Software Testing Short Notes

---

1. Software Testing

- The process of evaluating and verifying that a software product meets business and technical requirements and is defect-free.

---

2. Importance of Testing

- Detects defects early
- Reduces cost and rework
- Ensures quality and user satisfaction

---

3. Principles of Software Testing

1. **Testing shows the presence of defects**.
2. **Exhaustive testing is impossible**.
3. **Early testing saves time and money**.
4. **Defect clustering**.
5. **Pesticide paradox**.
6. **Testing is context-dependent**.
7. **Absence-of-errors fallacy**.

---

4. Quality Assurance (QA) vs Quality Control (QC)

Feature	**QA**	**QC**
Focus	Process	Product
Goal	Prevent defects	Detect defects
Activity	Proactive	Reactive
Example	Process audit	Testing features

5. Static Testing vs Dynamic Testing

Feature	Static Testing	Dynamic Testing
Meaning	No code execution	Code execution
Type	Reviews, Walkthroughs	Unit Testing, System Testing
Example	Code inspection	Running test cases

---

6. Verification vs Validation

Term	Verification	Validation
Meaning	Are we building the product **right**?	Are we building the **right** product?
Type	Static Testing	Dynamic Testing
Activity	Reviews, walkthroughs	Execution of code

---

7. Types of SDLC Models

1. **Waterfall Model**: Linear sequential model.
2. **V-Model**: Each development phase has a corresponding testing phase.
3. **Incremental Model**: Software is built incrementally with partial implementations.
4. **Iterative Model**: Software is developed through repeated cycles (iterations).
5. **Spiral Model**: Focus on risk analysis and prototyping.
6. **Agile Model**: Iterative, collaborative approach with frequent feedback loops (e.g., Scrum).

---

8. Software Development Life Cycle (SDLC)

1. **Requirement Gathering**: Understanding user needs.
2. **System Design**: Creating a blueprint for the system.
3. **Implementation (Coding)**: Writing the code.
4. **Testing**: Verifying the code works.
5. **Deployment**: Making the software available for users.
6. **Maintenance**: Updating and fixing the software post-deployment.

---

9. Types of Testing

- **Manual Testing**: Performed by testers manually.

- **Automation Testing**: Performed using testing tools like Selenium, JUnit.

---

10. Levels of Testing

1. **Unit Testing**: Testing individual components.
2. **Integration Testing**: Testing interactions between integrated components.
3. **System Testing**: Testing the complete system.
4. **Acceptance Testing**: Verifying if the system meets business needs.

---

11. Black Box Testing

- Tester doesn't need to know the internal workings of the application.
- Focuses on inputs and outputs, checking whether the system behaves as expected.
- Techniques:
    - Equivalence Partitioning-divides input data into valid & invalid groups
    - Boundary Value Analysis-test the values as minimum & maximum
    - Decision Table Testing-check different input condition and expected output
    - State Transition Testing- eg in Atm counter

---

12. White Box Testing

- Involves testing the internal logic, structure, and code of the application.
- Tester needs to have knowledge of the internal workings.
- Techniques:
    - Statement Coverage- execute all line at least once
    - Decision Coverage-Ensure T&F Outcomes in every decision(if statement)
    - Condition Coverage-Inside the decision T&F Tested
    - Branch Coverage- ensure all possible branches test(if else, if else)