Unit 1

1. **Explain the purpose of testing in software development.**

   Purpose of Testing in Software Development (5 Marks):

   1. **Error Detection – Helps to identify defects, bugs, and missing functionalities early, preventing failures after release.**
   2. **Validation and Verification – Ensures that the developed software meets user requirements and functions according to specifications.**
   3. **Quality Assurance – Improves overall quality by checking reliability, performance, usability, compatibility, and security.**
   4. **Cost and Risk Reduction – Early detection of errors reduces rework costs, prevents major failures, and lowers project risks.**
   5. **User Confidence and Satisfaction – Builds trust among stakeholders by delivering a stable, efficient, and user-friendly product.**
   6. **Ensures Maintainability – Properly tested software is easier to update, extend, and maintain in the future.**

2. **Describe any 3 dichotomies used in software testing with examples.**

1. Testing vs. Debugging

Aspect	Testing	Debugging
Definition	Process of finding errors or defects in the software	Process of locating and fixing the cause of errors
Performed By	Testers (QA team)	Developers
Goal	To identify defects	To remove defects
Execution	Involves executing the program with different inputs	Involves analyzing code, error logs, and fixing issues
Example	Running test cases that show incorrect output	Correcting the code that caused incorrect output

---

2. Black Box Testing vs. White Box Testing

Aspect	Black Box Testing	White Box Testing
Definition	Tests functionality without looking at internal code	Tests internal logic, structure, and paths of the code
Focus	Input-output behavior	Internal program structure
Knowledge Required	No programming knowledge	Programming knowledge required
Performed By	Testers	Developers/Testers with coding skills
Example	Checking login with valid/invalid inputs	Checking all conditions and loops in login code

---

3. Designer vs. Tester

Aspect	Designer	Tester
Role	Responsible for creating design of the software system	Responsible for validating and verifying the software
Focus	Functionality, architecture, and meeting requirements	Quality, correctness, and finding defects
Knowledge	Strong domain and design knowledge	Strong testing techniques and defect detection
Goal	Build the system as per specifications	Ensure the system is error-free and meets expectations
Example	Designing a database schema for a banking app	Checking whether transactions are processed correctly

3. Analyze the different types of bugs based on their taxonomy.
Bugs can be classified into different categories based on their nature and impact. Common types are:

1. **Logical Bugs – Errors in program logic or algorithms that produce incorrect results.**
   - ***Example:*** **Wrong formula used in interest calculation.**
2. **Syntax Bugs – Occur due to incorrect use of programming language grammar.**
   - ***Example:*** **Missing semicolon in Java code.**
3. **Runtime Bugs – Appear during execution, often due to invalid operations or environment issues.**
   - ***Example:*** **Division by zero, null pointer exception.**
4. **Interface Bugs – Arise from incorrect interaction between modules or systems.**
   - ***Example:*** **API mismatch between frontend and backend.**
5. **Performance Bugs – Cause slow response, memory leaks, or inefficiency in execution.**
   - ***Example:*** **Page load taking too long in a web application.**
6. **Security Bugs – Vulnerabilities that allow unauthorized access or attacks.**
   - ***Example:*** **SQL Injection or weak password storage.**

4. Construct a simple flow graph and explain its nodes and edges.
See in Notes

5. Explain how path predicates and path sensitizing are used in testing.
Path Predicates and Path Sensitizing in Testing (5 Marks):

1. **Path Predicate**
   - **A *path predicate* is a logical expression that represents the necessary conditions on input values for a particular execution path in the program to be taken.**
   - **It is derived from decision statements (like if, while) along the path.**
   - ***Example:*** **For a path passing through if (x > 0 && y < 5), the path predicate is (x > 0) ∧ (y < 5).**
2. **Path Sensitizing**
   - ***Path sensitizing*** **is the process of finding input data that satisfies the path predicate, so that the chosen path can actually be executed during testing.**
   - **It ensures that test cases are not just theoretical but practically executable.**
   - ***Example:*** **For the predicate (x > 0) ∧ (y < 5), test input x = 2, y = 3 sensitizes the path.**

3. **Usage in Testing**
    - **Helps design test cases that cover different execution paths.**
    - **Ensures thorough structural testing by executing feasible paths.**
    - **Aids in detecting hidden defects by checking complex condition combinations.**

6. **Apply path instrumentation on a simple code snippet and explain the result.**

Example Code

```
def maximum(a, b):
  if a > b:
    return a
  else:
    return b
```

Step 1: Flow Graph Construction

- **Node 1: Start and condition check a > b.**
- **Node 2: Return a.**
- **Node 3: Return b.**

Edges:

- **1 → 2 (if condition true),**
- **1 → 3 (if condition false).**

Step 2: Path Instrumentation

Path instrumentation means inserting extra statements (like counters or print statements) to record which path is taken during execution.

