1. Define Software Engineering.

Software Engineering is a systematic approach to the development, operation, maintenance, and retirement of software. It applies engineering principles to software development in order to create reliable and efficient software products.

2. Write the components and characteristics of Software Engineering.

- **Components**:
  - **Processes**: Defines the framework for planning, structuring, and controlling software development.
  - **Methods**: The technical approaches and practices applied during the development of software.
  - **Tools**: Software tools that support the processes and methods.
- **Characteristics**:
  - **Systematic Approach**: Emphasizes a structured and organized approach to software development.
  - **Quality Focus**: Aims to deliver high-quality software that meets user requirements.
  - **Iterative Process**: Involves continuous improvement and iterative cycles.
  - **Scalability**: Software engineering practices support the development of both small and large systems.

3. Explain the generic view of Software Engineering.

The generic view of Software Engineering involves:

- **Software Process**: A set of activities, methods, practices, and transformations that people use to develop and maintain software.
- **Software Project Management**: Involves planning, monitoring, and controlling software projects.
- **Software Construction**: The detailed creation of working, meaningful software through coding, testing, debugging, and refactoring.

4. Describe the life cycle of Extreme Programming (XP).

Extreme Programming (XP) life cycle includes the following phases:

- **Planning**: User stories are written, and an initial plan is created.
- **Design**: Simple designs are created, focusing on what is necessary.
- **Coding**: Code is developed following XP practices such as pair programming.
- **Testing**: Continuous testing to ensure quality.
- **Release**: Frequent releases in short iterations.

5. Write about different types of software development models and their phases.

- **Waterfall Model**: Sequential phases including requirements, design, implementation, testing, deployment, and maintenance.
- **V-Model**: An extension of the waterfall model that emphasizes verification and validation.
- **Iterative Model**: Develops the system through repeated cycles (iterative) and in smaller portions at a time (incremental).

- **Spiral Model**: Combines iterative development with systematic aspects of the waterfall model, focusing on risk assessment.
- **Agile Model**: An incremental model focusing on adaptability, collaboration, and customer feedback.

6. Define agility in the context of software development.

Agility refers to the ability to adapt quickly and efficiently to changes in software development requirements, environments, and goals. It emphasizes flexibility, customer collaboration, and rapid delivery of functional software.

7. Explain Agile Development.

Agile Development is a methodology focused on iterative development, where requirements and solutions evolve through collaboration between cross-functional teams. Agile methods prioritize customer satisfaction, continuous delivery of working software, and responsiveness to changing requirements.

8. Provide examples and numerical related to the Basic COCOMO Model.

The Basic COCOMO (Constructive Cost Model) estimates software development cost based on the size of the project:

- **Example**: A small project of 2,000 lines of code (KLOC) could be estimated as follows:
    - Effort (E) = a * (KLOC)^b = 2.4 * (2)^1.05 = 5.1 person-months
    - Development Time (D) = c * (E)^d = 2.5 * (5.1)^0.38 = 6.5 months

9. Provide examples and numerical related to the Intermediate COCOMO Model.

The Intermediate COCOMO model adds more factors like product complexity and team experience:

- **Example**: For a 10,000 KLOC project with complexity and experience factors:
    - Effort (E) = a * (KLOC)^b * (Product complexity factor) * (Team experience factor)
    - Suppose factors adjust the effort to 3.0 and 1.2 respectively, then E = 3.0 * (10)^1.05 * 1.

2 * 1.2 = 37.5 person-months.

10. Write about Function Point Analysis and its application.

Function Point Analysis (FPA) is a standardized method to measure the functionality delivered by the software. It focuses on assessing the user functionality from a user's perspective:

- **Steps in FPA**:
    1. **Identify Functions**: Includes inputs, outputs, user interactions, files, and external interfaces.
    2. **Classify Complexity**: Each function is classified based on its complexity (simple, average, complex).
    3. **Calculate Function Points**: Assign weights based on complexity and sum them up to get the total Function Points (FP).
- **Applications**: FPA is used for project estimation, measuring productivity, benchmarking, and managing software projects.

11. List and explain the tasks involved in Requirement Engineering.

Requirement Engineering involves the following tasks:

- **Requirement Elicitation**: Gathering requirements from stakeholders through interviews, surveys, and observation.

- **Requirement Analysis**: Analyzing requirements to ensure they are complete, consistent, and unambiguous.
- **Requirement Specification**: Documenting the requirements in a Software Requirement Specification (SRS) document.
- **Requirement Validation**: Checking that the documented requirements meet the needs and expectations of stakeholders.
- **Requirement Management**: Managing changes to requirements over time, ensuring they remain aligned with project goals.

12. Provide an overview of stakeholders in a software project.

Stakeholders in a software project include:

- **Customers**: The individuals or organizations who request the software.
- **Users**: The end-users who will use the software.
- **Project Managers**: Individuals responsible for planning, executing, and closing the project.
- **Developers**: Engineers who write code and build the software.
- **Testers**: Professionals who ensure the software meets quality standards.
- **Regulators**: Authorities who ensure the software complies with regulations and standards.

13. Discuss the mandatory components of an SRS document.

An **SRS (Software Requirement Specification)** document typically includes:

- **Introduction**: Overview of the project, including purpose, scope, and objectives.

- **Overall Description**: General factors affecting the product and its requirements.
- **Functional Requirements**: Specific behaviors and functionalities the software must have.
- **Non-Functional Requirements**: Performance, security, usability, and other quality attributes.
- **Design Constraints**: Constraints imposed by hardware, system architecture, or other external factors.

14. Provide an example of an SRS.

An example SRS might include:

- **Project Name**: Online Bookstore
- **Introduction**: This document outlines the requirements for the Online Bookstore project, which allows users to browse, search, and purchase books online.
- **Functional Requirements**:
    - **Search Functionality**: Users should be able to search for books by title, author, or genre.
    - **Shopping Cart**: Users can add, remove, and view items in a shopping cart.
    - **Checkout Process**: Users should be able to purchase items using credit cards or digital wallets.