Q.1 Attempt the following (any TWO) [10]

Q.1(a) Define the terms error, bug defects, failure and fault. [5]

Ans.:
- A failure is of software is non-fulfillment of requirements.
- A discrepancy between actual result or behavior (identified while executing test) and expected results or behavior (Identify in requirement specifications).
- Failure in a software means a software does not perform functional requirements. The failure occurs in software because of faults. Every faults or defect or bug present in the software since it was developed or changed.
- A failure has its roots in a fault in the software. These faults are also called as defects or internal errors. Programmers generally use a term bug for the faults. Example of faults can be a communication gap between customer and developer or customer not able to specify all there requirements or customer requirements are misinterpreted or programmers might be wrongly programmed or wrong programming language selection or forgotten code in application.
- It is possible that the fault is hidden by one or more other faults present in different parts of application. In that case, a failure only occurs after masking defects that have been corrected.
- The cause of fault or defect is an error or mistake by person by environmental conditionals like radiation, magnetism, etc. that introduce hardware problems.

Q.1(b) What are the causes of software defects? [5]

(A)
- If someone makes an error or mistake in using the software, this may lead directly to a problem.
- People also design and build the software and they can make mistakes during the designing and building phase.
- These mistakes means that there are flaws in the software itself. These are called as defects or bugs or faults.
- Error – A human action that produces an incorrect result.
- Defect (bug, fault) - A flaw in a component or system that can cause the component or system to fail to perform its required function.
  A defect is an error in coding or logic that causes a program to malfunction or to produce incorrect/unexpected results. A defect is said to be detected when a failure is observed.
- Failure – Deviation of the component or system from its expected result or service.
  Failure is the inability of a system or a component to perform its required functions within specified performance requirements. Failure occurs when fault executes.

Defects and failures arise from :
- Errors in the specification, design and implementation of the software.
- Errors in use of the system
- Environmental conditions
- Intentional damage
- Potential consequences of earlier errors, intentional damage, defects or failures.

Q.1(c) State and explain the principles of software testing. [5]

(A) Principles of software testing

(1) Testing shows the presence of defects, not their absence. Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

(2) Exhaustive testing is not possible. Testing everything is not feasible except for trivial cases, instead of exhaustive testing, we use risks & priorities to focus testing efforts.
(3) Testing activities should start as early as possible testing activities should start as early as possible in the software of or system development life cycle & should be focused on defined objectives.
(4) Defects tend to cluster together. A small number of modules contain most of the defects discovered during prerelease testing or show the most operational failures.
(5) The pesticide paradox, If the test are repeated over & over again, eventually the same set of test cases will no longer find any new bugs. To overcome this pesticide paradox the test cases need to be regularly reviewed & revised & new & different tests need to be written to exercise different parts of the software or system to potentially find more defects.
(6) Test is context dependent, Testing is done differently in different dependent context. e.g. safety critical software in tested differently from as e-commerce site.
(7) The fallacy of assuming that no failures means a useful system. Finding & fixing defects does not help if the system build is unusable & does not fulfill the users needs & expectations.

Q.1(d) Explain psychology of testing. [5]

(A) Psychology of Testing

- The mindset we want to use while testing and reviewing is different from the one we use while analyzing or developing.
- By this we mean that, when we are building something we are working positively to develop a product that meets some need. However, when we test or review a product, we are looking for defects in the product and thus are critical of it.
- It is not like that tester cannot be a programmer, or that a programmer cannot be a tester, although they often are separate roles. In fact, programmers are testers - they test the components which they build, and the integration of the components into the system. With the right mindset, programmers can test their own code; indeed programmers do test their own code and find many problems, resolving them before anyone else sees the code. Business analysis and marketing staff should review their own requirements. System architects should review their own designs.
- However, we all know it is difficult to find our own mistakes. So, business analysts, marketing staff, architects and programmers often rely on others to help test their work. This other person might be a fellow analyst, designer or developer. A person who will use the software may help test it. Testing specialists - professional testers - are often involved. In fact, testing may involve a succession of people each carrying out a different level of testing. This allows an independent test of the system.
- Early in the life cycle, reviews of requirements and design documents by someone other than the author helps find defects before coding starts and helps us build the right software. Following coding, the software can be tested independently. This degree of independence avoids author bias and is often more effective at finding defects and failures.

Q.2 Attempt the following (any TWO) [10]

Q.2(a) Describe the purpose of confirmation testing and regression testing. [5]

(A) Confirmation Testing (re-testing) :

When doing confirmation testing, it is important to ensure that the test is executed in exactly the same way as it was the first time, using the same inputs, data and environments. When a test fails and we determine that the cause of the failure is software defect, the defect is reported and we can expect a new version of the software that has had the defect fixed. In this case we will need to execute the test again to confirm that indeed fixed. This is known as confirmation testing.

Regression Testing :

Regression testing involves executing test cases that have been executed before. For regression testing, the test cases probably passed the last time they were executed.

