigate.
4. **Feedback**: Show user actions (e.g., button clicks).
5. **Accessible**: Works for all users.

Memorize Tip: SCIFA = Simple, Consistent, Intuitive, Feedback, Accessible.

---

53. Distinguish between LOC-based and FP-based estimation methods.

Answer:

- **LOC (Lines of Code)**: Estimates effort based on code size. *Pro*: Simple. *Con*: Varies by language.
- **FP (Function Points)**: Measures functionality (inputs, outputs). *Pro*: Language-independent. *Con*: Complex to calculate.

Memorize Tip: LOC = code count; FP = function count; FP more universal.

---

54. What is software testing? Lehman's evolution categories.

Answer:

- **Software Testing**: Verifying system functionality, reliability.
- **Lehman's Categories**:
    1. **S-Type**: Static, fixed requirements (e.g., calculator).
    2. **P-Type**: Practical, evolving needs (e.g., business apps).
    3. **E-Type**: Embedded, complex, adaptive (e.g., OS).

Memorize Tip: Testing = system check; Lehman = SPE (Static, Practical, Embedded).

---

55. Software documentation.

Answer: Records system details:

- **Types**: User manuals, technical specs, design docs.
- **Purpose**: Guides users, developers; aids maintenance.

Memorize Tip: Docs = system's user manual + tech guide.

---

56. Basic idea of the COCOMO model and its types.

Answer:

- **COCOMO (Constructive Cost Model)**: Estimates project effort, time, cost based on size.
- **Types**:
    1. **Basic**: Simple, size-based estimates.
    2. **Intermediate**: Adds complexity factors.
    3. **Detailed**: Deep analysis of team, process.

Memorize Tip: COCOMO = cost predictor; Basic, Intermediate, Detailed.

---

57. Software testing levels.

Answer:

1. **Unit**: Test single modules.
2. **Integration**: Test module interactions.
3. **System**: Test entire system.
4. **Acceptance**: Test user requirements.

Memorize Tip: UISA = Unit, Integration, System, Acceptance.

---

58. Documents needed before and after software testing.

Answer:

- **Before**: Test plan, requirements doc, design specs.
- **After**: Test results, bug reports, user acceptance doc.

Memorize Tip: Before = plan + specs; After = results + approval.

---

59. What is software reengineering? Need and process.

Answer:

- **Reengineering**: Updating old software to improve performance, maintainability.
- **Need**: Fix outdated code, add features, reduce costs.
- **Process**: Analyze code, restructure, test, deploy.

Memorize Tip: Reengineering = software makeover; Need = old to new; Process = analyze, fix, test, launch.

---

60. Software reengineering vs. Reverse Engineering.

Answer:

- **Reengineering**: Modifies code for improvement.
- **Reverse Engineering**: Analyzes code to understand its design. **Difference**: Reengineering = improve; Reverse = decode.

Memorize Tip: Reengineering = upgrade; Reverse = uncover.

---

61. Component Reuse Process.

Answer:

1. **Identify Components**: Find reusable parts (e.g., modules).
2. **Evaluate**: Check compatibility, quality.
3. **Adapt**: Modify for new system.
4. **Integrate**: Use in new project.
5. **Test**: Ensure it works.

Memorize Tip: Reuse = find, check, tweak, use, test.

---

62. Why CASE tools are needed? Components of CASE tools.

Answer:

- **Need**: Automate design, coding, testing; save time, reduce errors.
- **Components**:
    1. Diagramming tools (DFDs, ERDs).
    2. Code generators.
    3. Testing tools.
    4. Documentation tools.

Memorize Tip: CASE = automation helper; Components = diagrams, code, test, docs.

---

63. Short note on CASE tools and CASE workbenches.

Answer:

- **CASE Tools**: Software for automating system development tasks (e.g., Visio for DFDs).
- **CASE Workbenches**: Integrated sets of CASE tools for end-to-end support (e.g., Rational Rose).

Memorize Tip: CASE Tools = single task helpers; Workbenches = all-in-one toolkits.

Viva Prep Tips:

- **Practice**: Recite answers using the **Memorize Tip** phrases.
- **Visualize**: Link terms to examples (e.g., DFD = system map).
- **Group**: Study related questions together (e.g., SDLC, Agile, project management).
- **Examples**: Use real-world cases (e.g., bank system for MIS, bookstore for decision tree).

Additional Important Questions and Viva-Ready Answers

1. What is a Data Dictionary, and why is it important in system analysis? (Page ~180)

Answer: A Data Dictionary is a centralized document that defines all data elements in a system (e.g., fields, formats, meanings). It ensures everyone understands data

consistently.
Importance:

- Clarifies data for developers and users.
- Prevents errors in database design.
- Supports maintenance and updates.
  Example: In a library system, it defines "Book_ID" as a 6-digit number.
  Memorize Tip: Data Dictionary = system's data glossary; clarifies, prevents errors, aids upkeep.

2. What is Normalization, and why is it used in database design? (Page ~250)

Answer: Normalization organizes data into tables to remove redundancy and ensure data integrity.
Why Used:

- Eliminates duplicate data (e.g., same customer name in multiple tables).
- Prevents anomalies in data updates.
- Improves query efficiency.
  Example: Splitting a customer-order table into separate Customer and Order tables.
  Memorize Tip: Normalization = tidy data tables; cuts duplicates, saves integrity.

