1) **what is regression testing?**

Ans:Regression testing is like checking to make sure recent changes to software haven't caused old things to break. It ensures that new updates or fixes don't accidentally mess up what was working fine before. This is done by retesting specific parts of the software to catch any unintended issues and make sure everything still works as intended. Automated tools are often used to make this process faster and more efficient, especially in environments where changes are made frequently.

2) **Who is responsible for unit testing?**

Ans:In software development, the responsibility for unit testing typically falls on the shoulders of the developers themselves. The individuals who write the code are usually responsible for creating and running unit tests to ensure that each component or unit of code behaves as intended. Unit testing is a fundamental aspect of the development process, and developers perform it during the coding phase before the code is integrated into the larger software system.

3) **What is Bug Model?**

Ans:

4) **what is a race condition bug?**

Ans:A race condition bug happens when the order of events in a program causes unexpected issues. Imagine you have two events, A and B. The program expects A to happen before B, but sometimes, due to specific conditions, B occurs first. If the program doesn't handle this situation correctly, it leads to a race condition bug, causing errors or failures. It's like expecting a light switch to turn on the lights, but if you quickly flick the switch off and on, the lights might not behave as expected due to the rapid change in events.

5) **Write down the software Devvelopment Life Cycles steps?**

Ans:

Planning: Define project goals, scope, timelines, and resources.

- **Analysis**: Gather and analyze requirements from stakeholders.
- **Design**: Create a detailed system design based on requirements.
- **Implementation**: Write code and develop the software based on the design.
- **Testing**: Conduct thorough testing to identify and fix bugs.
- **Deployment**:Release the software for public use.
- **Maintenance**: Provide ongoing support, updates, and bug fixes.

6) **what is cyclomatic complexity?**

Ans:Cyclomatic complexity is a way to measure how complex and twisty a computer program's path is. The higher the complexity, the more convoluted the code might be. It's like counting the decision points (if statements, loops) to figure out how many different paths your program can take. Lower complexity is usually better, making code easier to understand and test.

7) what is system testing?

Ans:System testing is checking if the whole software system works as expected. It tests the entire application, including all components and interactions, to make sure it meets the specified requirements. It's like giving the software a final exam to ensure it functions correctly as a complete and integrated system.

8) what are testing bugs?

Ans:Testing bugs refer to defects or issues in software that are discovered during the testing phase of the software development life cycle. These bugs can manifest in various forms and affect the functionality, performance, security, or user experience of the software. Testing is a crucial step in the software development process to identify and rectify such bugs before the software is released to users.

Here are some common types of testing bugs:

i)Functional Bugs
ii)Performance Bugs
iii)Compatibility Bugs
iv)Security Bugs
v)Usability Bugs
vi)Interface Bugs

9) what is inspection testing?

Ans: "Inspection testing" is not a standard term in software testing. If you are referring to code inspection or review, it involves a thorough examination of code by peers to find defects, improve quality, and ensure adherence to coding standards.

10) what will happen if we test the whole system directly?

Ans: Testing the whole system directly may lead to difficulty in identifying and fixing specific issues, making it challenging to isolate and address individual components' problems efficiently.

11) what type of test plan can we make after verifying the SRS?

Ans:After verifying the Software Requirements Specification (SRS), you can create a System Test Plan to outline the approach, scope, resources, and schedule for testing the entire system to ensure it meets the specified requirements

12) what is High-Level design of a software system?

Ans: The High-Level Design of a software system provides an abstract representation, outlining major components, their interactions, and the overall architecture without delving into detailed implementation.

13) Who are the stakeholders of regression testing?

Ans:Stakeholders of Regression Testing: Developers, Testers, Product Managers, and System Users.

14) Give 2 example of validation testing technique?

Ans:
1. User Acceptance Testing (UAT)
2. Compliance Testing

15) What is software Crisis?

Ans: Software Crisis refers to the challenges and issues that arise due to the increasing complexity of software development, resulting in difficulties in meeting deadlines, maintaining quality, and managing costs.