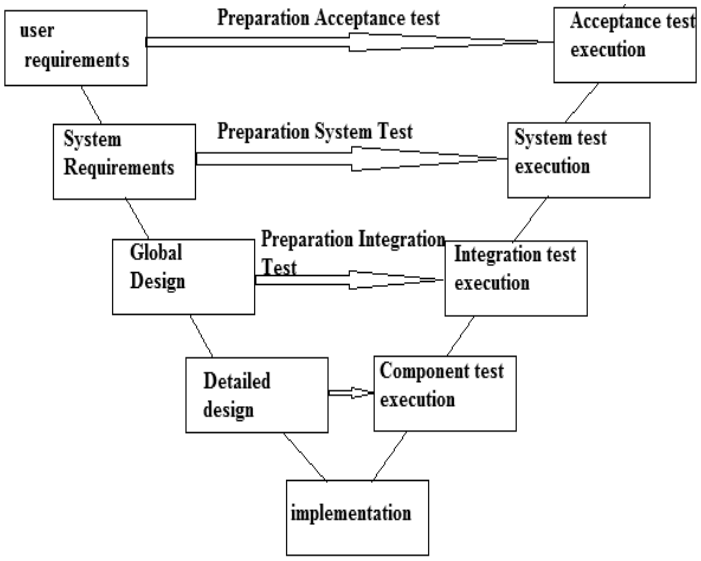
The fix may have introduced or uncovered a different defects elsewhere in software. The way to detect these 'unexpected side–effects' of fixes is to do regression testing.

Regression tests are executed whenever the software changes, either as a result of fixes or new or changed functionality. It is also a good idea to execute them when some aspect of the environment changes, for example when a new version of a database management system is introduced or a new version of a source code compiler is used.

Q.2(b) With the help of diagram explain V Model. [5]

(A) V-Model

- **V-model** was developed to address some of the problem experienced using the traditional waterfall approach. Defects were found too late in the life cycle, as testing was not involved until the end of the project.
- **The V-model** provides guidance that testing needs to begin as early as possible in the lifecycle.
- Testers need to work with developers and produce a set of test deliverables.
- The work products produced by system analyst and developers during development are the basis of the basis of testing in one or more levels.
- By starting test design early, defects are often found in the test basis documents.
- The V-model illustrates how the testing activities can be integrated into each phase of the life cycle.



V –Model

- A common V-Model uses four test levels. These are as follows :
  - *Component testing* – searches for defects in and verifies the functioning of software components (i.e. modules, classes etc.) that are separately testable.
  - *Integration testing* – tests interfaces between components, interaction to different parts of a system such as an operating system, file system and hardware and interfaces between systems.
  - *System testing* – concerned with the behavior of the whole system. The main focus of system testing is verification against specified requirements.
  - *Acceptance testing* – validation testing with respect to user needs, requirements and business process conducted to determine whether or not to accept system.

Q.2(c) Explain functional & non-functional testing. [5]

(A) A test type is focused on a particular test object which could the testing of function to be performed component or system.

A non–functional quality characteristic such as reliability useability; A structure or architecture of system or component; or related to change i.e. confirming the defects have been fixed and looking for unintended changes. i.e. Regretion Testing. Depending on test objectives, a different testing will be organised.

1. **Testing of function [functional testing]**
   The function of system is "what is does." This is typically describe in requirement specification, functional specification & in use cases.
   There may be some functions That are assumed to be provided that are not documented as well as not the part of requirement for system functional tests are based on these functions, describe in documents or understood ny the testor. Functional testing considers the specified behaviour & is often also reffered as Black Box testing or Behavaioural or Performance testing.
   Testing functionality can be done from two prospective Requirement based or Business Process Based.
   (A) Req. Based testing uses a specification of the functional Requirements for the system as the basis for designing test cases. A good way to start is to use the table of content of the requirement specification as an initial test inventory or list of items to test. As a testor we should also priorities a requirement based on risk criteria and use these to prioritize the test. These will ensure that the important & critical test are included in testing efforts.
   (B) Business Process Base testing uses knowledge of business processes. Use cases are very useful basis for test cases for business prospectives.
   The technique use for functional testing are often specification based but experience based technique can be used as a part of test design in functional testing, a model may be developed such as state transition model re use case model.

2. **Non–Functional Testing [Testing of s/w product]**
   1. A second target for testing is testing of the quality characteristics or non-functional attributes of the system here we are interested in "How well or fast something is done". We are testing something that we need to measure on scale of measurement e.g. time to respond. Non-functional testing, as functional testing is performed at all test levels. No–Functional testing includes performance testing, load, stress usability, maintenance, reliability & portability testing. It is the testing of "How well" the system, work.

   To the characteristics & sub characteristics of the software to be tested under non-functional testing are as follows:

   1. Functionality Sub–characteristics :
      Suitability, Accuracy, Security, Interoperability, complaints. These deals with functional testing.
   2. reliability – Sub characteristics –
      Maturity, facect Tolerance, Recoverability & complaints.
   3. Usability – Sub characteristics –
      Understandability, Learnability, Operability, attractiveness.
   4. Efficiency – Which is divided into Behavior & resource Utilisation.
   5. Maintainability : Sub characteristics :
      Analyzability, Changeability, Stability, Testability,
   6. Portability : Sub characteristics :
      Adaptability, install ability, Co–existence, Replaceability.