3. What is the role of a Use Case Diagram in system design? (Page ~190)

Answer: A Use Case Diagram (part of UML) shows system functions and user interactions.
Role:

- Maps what users do (e.g., "Place Order").
- Clarifies system scope and requirements.
- Guides developers on functionality.
  Example: For an online store, shows "Customer" linked to "Browse Products."
  Memorize Tip: Use Case = user action map; shows who does what.

4. What is System Maintenance, and why is it critical? (Page ~410)

Answer: System Maintenance is updating and fixing a system after deployment to keep it running smoothly.
Types:

- Corrective: Fix bugs.
- Adaptive: Update for new needs.
- Perfective: Add features.
  Why Critical: Ensures reliability, meets new user needs, extends system life.
  Memorize Tip: Maintenance = system upkeep; Fix, Adapt, Improve.

5. What is User Training, and why is it essential in system implementation? (Page ~400)

Answer: User Training teaches users how to use the new system effectively.
Why Essential:

- Reduces errors and resistance.
- Boosts user confidence and adoption.
- Ensures system meets goals.
  Example: Training staff on a new inventory system.
  Memorize Tip: Training = user empowerment; cuts errors, boosts use.

6. What is a Context Diagram, and how does it differ from a DFD? (Page ~170)

Answer: A Context Diagram is a high-level DFD showing the system as a single process and its interactions with external entities.
Difference:

- Context Diagram: One process, shows system boundaries.
- DFD: Multiple processes, shows detailed data flow.
  Example: Context Diagram shows "Library System" with "User" and "Supplier."
  Memorize Tip: Context = big picture (1 process); DFD = detailed flow.

7. What is the purpose of a System Flowchart? (Page ~175)

Answer: A System Flowchart visually represents the flow of data and processes in a system, including hardware, software, and manual steps.
Purpose:

- Shows system overview.
- Identifies bottlenecks or inefficiencies.
- Guides implementation.
  Example: Flowchart for order processing with input, processing, and output steps.
  Memorize Tip: Flowchart = system roadmap; shows data path.

8. What is Decision Support System (DSS), and how does it differ from MIS? (Page ~30)

Answer: A DSS helps managers make complex, unstructured decisions using data and models.
Difference:

- DSS: Flexible, for specific decisions (e.g., pricing strategy).
- MIS: Routine, structured reports (e.g., sales summaries).
  Memorize Tip: DSS = decision helper (flexible); MIS = routine data (fixed).

9. What is the role of a System Specification Document? (Page ~210)

Answer: A System Specification Document details the system's requirements, design, and functionality.
Role:

- Guides developers during implementation.
- Ensures user and developer alignment.
- Serves as a reference for testing.
  Example: Specs for a payroll system include input forms, calculations.
  Memorize Tip: Spec Doc = system blueprint; guides build, aligns teams.

10. What is Rapid Application Development (RAD), and how does it work? (Page ~220)

Answer: RAD is a fast development method using prototyping and iterative feedback.
How It Works:

- Build quick prototypes.
- Get user feedback.
- Refine until complete.
  Example: Developing a mobile app with user-tested prototypes.
  Memorize Tip: RAD = fast build; prototype, feedback, refine.

11. What is the purpose of a Data Flow Audit? (Page ~415)

Answer: A Data Flow Audit checks how data moves through the system to ensure accuracy and security.
Purpose:

- Detects errors or leaks.
- Ensures compliance with standards.
- Improves system reliability.
  Example: Auditing customer data flow in a banking system.
  Memorize Tip: Data Flow Audit = data tracker; checks accuracy, security.

12. What is Structured English, and how is it used in system analysis? (Page ~185)

Answer: Structured English is a clear, simplified language to describe system processes logically.
Use:

- Documents process logic (e.g., "IF order > 50, apply 10% discount").
- Bridges user and developer understanding.
- Supports DFDs and decision tables.
  Memorize Tip: Structured English = process in plain words; clarifies logic.

13. What is the role of a System Analyst in Quality Assurance? (Page ~420)

Answer: The System Analyst ensures quality by:

- Defining clear requirements.
- Reviewing designs and tests.
- Ensuring system meets user needs.
- **Example**: Checking if a payroll system calculates taxes correctly. **Memorize Tip**: Analyst in QA = quality gatekeeper; defines, reviews, ensures.

14. What is the importance of Backup and Recovery in systems? (Page ~430)

Answer: Backup saves data copies; Recovery restores data after loss. **Importance**:

- Protects against crashes, hacks.
- Ensures business continuity.
- Maintains user trust. **Example**: Daily backups for an e-commerce database. **Memorize Tip**: Backup = data safety net; Recovery = data comeback.

15. What is a Decision Table, and how is it used? (Page ~172)

Answer: A Decision Table lists conditions and actions to define system logic. **Use**:

- Simplifies complex decisions.
- Guides programming and testing. **Example**: Table for discounts: "If quantity > 50, apply 15%." **Memorize Tip**: Decision Table = logic chart; lists conditions, actions.