16) What is static testing?

Ans: Static testing is a method used to evaluate the structure and characteristics of source code or design specifications without actually running the code. Unlike dynamic testing, where the code is executed, static testing focuses on examining the code's syntax and design without actively running it.

17) What is dynamic testing?

Ans: Dynamic testing is a method used to test software by actively running the code with various inputs provided by users and checking the resulting outcomes. Unlike static testing, which examines the code without executing it, dynamic testing involves the actual execution of the software

18) what is exhaustive testing? why should it be avoided?

Ans:Exhaustive testing, also known as complete testing or brute-force testing, is a testing approach where every possible combination of inputs, states, and scenarios for a software application are tested. The idea is to test all possible combinations to ensure that no bugs or defects are overlooked. While this may sound thorough, exhaustive testing is often impractical and, in many cases, impossible. Here's why it should be avoided:

Infeasibility: For non-trivial software applications, the number of possible combinations of inputs and states can be incredibly large or infinite. Testing all possible combinations would require an impractical amount of time, resources, and effort.

Time and Cost Constraints: Exhaustive testing is time-consuming and expensive. Given the dynamic nature of software development and the need for rapid releases, spending an excessive amount of time on testing every possible combination may not be feasible within project timelines and budgets.

Diminishing Returns: The effort required to test every possible combination does not necessarily result in a proportional increase in the discovery of defects. Many defects can be identified through a well-designed and targeted subset of test cases, making exhaustive testing an inefficient use of resources.

Redundancy: Some test cases may cover similar scenarios or paths through the application. Testing all possible combinations can lead to redundant testing efforts, as certain paths may have already been adequately covered by other test cases.

Complexity: The complexity of managing and executing a massive number of test cases in an exhaustive testing scenario can be overwhelming. Test case design, execution, and analysis become increasingly difficult and error-prone.

Resource Constraints: Limited resources such as time, personnel, and testing environments make it practically impossible to execute exhaustive testing in real-world software development projects.

19) Write down the difference between bottom up and top down integration testing?

Ans:

Feature	Bottom-Up	Top-Down
Integration Direction	Low-level to high-level	High-level to low-level
Data Flow	Bottom-up	Top-down
Complexity	More complex	Less complex
Data Intensity	More data-intensive	Less data-intensive
Error Detection	Early detection of errors in low-level modules	Errors in lower-level modules may go undetected
Testing Effort	More effort initially, less effort later	Less effort initially, more effort later
Suitable Scenarios	Systems with well-defined interfaces and independent modules	Systems with complex control flows and tightly coupled modules

20) "Testing is not a single phase in SDLC" discuss in brief?

Ans:

- **Requirements Analysis:**
  Testing begins with the analysis of requirements. This involves reviewing specifications, identifying testable components, and understanding the expected behavior of the software.
- **Design Phase:**
  Test planning starts during the design phase. Testers create a testing strategy, define test cases, and plan for various types of testing, such as unit testing and integration testing.
- **Coding and Unit Testing:**

Developers write code for individual units or components. Concurrently, unit testing is performed to ensure that each unit functions as intended in isolation. This phase involves identifying and fixing defects at the unit level.

- **Integration Testing:**
  As units are integrated to form larger modules or subsystems, integration testing is conducted. This phase aims to detect issues related to the interaction between different components.
- **System Testing:**
  Once integration is complete, the entire system undergoes testing. This phase focuses on validating that the integrated components work together as a cohesive unit, meeting the specified requirements.
- **Acceptance Testing:**
  Acceptance testing is performed to ensure that the software satisfies the business requirements and is ready for deployment. This phase involves user acceptance testing (UAT), where end-users validate the software against their needs.
- **Deployment and Maintenance:**
  Even after deployment, testing remains crucial. Maintenance testing is conducted to address any issues that arise in the production environment, apply patches, and introduce new features or enhancements.