Q.2(d) Explain Agile model. [5]

(A) Agile Model

1. Extreme programming is currently one of the most well known agile development life cycle model.
2. The methodology claims to be more human friendly than the traditional development method.
3. Using Agile model, Developer can develop simple and Interesting GUI for S/W.

Some of the characteristics of XP are:

- It promotes the generation of business stories to define functionality.
- It demands an on side costumer for continues feedback and to define and carry out functional test.
- It promotes pair programming and shared code ownership among developers.
- It states that component test scripts shall be written before code is written and those test should be automated.
- It states that integration and testing of code shall happen several times a day.
- It always states that we should always implement simplest that should meet solutions to the today's problem.

Q.3 Attempt the following (any TWO) [10]

Q.3(a) Explain phases of review process. [5]

(A) Phases of formal review

In contrast to informal reviews, formal reviews follow a formal process. It consists of six main steps :

1. Planning
2. Kick-off (Meeting)
3. Preparation
4. Review meeting
5. Rework
6. Follow-up

1. Planning :

- The review process for a particular review begins with a 'request for review' by the author to the moderator (or inspection leader). The leader is often assigned to take care of the scheduling (date, time, place and invitation) of the review.
- During overall planning, the management must decide which documents in the software development process are subject to which review technique.
- During planning of the individual review, the review leader selects technically competent staff and assembles a review team. In cooperation with the author of the document to be reviewed, he makes sure that the document is in a reviewable state. i.e. it is complete enough and the work on it has been finished.

2. Overview (Kick-off meeting)

- The overview serves to provide those involved in the review with all necessary information. This can happen through a written invitation or a first meeting when the review meeting for planning is organized.
- The purpose is to share information about the document to be reviewed.
- If the involved people are not familiar with the domain or application area of the review object, then there can be a short introduction to the material, as well a description of how it fits into the application or environment.
- Role assignments, checking rate, the pages to be checked and possible other questions are also discussed during this meeting. The distribution of document under review, source documents and other related documents can also be done during the kick-off.
- In addition to review object, those involved must have access to the other document. These include the documents that must be used to decide if a particular statement is correct or not. Such documents are called Base documents or baseline.
  - Distributing documents
  - Explaining the objectives, process and documents to the participants

3. Preparation :

- In this stage, intensive study of the review object happens.
- The members of the review team must prepare individually for the review meeting. A successful review meeting is only possible with adequate preparation. The reviewer intensively studies the review object and checks it against the document given as a basis for it. They note deficiencies, questions or comments.
- The annotated document is to be given to the author at the end. Using checklist during this phase can make reviews more effective. For eg. A specific checklist based on perspectives such as user, maintainer, tester or a checklist for typical coding problems.
- A critical success factor for a thorough preparation is the number of pages checked per hour. This is called as checking rate. The optimum checking rate is the result of multiple factors such as type of document, its complexity, the number of related documents and the experience of the reviewer.
- Usually the checking rate is 5 to 10 pages per hour. But may be much less for the formal inspection – one page per hour.

4. Review meeting :

- The review meeting is led by a review leader or moderator. Managing and participating in the review requires good people management skills in order to protect the participating people and motivate them to best contribute to the review.
- The review leader must ensure that all experts will be able to express their opinion without fear that the product will be evaluated and not the author, and conflicts will be prevented.
- The objective of the review meeting is to decide if the review object has met the requirements and complies with the standards, as well as to find defects. The result is a recommendation to accept repair. All the reviewers should agree upon the findings of this evaluation and the general result.
- The meeting consists of the following elements –
  - Logging phase
  - Discussion phase
  - Decision phase
- **Logging phase** – during this phase, the issues e.g. defects that have been identified during the preparation are mentioned page by page, reviewer by reviewer and are logged either by the author or a scribe.
  - If an issue needs discussion, the item is logged and then handled in the discussion phase.
  - A detailed discussion on whether an issue is a fault or not is very meaningful. In spite of the opinion of the team, a discussed and discarded defect may well turn out to be a real one during rework.
  - Every defect and its severity should be logged.
    - **Critical** – defect will cause downstream damage, the scope and the impact of the defect is beyond the document under inspection.
    - **Major** – defects could cause a downstream effect (e.g fault in a design can result in a n error in the implementation)
    - **Minor** – defects are not likely to cause downstream damage (e.e. non-compliance with the standards)
  - During the logging phase, the focus is on logging as many defects as possible within a certain time. To ensure this, the moderator tries to keep a good logging rate.
- **Discussion phase :**
  - The issue classified as discussion items will be handled during this meeting phase.
  - Reviewers who do not need to be in the discussion phase may leave or stay a learning exercise. The outcome of the discussion is documented for future reference.
- **Decision phase :**
  - At the end of the meeting, a decision on the document under review has to be made by the participants, based on the exit criteria.

- The most important exit criteria is the average number of critical and /or major defects found per page. If the number of defects found per page exceeds a certain level, the document must be reviewed again, after it has been reworked.
- If the document complies with the exit criteria, the document will be checked during follow-up by the moderator. Subsequently, the document can leave the review process.

5. Rework :

- Based on the defects detected, the author will improve the document under review step by step.
- Not every defect that is found leads to rework. it is the author's responsibility to judge if a defect has to be fixed. If nothing is done about an issue for a certain reason, it should be reported to at least indicate that the author has considered the issue.
- Changes that are made to the document should be easy to identify during follow-up. Therefore the author has to indicate where changes are made.