Instrumented code:

```
def maximum(a, b):
  print("At Node 1: Checking condition a > b")  # instrumentation
  if a > b:
    print("Path taken: 1 → 2 (return a)")    # instrumentation
    return a
  else:
    print("Path taken: 1 → 3 (return b)")    # instrumentation
    return b

# Test cases
print("Result:", maximum(7, 3))  # condition true
print("Result:", maximum(2, 5))  # condition false
```

Step 3: Execution Result

- **For (7, 3) → Output: Path taken: 1 → 2 → Result = 7.**
- **For (2, 5) → Output: Path taken: 1 → 3 → Result = 5.**

Explanation

- **Path Instrumentation helps in *observing which execution path is followed* by a given input.**

- **It validates test cases by ensuring that all feasible paths of the program have been executed at least once.**

👉 **Thus, path instrumentation makes hidden paths visible during testing.**

7. **Differentiate between basic and achievable paths in flow graph testing.**

Aspect	Basic Paths	Achievable Paths
Definition	Independent paths through the program that ensure complete branch coverage	All possible paths that can actually be executed during program run
Purpose	Used in basis path testing to guarantee minimum coverage of all edges	Represent the feasible execution flows a program may follow
Count	Limited and finite, based on *cyclomatic complexity*	Can be very large, sometimes infinite (due to loops)
Feasibility	Always feasible by design (chosen to be independent)	Some paths may be infeasible if conditions contradict
Example	For an if-else structure, two basic paths: one for *true*, one for *false*	If a loop exists, achievable paths include loop executed 0 times, 1 time, 2 times, etc.

👉 **In summary:**

- **Basic paths = *minimum independent set* to ensure test coverage.**
- **Achievable paths = *all possible executable flows* that inputs can trigger.**

8. **Discuss any two models for testing and their real-world application.**
Two Models for Testing and Their Applications (5 Marks):

---

1. V-Model (Verification and Validation Model)

- **Description:**
  - **Extension of the waterfall model where each development phase has a corresponding testing phase.**
  - **Emphasizes verification (design, requirements) and validation (unit, system, acceptance testing).**
- **Real-World Application:**
  - **Used in safety-critical systems like automotive software, avionics, and medical devices, where every stage must be validated before moving forward.**

---

2. Agile Testing Model

- **Description:**
  - **Testing is integrated continuously with development in short iterations (sprints).**
  - **Focuses on collaboration, quick feedback, and adapting to changes.**

- **Real-World Application:**
  - **Widely used in web and mobile app development, e.g., e-commerce platforms (Amazon, Flipkart), where features are delivered incrementally and tested continuously.**

👉 Summary:

- **V-Model = structured, phase-wise, used in critical systems.**
- **Agile Testing Model = iterative, flexible, used in dynamic application development.**

9. **Analyze the impact of severe bugs in real-time systems.**
   Impact of Severe Bugs in Real-Time Systems (5 Marks):
   1. **System Failure: Severe bugs can cause the entire system to crash, leading to downtime in mission-critical applications.**
      - ***Example:* A bug in an aircraft control system may cause autopilot malfunction.**
   2. **Safety Risks: In real-time systems, incorrect outputs or delays can endanger human life.**
      - ***Example:* A medical device bug giving wrong dosage to patients.**
   3. **Data Loss and Corruption: Bugs may lead to loss or corruption of sensitive data processed in real-time.**
      - ***Example:* Stock trading systems losing transaction records.**
   4. **Performance Degradation: Timing constraints may be violated, resulting in missed deadlines.**
      - ***Example:* Delays in traffic signal systems causing accidents.**
   5. **Financial and Reputational Damage: Failures in real-time banking or telecom systems can cause monetary loss and loss of trust.**
      - ***Example:* Bugs in ATMs or online payment systems deducting wrong amounts.**

10. **Illustrate the steps involved in applying path testing.**
    Steps Involved in Applying Path Testing (5 Marks):
    1. **Construct the Flow Graph**
       - **Represent the program with nodes (statements/conditions) and edges (control flow).**
       - ***Example:* Draw flow graph for an if-else structure.**
    2. **Determine Cyclomatic Complexity (V(G))**
       - **Use formula: V(G) = E – N + 2, where *E* = *edges* and *N* = *nodes*.**
       - **Cyclomatic complexity gives the number of independent paths.**
    3. **Identify Independent Paths**
       - **Select a basis set of linearly independent paths that ensure complete branch coverage.**
       - **Each path should introduce at least one new edge not in previous paths.**
    4. **Derive Path Predicates**
       - **Write logical conditions for each path (based on decisions like if, while).**
    5. **Perform Path Sensitization**

- **Choose input values that satisfy the path predicates, ensuring each independent path can be executed.**

6. **Execute Test Cases**
    - **Run the program with chosen inputs to traverse all selected paths and detect possible defects.**