21) Explain the elements of security testing?

Ans: Security testing is a crucial aspect of software testing that focuses on identifying vulnerabilities and weaknesses in a system to ensure that it is resistant to security threats and attacks. The elements of security testing include various aspects that assess the security posture of a software application. Here are the key elements:

Authentication Testing:

- Verify the effectiveness of user authentication mechanisms.
- Test for weak or easily guessable passwords, password policies, and multi-factor authentication.

Authorization Testing:

- Assess whether users have appropriate access rights and permissions.
- Check for proper segregation of duties and ensure that users can only access the resources they are authorized to use.

Data Confidentiality Testing:

- Evaluate the protection of sensitive information from unauthorized access.
- Test encryption methods, secure storage practices, and data transmission security.

Integrity Testing:

- Check if data remains unaltered during storage or transmission.
- Verify the integrity of data by testing against unauthorized modifications or tampering.

Session Management Testing:

- Evaluate the security of user sessions and session-related functionalities.
- Check for session timeouts, session hijacking vulnerabilities, and proper session termination.

Input Validation Testing:

- Ensure that input data is properly validated to prevent common security issues such as SQL injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).

Security Configuration Testing:

- Assess the security settings and configurations of servers, databases, and other components.
- Verify that default settings are changed, unnecessary services are disabled, and security features are properly configured.

Error Handling Testing:

- Evaluate how the system handles errors and exceptions.
- Ensure that error messages do not reveal sensitive information and that users are provided with minimal information in case of failure.

Security Patch Management Testing:

- Assess the system's ability to detect, apply, and manage security patches.
- Verify that the software is up-to-date with the latest security patches and updates.

Penetration Testing:

- Conduct simulated attacks to identify vulnerabilities that could be exploited by real attackers.
- Test the system's resilience against external threats and attempt to compromise the system's security.

Security Code Review:

- Analyze the source code for security vulnerabilities.
- Manual or automated code reviews help identify issues like insecure coding practices, backdoors, and other potential security risks.

Security Compliance Testing:

- Evaluate the application against industry standards, regulatory requirements, and security best practices.
- Ensure that the software complies with relevant security standards and regulations.

22) **Draw the diagram of the life cycle of a bug.**

Ans:

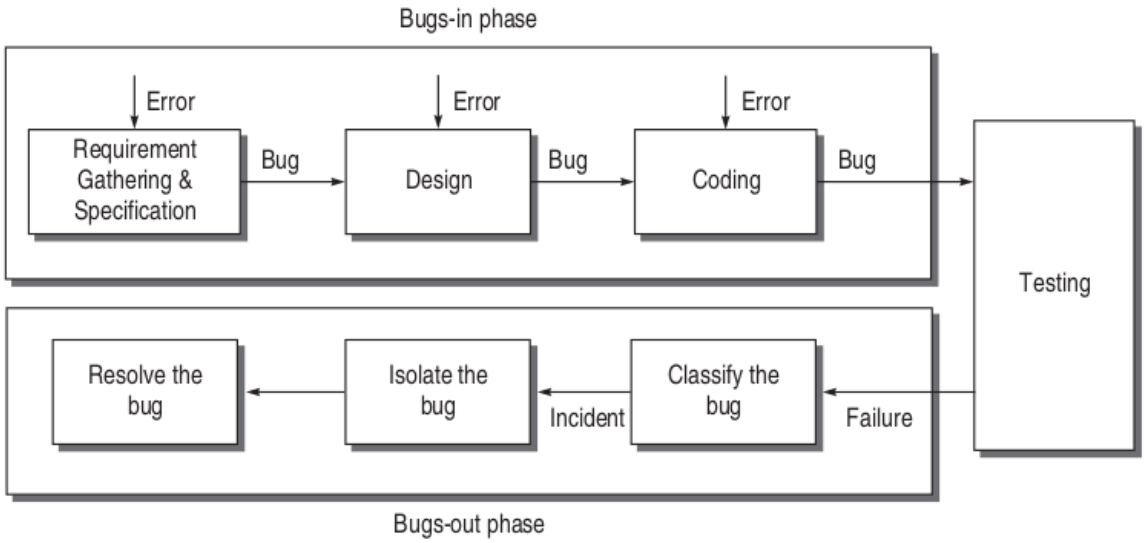

Figure 2.4 Life cycle of a bug

23) Write a short note on the state of a bug in its life cycle.