6. Follow-up :

- The moderator is responsible for ensuring that satisfactory actions have been taken on all defects, process improvements suggestions.
- Although the moderator checks to make sure that the author has taken action on all known defects, it is not necessary for the moderator to check all the corrections in detail.
- If it is decided that all the participants will check the updated document, the moderator takes care of the distribution and collects the feedback.
- Recurring defect types point to deficiencies in the software development process. Such defect types should be included in the checklist.

Q.3(b) Explain success factors of review process. [5]

(A) Success factors of reviews :

Implementing (formal) reviews is not easy as there is no one way to success and there are numerous ways to fail.

Find a 'champion'

- A champion is needed, one who will lead the process on a project or organizational level. He needs expertise, enthusiasm and a practical mindset in order to guide moderators and participants.
- The authority of this champion should be clear to the entire organization.
- Management support is also essential for success.
- He should, amongst other things, incorporate adequate time for review activities in project schedules.

Pick things that really count

- Select the documents for review that are most important in a project. Reviewing highly critical, upstream documents like requirements and architecture will most certainly show the benefits of the review process to the project.
- These invested review hours will have a clear and high return on investment.
- In addition make sure each review has a clear objective and the correct type of review is selected that matches the defined objective.
- Don't try and review everything by inspection; fit the review to the risk associated with the document.
- Some documents may only warrant an informal review and others will repay using inspection.
- Of course it is also of utmost importance that the right people are involved.

Explicitly plan and track review activities

- To ensure that reviews become part of the day-to-day activities, the hours to be spent should be made visible within each project plan.

- The engineers involved are prompted to schedule time for preparation and, very importantly, rework.
- Tracking these hours will improve planning of the next review.
- As stated earlier, management plays an important part in planning of review activities.

Train participants

- It is important that training is provided in review techniques, especially the more formal techniques, such as inspection.
- Otherwise the process is likely to be impeded by those who don't understand the process and the reasoning behind it.
- Special training should be provided to the moderators to prepare them for their critical role in the review process.

Manage people issues

- Reviews are about evaluating someone's document.
- Some reviews tend to get too personal when they are not well managed by the moderator.
- People issues and psychological aspects should be dealt with by the moderator and should be part of the review training, thus making the review a positive experience for the author.
- During the review, defects should be welcomed and expressed objectively.

Follow the rules but keep it simple

- Follow all the formal rules until you know why and how to modify them, but make the process only as formal as the project culture or maturity level allows.
- Do not become too theoretical or too detailed.
- Checklists and roles are recommended to increase the effectiveness of defect identification.

Continuously improve process and tools

- Continuous improvement of process and supporting tools (e.g. checklists), based upon the ideas of participants, ensures the motivation of the engineers involved.
- Motivation is the key to a successful change process.
- There should also be an emphasis, in addition to defect finding, on learning and process improvement.

Report results

- Report quantified results and benefits to all those involved as soon as possible, and discuss the consequences of defects if they had not been found this early.
- Costs should of course be tracked, but benefits, especially when problems don't occur in the future, should be made visible by quantifying the benefits as well as the costs.

Just do it!

- The process is simple but not easy.
- Each step of the process is clear, but experience is needed to execute them correctly.
- So, try to get experienced people to observe and help where possible. But most importantly, start doing reviews and start learning from every review.

Q.3(c) Explain the goals of technical review. [5]

(A) The goals of a technical reviews are as follows.

1) Access the value of technical concepts and alternatives in the product and project environment.
2) Establish consistency in the use and separately of technical concepts.
3) Ensure at an early stage, that technical concepts are used correctly.

4) Inform participants of technical content of document.
5) Technical review is a peer group discussion activity that focuses on achieving consensus on technical approach to be taken.

Q.3(d) Define exit criteria in software testing. What is its purpose? Write few examples of exit criteria. **[5]**

(A) Exit criteria : For formal reviews the moderator & always performs an entry check & defines the formal exit criteria.

Once the document passes the entry criteria check the moderator & author decide which part of the document to review. The human mind can comprehend a limited set of pages at one time, the number should not be too high. The maximum number of pages depends on the objective, review type & document type & should be derived from practical experiences within the organization. For a review, the maximum size is usually between 10 & 20 pages.
In formal inspection, only a page or two may be looked at in depth in order to find the most serious defects that are not obvious.
After the document size has been set & the pages to be checked have been selected, the moderator determines the composition of the review team.

Exit criteria is defined as "The specific conditions or on-going activities that must be present before a life cycle phase can be considered complete. The life cycle specifies which exit criteria are required at each phase."
e.g.
I) Exit criteria for unit testing
1) Unit test has passed
2) All priority bugs have been fixed L closed.

II) Exit criteria for integration testing.
1) Internal documentation has been updated.
2) The integration engineer has tested for install ability.

Q.4 Attempt the following (any TWO) **[10]**

Q.4(a) Explain Equivalence Class Partioning and BVA (Boundary Value Analysis). **[5]**

(A) Equivalence Class Partioning

(i) It is very difficult to test the entire software as a whole because software is a collection of set of programs.
(ii) In equivalence partionining software tester divides the entire software in to set of equivalence classes, and each and every equivalence program is tested by using a set of valid as well as invalid input condition.
(iii) The domain of possible input data for each input data element is divided into equivalence class (Equivalence Class Partioning).
(iv) An equivalence class is a group of data values where tester assumes that the test object processes them in same way.
The test of one representative of equivalence class is seen as sufficient because it assumes that for any other input values of same equivalence class the test object will not show different results.
(v) Besides equivalence classes for correct input, those for incorrect input values must be tested as well.
(vi) To test equivalence classes we have 4 types of equivalence class :
(a) If continuous numerical domain is specified then create one valid and two invalid equivalence class.
Input Range : $10 \leq x \leq 20$
Valid $\Rightarrow$ $x = \{10, 11, 12, \ldots\ldots, 20\}$
Invalid $\Rightarrow$ $x < 10$
$x > 20$

(b) If number of values should be entered then create one valid (all possible correct values) and two invalid (less and more than the correct number) equivalence classes are defined.

Input Range : Specific value x = 5

Valid $\Rightarrow$ $x \in 5$

Invalid $\Rightarrow$ $x < 5$

$x > 5$

(c) If a set of values are specified where ach value may possibly be treated differently then create one valid equivalence class for each value of the set (containing exactly these values) and one addition invalid equivalence class (containing all possible other values) are defined.

Input : Member of set

$x = \{a, e, i, o, u\}$

Valid $\Rightarrow$ $x \in \{a, e, i, o, u\}$

Invalid $\Rightarrow$ $x \notin \{a, e, i, o, u\}$

(d) If there is a condition that must be fulfilled then create one valid and one invalid equivalence class to test the condition fulfilled or not.

Input : Boolean value

Valid : True

Invalid : False

in a = 3, b = 2, c = 5

if (a + b = = c && b < c)

(vii) Test completion criteria for equivalence class portioning can be defined as equivalence class coverage.

$$\text{EC coverage} = \frac{\text{Number of Tested EC}}{\text{Total number of EC}} \times 100$$

BVA (Boundary Value Analysis)

(i) It delivers a very reasonable addition to the test cases that have been identified by equivalence classes.

(ii) Faults are often appears at the boundaries of equivalence classes. In BVA software tester emphasizes on boundary condition because maximum number of errors are generally occur at the boundary rather than within the operational bound.
Tester assumes that if software works properly on the boundary then it will definitely work within the boundary condition. The most of the errors in software lies on boundary because boundaries are often not defined clearly or programmers misunderstand them.

(iii) It test with boundary value usually discovers the failure. On the boundary condition because BVA checks the borders of an equivalence class and on every border exact boundary value and both nearest adjacent values (Inside or Outside equivalence class) are tested.

(iv) If the input condition for program is the range within the closed interval i.e. [a, b] then using BVA we have 6 possible test cases as
a–1, a+b, b–1, b+1. Where valid $\Rightarrow a, a + 1, b - 1, b$
invalid $\Rightarrow a - b, b + 1$

(v) An input file has restricted number of data records between 1 to 100, the test value should be 1, 2, 99, 100.

(vi) If permitted number of output values is to be tested, then proceed same as the number of input values. Say if output of 1 to 4 data values are allowed than test cases are 0, 1, 4, 5.

(vii) Test completion criteria for BVA :

$$\text{Boundary Value Coverage} = \frac{\text{Number of tested value boundary}}{\text{Total number of boundary value}} \times 100$$

Q.4(b) Compare Black Box Testing and White Box Testing. **[5]**

(A)

	Black box testing	White box testing
(i)	Here internal structure design/implementation of the item being tested is NOT known to the tester.	Here internal structure/design/implementation of the item being tested is known to the tester.
(ii)	Mainly applicable to higher levels of testing : Acceptance testing, system testing.	Mainly applicable to lower levels of testing : Unit testing, Integration testing.
(iii)	Generally independent software testers.	Generally software developers.
(iv)	Knowledge of programming is not necessary.	Knowledge of programming is required.
(v)	Focus on functionality of the system.	Focus on structure i.e. program/code of the system.
(vi)	Mostly done by testers.	Mostly done by developers.
(vii)	Also known as specification based testing.	Also known as structured based testing.

Q.4(c) Explain state transition based testing. **[5]**

(A) State transition diagram

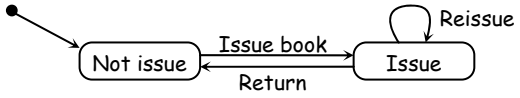
(i) State –

(ii) Start state – ●

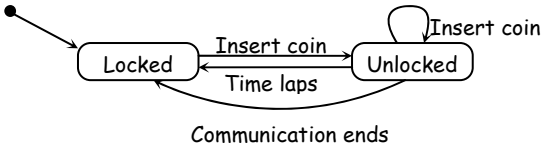
(iii) End state – ◉

(iv) Event / Action – event / action →

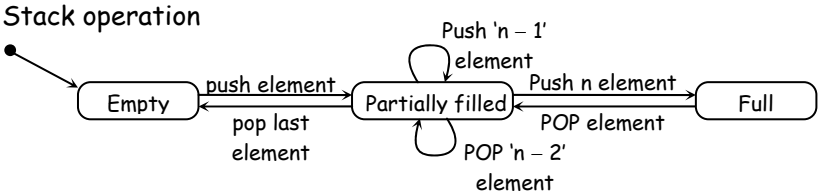
- Book in a library



- Turn style machine



- Stack operation



(a) In many cases not only the current input but also the history of execution or events or inputs influences the output and how the test object will behave to illustrate the depends on history, state transition diagram is sued.