Ans:

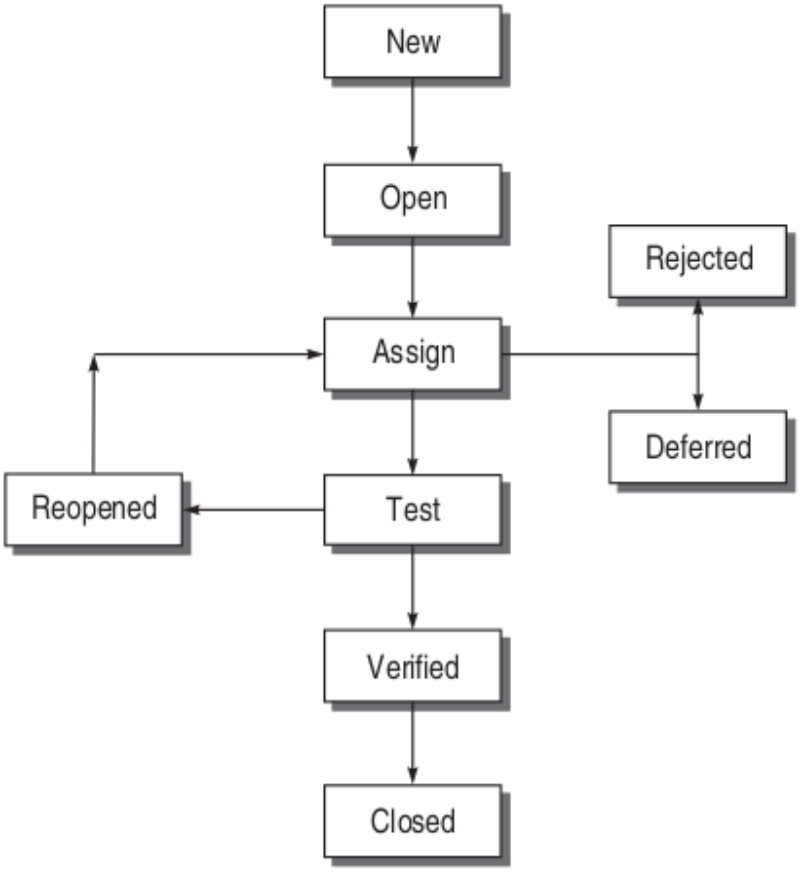

Figure 2.5 States of a bug

24) Distinguish between verification and validation activities.

Ans: Verification ensures that the product is built correctly by adhering to specified requirements. It involves reviews, inspections, and walkthroughs.

Validation ensures that the right product is built by assessing if it meets the customer's needs. Testing, including dynamic analysis, is a common validation activity.

25) Classify bug based on criticality.

Ans:

Critical Bugs: This type of bugs has the worst effect such that it stops or hangs the normal functioning of the software. For example, in a sorting program, after providing the input numbers, the system hangs and needs to be reset.

Major Bug: This type of bug does not stop the functioning of the software but it causes afunctionality to fail to meet its requirements as expected. For example,in a sorting program, the output is being displayed but not the correct one.

Medium Bugs: Medium bugs are less critical in nature as compared to critical and major bugs. If the outputs are not according to the standards or conventions, e.g.redundant or truncated output, then the bug is a medium bug.

Minor Bugs: These types of bugs do not affect the functionality of the software. These are just mild bugs which occur without any effect on the expected behaviour or continuity of the software. For example, typographical error or misaligned printout.