(b) State transition diagrams are the basis of designing test cases during state transition technique. The system or test object start from initial state and can then came into different states. Events trigger state transition where an event normally is a function invocation.

(c) State transition diagram can invoke action, resides the initial state the other special state is an end state which is optional for state transition diagram.

(d) Finite state machine or state transition diagram shows behavior of the system. State transition diagram is a diagram which depicts the states that the system or component can assume and shows the event or circumstances that cause or result for a change of state from one to another.

(e) Consider state transition diagram for stack operation, the test cases can be given as pre-condition.
(1) Stack is initialized → Stack is empty.
(2) Input → Push an element into stack (push "Hello").
(3) Expected result → Stack contains "Hello"

Post condition – The state of stack is partially filled.
The test cases for state transition testing are :
(1) Initial state of the test object.
(2) Input to the test object.
(3) Expected outcome or result
(4) Expected final state.

Initial statement	–	Full
Input	–	POP element "Hi"
Expected result	–	Hi is removed from stack
Expected final state	–	Partially filled

Initial statement	–	Empty
Input	–	Push element "Hi"
Expected result	–	Stack contains "Hi"
Expected final state	–	Partially filled

Q.4(d) Explain the testing technique which is used when there is no specification, or if the specification is inadequate or out of date. **[5]**

(A) Test technique used when there is no specification is the experienced base technique :
It is of two types :

(i) Error guessing : It is a testing technique based on guessing. It depends on skill of tester. The tester acquires this skill from his prior experience. Error guessing mainly depends on the skills and understanding of the tester.
It is considered as an effective test case design technique as it covers all possible test cases.

(ii) Exploratory testing : It is hands on approach in which testers are involved in minimum planning & maximum text execution. The planning includes the creation of a test cases, a short declaration of the scope of a short time-boxed test efforts, the objectives & possible approaches to be used.
The test design & test execution activities are performed in parallel typically without formally documenting the test conditions, test cases of test scripts.
Test logging is undertaken as test execution is performed, documenting the key aspects of what is tested any defects found and any thoughts about possible further testing.
It can also serve to complement other formal testings.

Q.5 Attempt the following (any TWO) **[10]**

Q.5(a) What are the skills that are needed for the test leader? Discuss. **[5]**

(A) Test leader is playing very important role in the testing process. The skills that are needed for the test leader with discussion are as follows.
Test leader tend to be involved in the planning, monitoring and control of the testing activities and task on the fundamental test process.
At the outset of the project, test leader leads the activates.

- Leaders in collaboration with the other stakeholders devise the test objectives organizational test policies, test strategies and test plans. They estimate the testing to be done and negotiate with management to acquire the necessary resources.
- They recognize when test automation is appropriate and if it is, they plan the effort, select the tools, and ensure training of the team.

- They may consult with other groups.
  Example:
  Programmers – to help them with their testing.
- They lead, guide, and monitor the analysis design, implementation and execution of the test cases, test resources, test procedure and test suites.
- They ensure the proper configuration management of the test ware produced and traceability of the tests to the test basis.
- As test execution and then they monitor, measure, control and report on the test progress, the product quality and the test product quality, and the test results, adapting the test plan and competing as needed to adjust to evolving conditions. During test execution and as the project winds down they write summary report on test status.

  Sometimes test leader wear different titles such as test manager, or test coordinator. Alternatively, the test leader role may wind up assigned to a project manager, a development manager or a quality assurance manager.

Q.5(b) Explain test strategies of approaches. [5]

(A) Test approach – is the implementation of the test strategy for a specific project. It includes the decisions made based on the (test) project's goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria and test types to be performed.

The choice of **test approaches** or **test strategy** is one of the most powerful factors in the success of the test effort and the accuracy of the test plans and estimates. This factor is under the control of the testers and test leaders.

Test strategies :

- **Analytical:** Analytical test strategy is the requirements-based strategy, where an analysis of the requirements specification forms the basis for planning, estimating and designing tests. Analytical test strategies have in common the use of some formal or informal analytical technique, usually during the requirements and design stages of the project.
- **Model-based :** You can build mathematical models for loading and response for e commerce servers, and test based on that model. If the behavior of the system under test conforms to that predicted by the model, the system is said to be working. Model-based test strategies have in common the creation or selection of some formal or informal model for critical system behaviors, usually during the requirements and design stages of the project.

Q.5(c) Explain incident report life cycle with help of an example. [5]

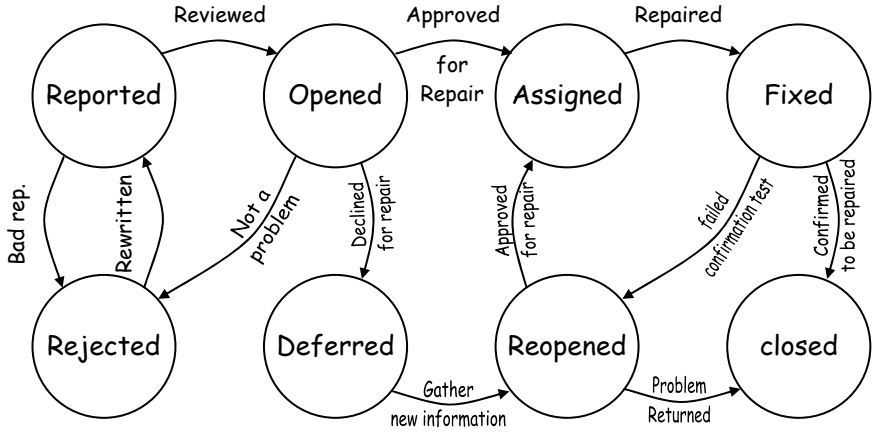
(A) Incident report life cycle :

Incident report describes some situation, behavior or event that occurred during testing that requires further investigation.

Incident report are managed through a life cycle from discovery to resolution. The incident report life cycle is often shown by transition diagram.

All incident reports more through a series of clearly identified states after being reported. Some of these state transitions occur when a member of the project team completes some assigned task related to closing an incident report & so on.

The path taken by incident reports which are ultimately fixed, After an incident is reported, a peer or test manager reviews the report. If successful in the review, the incident report becomes opened. If the defect is to be repaired, a program is assigned to repair it.



Once the programmer believes the repairs are complete the incident report returns to the tester for confirmation testing. If the confirmation test fails, the incident report is re-opened & then re-assigned. Once the tester confirms a good repair, the incident report is closed. No further work remains to be done the owner is responsible for transitioning the incident into an allowed subsequent state. The arrows in the diagram show these allowed transitions.

Q.5(d) Explain risk based testing and project v/s product risk [5]

(A) Risk Based testing

- Testing oriented towards exploring and providing information about product risks.
- Testing efforts can be organized in a way that reduces level of product risk when the system ships.
- Risk based testing uses risk to prioritize and emphasize the appropriate tests during test execution.
- Risk-based testing starts early in the project, identifying risks to system quality and using that knowledge of risk to guide testing planning, specification, preparation and execution.
- Risk based testing also involves measuring how well we are doing at finding and removing defects on critical areas.
- It starts with product risk analysis.
- One technique for risk analysis is reading the requirements specification, design specifications, user documentation and other items.
- Risk analysis is performed by asking , "What should we worry about?".
- When we talk about specific risks, we mean a particular kind of defect or failure what might occur. For ex, if you were testing the calculator utility that is bundled with Microsoft Windows, you might identify 'incorrect calculation' as a specific risk with the category of functionality.

Product Risks (factors relating to what is produced by the work, i.e the thing we are testing)

It is a risk to product quality.

- Product risks is a risk directly related to the test object.
- The delivered product has inadequate functional quality or is nonfunctional. The product is not fit for its intended use and is thus unusable.
- Product risk can be thought of as the possibility that the system or software might fail to satisfy some reasonable customer, user expectation.
- Unsatisfactory software might omit some key functions- that the customer specified.
- Unsatisfactory software might fail in ways that cause financial or other damage to a user or the company that user works for.

Project Risks

- A risk related to management and control of the project.

- There are direct risks such as the late delivery of the test items to the test team or availability issues with the test environment.
- There are indirect risks such as excessive delays in repairing defects found in testing or problems with getting professional system admin support for the test environment.
- To discover risks, ask yourself and other project participants , "What could go wrong on the project to delay or invalidate the test plan, the test strategy and test estimate?.
- What are unacceptable outcomes of testing or in testing? What are the likelihoods and impacts of each of these risks?"

Q.6 Attempt the following (any TWO) [10]

Q.6(a) Explain potential benefits of risk of using tools. [5]

(A) There are many benefits that can be gained by using tools to support testing. They include :

- **Reduction of repetitive work**
  Repetitive work is tedious to do manually. People become bored and make mistakes when doing the same task over and over.
- **Greater consistency and repeatability**
  People tend to do the same task in a slightly different way even when they think they are repeating something exactly. A tool will exactly reproduce what it did before, so each time it is run then result is consistent.
- **Objective assessment**
  If a person calculates a value from the software or incident reports, they may inadvertently omit something. Using a tool means that subjective bias is removed and the assessment is more repeatable and consistently calculated.
- **Ease of access to information about tests or testing**
  Having lots of data doesn't mean that information is communicated. Information presented visually is much easier for the human mind to take in and interpret. For ex. a chart or a graph is a better way to show information than a long list of numbers. Special purpose tools give these features directly for the information they process.

Risks of using tools : Risks include

- Unrealistic expectations for the tool;
- Underestimating the time, cost and effort for the initial introduction of a tool;
- Underestimating the time and effort needed to achieve significant and continuing benefits from the tool;
- Underestimating the effort required to maintain the test assets generated by the tool.
- Over-reliance on the tool.

Q.6(b) Explain Dynamic analysis tools used for Performance Monitoring. [5]

(A) Dynamic analysis tools

- Detecting memory leaks;
- Identifying pointer arithmetic errors such as null pointer;
- Identifying time dependencies.