26) Write the difference between verfication and validation activities.

Ans:

Verification	validation
Are you building it right?	Have you build the right thing?
Check wheater an artifact conforms its previous artifact.	Check the final product against specification.
Done by developer.	Done by tester.
Concern with phase containment of errors.	Aim is to make final product error free.
Methods involves review ,inspection,unit testing and integration testing.	Involves system testing.
Static and dynamic activities.	Only dynamic.

27) **Draw and explain the "V" testing model.**

Ans:

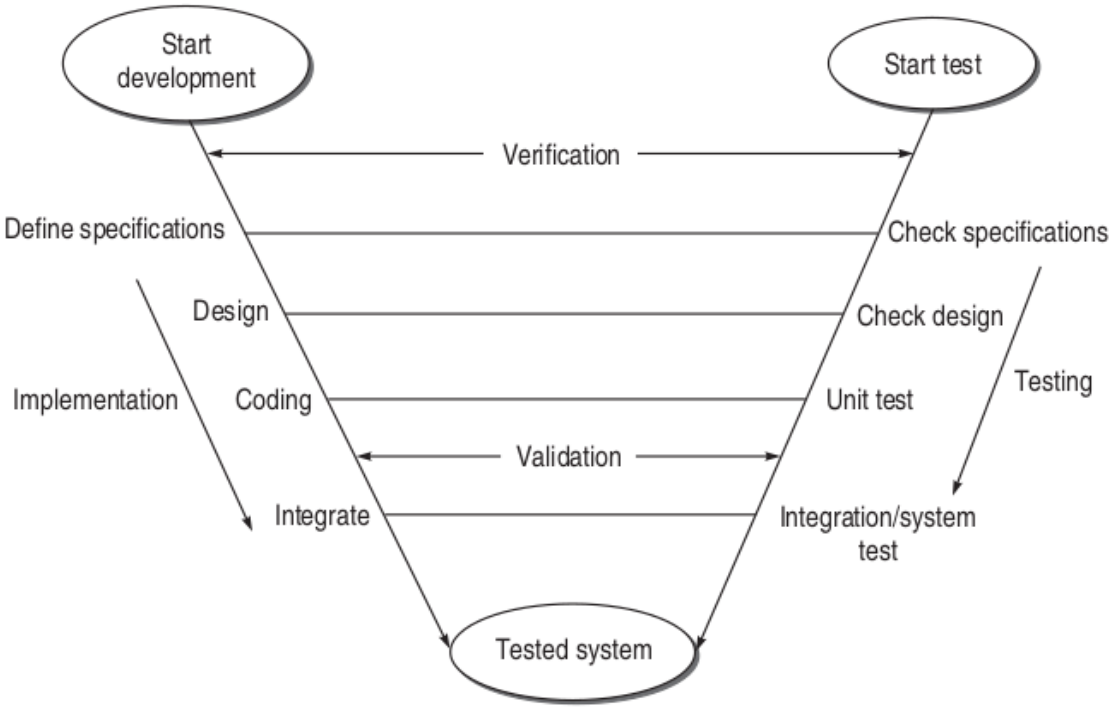

Figure 2.13 Expanded V-testing model

In V-testing concept [4], as the development team attempts to implement the software, the testing team concurrently starts checking the software. When the project starts, both the system development and the system test process begin. The team that is developing the system begins the system development process and the team that is conducting the system test begins planning the system test process, Both teams start at the same point using the same information. If there are risks, the tester develops a process to minimize or eliminate them.

28) **What is the difference between debugging and testing?**

Ans: Testing is the systematic process of executing a program or system to find defects or ensure that it meets specified requirements. Debugging, on the other hand, is the subsequent process of identifying, isolating, and rectifying the specific defects or issues detected during testing or development. While testing is about validating overall functionality, debugging is the focused effort to troubleshoot and refine the code based on identified problems.

29) **What is dynamic testing technique? State different types of dynamic testing.**

Ans:

Dynamic Testing Techniques

Dynamic testing refers to software testing techniques that involve executing the software code to assess its functionality, performance, and usability. Unlike

static testing, which analyzes the code without running it, dynamic testing provides insights into the actual behavior of the software in a real-world environment.

Here are some of the different types of dynamic testing:

- **Black-Box Testing:**
  Focuses on testing the software from the user's perspective without any knowledge of the internal code structure.
- **White-Box Testing:**
  Focuses on testing the internal structure of the software code and ensuring it functions as intended.
- **Performance Testing:**
  Focuses on assessing the non-functional aspects of the software,such as its responsiveness,stability,and scalability.
- **Usability Testing:**
  Focuses on evaluating the user experience of the software, including its ease of use,learnability,and accessibility.
- **Security Testing:**
  Focuses on identifying and mitigating security vulnerabilities in the software that could be exploited by attackers.

30) Advantage of dynamic testing?

Ans:Advantages of dynamic testing are-

- It discloses very difficult and complex defects.
- It detects the defects that can't be detected by static testing.
- It increases the quality of the software product or application being tested.
- Dynamic testing detects security threats and ensure the better secure application.
- It can be used to test the functionality of the software at the early stages of development.
- It is easy to implement and does not require any special tools or expertise.
- It can be used to test the software with different input values.
- It can be used to test the software with different data sets.
- It can be used to test the software with different user profiles.
- It can be used to test the functionality of the code.
- It can be used to test the performance of the code.
- It can be used to test the security of the code.

31) Disadvantages of dynamic testing.

Ans:Disadvantages of dynamic testing are-

- It is a time consuming process as in dynamic testing whole code is executed.
- It increases the budget of the software as dynamic testing is costly.
- Dynamic testing may require more resources than static testing.
- Dynamic testing may be less effective than static testing in some cases.
- It is difficult to cover all the test scenarios.
- It is difficult to find out the root cause of the defects.

32) Advantages of static testing.

Ans:**Detecting Defects Early:**When applied early in the software development lifecycle, static testing enables the early detection of defects. Using static testing, we can identify defects in requirements, design specifications reviews or in, backlog refinement, etc. Defects found early are often much cheaper to remove than defects found later in the lifecycle. A defect in the requirements specification may propagate itself to the design, the code, and even the test cases. Defects discovered after the software is deployed to production are very costly.

Preventing Defects: Static testing prevents defects in design or coding by uncovering inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies in requirements.

Easy to find Defects: Static Testing identifies defects that dynamic testing does not detect easily. Review can find omissions. E.g., finding a missing requirement that may have been unlikely to find during dynamic testing.

Increase Development Productivity: Static Testing increases development productivity due to quality and understandable documentation, improved design, and more maintainable code.

Improve Team Communication: Static Testing improves communication between team members while participating in reviews.

Reduce Cost and Time: Static testing reduces development, testing, and overall organization cost. Defect found and fixed during static testing techniques is almost always much cheaper for the organization than dynamic testing. Especially considering the additional costs associated with updating other work products and performing confirmation and regression testing. It reduces the total cost of quality over the software's lifetime due to fewer failures later in the lifecycle or after delivery into operation.

33) Disadvantages of static testing.

Ans:

- Demand great amount of time when done manually
- Automated tools works with few programming languages
- Automated tools may provide false positives and false negatives
- Automated tools only scan the code
- Automated tools cannot pinpoint weak points that may create troubles in run-time

34) Write a short note on the levels of dynamic testing.

Ans:

- **Unit testing:** Unit testing is the process of testing individual software components or "units" of code to ensure that they are working as intended. Unit tests are typically small and focus on testing a specific feature or behavior of the software.
- **Integration testing:** Integration testing is the process of testing how different components of the software work together. This level of testing typically

involves testing the interactions between different units of code, and how they function when integrated into the overall system.

- **System testing:** System testing is the process of testing the entire software system to ensure that it meets the specified requirements and is working as intended. This level of testing typically involves testing the software's functionality, performance, and usability.
- **Acceptance testing:** Acceptance testing is the final stage of dynamic testing, which is done to ensure that the software meets the needs of the end-users and is ready for release. This level of testing typically involves testing the software's functionality and usability from the perspective of the end-user.
- **Performance testing:** Performance testing is a type of dynamic testing that is focused on evaluating the performance of a software system under a specific workload. This can include testing how the system behaves under heavy loads, how it handles a large number of users, and how it responds to different inputs and conditions.
- **Security testing:** Security testing is a type of dynamic testing that is focused on identifying and evaluating the security risks associated with a software system. This can include testing how the system responds to different types of security threats, such as hacking attempts, and evaluating the effectiveness of the system's security features.

35) **Explain the following terminologies:failure,error,bug.**

Ans:

Failure:

- **Definition**: A failure occurs when the software does not perform as expected and deviates from its intended behavior.
- **Explanation**: In simpler terms, when the end user experiences something wrong or unexpected in the software, it's referred to as a failure. Failures are observable issues from the user's perspective, such as a feature not working or producing incorrect results.

Fault/Defect/Bug:

- **Definition**: A bug is a fault in the code or design of the software that causes it to behave unexpectedly or produce incorrect results.
- **Explanation:** Bugs are a type of error that leads to a failure in the software. The term "bug" is often used more broadly to refer to any unexpected behavior or flaw in the software. Bugs can be the result of coding mistakes, incorrect algorithms, or issues in the software design.

Error:

- **Definition**: An error is a mistake made by a human, resulting in a fault or flaw in the code or design of the software.
- **Explanation:** Errors are part of the development process and are introduced by developers. They can manifest as syntax errors, logic

errors, or other mistakes that cause the software to behave unexpectedly. Errors are the root cause of faults in the software.

36) A program reads two number A and B within the range (0,100] and calculate the GCD of those numbers.Design test case for this program using BVC,robust testing methods.

Ans:Since there are two variables, a and b, the total number of test cases will be 4n + 1 = 9. The set of boundary values is shown below:

	a	b
Min value	0	0
Min value+	1	1
Max value	100	100
Max value-	99	99
Nominal value	50	50

Using these values, test cases can be designed as shown below:

Test Case ID	a	b	Expected Output
1	0	50	Undefined
2	1	50	1
3	100	50	50
4	99	50	1
5	50	0	50
6	50	1	1
7	50	100	50
8	50	99	1
9	50	50	50

6.Passenger who travel more than 50,000 km per calendar year and in addition ,pay cash for tickets or have been travelling regularly for more than eight years are to receive a free round trip ticket around india.passengers who travel less than 50,000 km per calender year have been availing railway services regularly for more than eight years also get a free round ticket around india . Design test cases for this system using decesion table testing.

Ans:Decision table testing is a technique used to test systems with complex business rules or decision-making processes. In this case, the decision table can help us design

test cases based on the specified conditions for providing free round trip tickets to passengers. Let's create a decision table based on the given criteria:

Condition 1: Distance Traveled (km)	Condition 2: Payment Method	Condition 3: Years of Regular Travel	Decision: Free Round Trip Ticket?
> 50,000	Cash	Any	Yes
>50,000	Not cash	>8	Yes
≤ 50,000	Any	>8	Yes
≤ 50,000	Any	≤ 8	Yes

13)what is the difference between debugging and testing?

Ans:

Testing	Debugging
Verify that software meets specified requirements.	Identify and fix errors or defects in the code.
Ensure correct behavior and adherence to specs.	Locate and correct specific issues in the code.
Design and execute test cases, evaluate results.	Use debugging tools, step through code, inspect variables, make changes.
Occurs at various stages (unit, integration, system testing).	Primarily after testing, during development or post-deployment.
Often done by dedicated testers.	Primarily done by developers.