Performance-testing, load-testing and stress-testing tools

- Performance-testing tools are concerned with testing at system level to see whether or not the system will stand a high volume of usage.
- A load-test checks that the system can cope with its expected number of transactions.
- A stress testing – is conducted to evaluate a system or component at or beyond the limits of its specified requirements;
- Generating a load on the system to be tested.
- Measuring the timing of specific transactions as the load on the system varies;
- Measuring average response times;
- Producing graphs or charts of responses over time.

Monitoring tools :

Monitoring tools used to continuously keep track of the status of the system in use, in order to have the earlier warning of problems and to improve service.

These are monitoring tools for servers, networks, databases, security etc…

- Identifying problems and sending an alert message to the admin.
- Logging real-time and historical information.
- Finding optimal settings
- Monitoring the number of users on a network.
- Monitoring network traffic.

Q.6(c) Explain requirement management tools. [5]

(A) (i) Requirement management explain characteristics of requirement management tool.

(ii) Requirement management tools are used to gather requirements from customers. Example of requirement management tools are web portals or set of questionnaires.

(iii) The features or characteristics of requirement management tools includes :

- Storing requirement statements.
- Storing information about requirement attributes.
- Checking consistency of requirements.
- Prioritizing requirements for testing purposes.
- Identifying undefined, missing or 'to be defined later' requirements.
- Traceability of requirements to tests and tests to requirement, functions or features.
- Traceability through levels of requirements.
- Interfacing to test management tools.
- Coverage of requirements by a set of tests.

Q.6(d) Describe the features of Test design of test execution tools. [5]

(A) Features of test execution tools include support for :

- Capturing (recording) test inputs while tests are executed manually.
- Storing an expected result in the form of a screen or object to compare to, the next time the test is run.
- Executing tests from stored scripts and optionally data files accessed by the script (if data-driven or keyword-driven scripting is used).
- Dynamic comparison (while the test is running) of screens, elements, links, controls, objects and values.
- Ability to initiate post-execution comparison.
- Logging results of tests run 9pass/fail, differences between expected and actual results).
- Masking or filtering of subsets of actual and expected results, for example excluding the screen-displayed current date and time which is not of interest to a particular test.
- Measuring timings for tests.
- Synchronizing inputs with the application under test, e.g. wait until the application is ready to accept the next input, or insert a fixed delay to represent human interaction speed.
- Sending summary results to a test management tool.

Q.7 Attempt the following (any THREE) [15]

Q.7(a) Explain the term "testing is not debugging". [5]

(A) (i) Testing and debugging are two different activities because difference between testing and debugging are :

- Debugging only conducted during coding phase of the software development whereas testing can be performed in each phase of software development according to v-model.

- Debugging emphasis on syntactical errors whereas software testing emphasis on syntactical as well as logical error.
- Debugging is compiler driven or executed activity whereas testing is performed by professional software tester.
- To perform debugging activity no extra set of inputs are required whereas a set of valid and invalid input conditions called as test cases required for software testing.

(ii) To be able to correct defects or bugs it must be localized in the software. Initially we only know that the effect of defect but not the precise location of bug in the software.

(iii) The localization and correction of defect are the job of software developer and are called debugging. Repairing of defects generally increases the quality of software.

(iv) Debugging is the task of localization and correcting faults whereas goal of resting is detection of failure.

(v) Testing software has following purposes:

- Executing programs in order to find failures.
- Executing programs in order to measuring quality.
- Executing program in order to provide confidence.
- Analyzing a program or its documentation in order to prevent defects.

(vi) The whole process of systematically executing programs to demonstrate correct implementation of requirements to increase confidence and to detect failure is called testing.

Q.7(b) Explain Alpha & beta testing with reference to acceptance testing. [5]

(A) Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions

The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

Beta Testing

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.

Q.7(c) Explain roles of responsibilities of moderator, reviewer of author in review process. [5]

(A) The Following are the members and there responsibilities in the review process.

(1) Moderator

(a) He is a chairperson of the entire review process.

(b) Moderator determines inco–operation with author, the type of review, approach and composition of review team.

(c) The moderator is responsible for scheduling of review process ie. date, time, venue, agenda of review process will be determined by review.
(d) Moderator prepare policy, plan and objectives of review process.
(e) The training of reviewer can be conducted by moderator.
(f) Moderator performs entry check and exit criteria.
(g) Moderator can keep a track of follow–up during rework in order to control the quality of inputs and outputs of the review process.

(2) Author

(a) Author is a writer of document under review.
(b) The author's goals should be to learn as much as possible with regards to improving the quality of the document, but also to improve his or her ability to write future documents.
(c) The author's task is to illuminate unclear areas and to understand the defects found.
(d) Author must be present physically during the review process of his own document.
(e) If Author document exceeds exist criteria then author must be involved in the rework phase.

(3) Reviewer

(a) The task of the reviewer is to check authors document for defects.
(b) The levels of domain knowledge or technical expertise needed by the reviewer will depends on the type of review and type of document which is under review.
(d) The Reviewers should be chosen to represent different perspectives and roles in the review process. In addition to the document under review, the material reviewer receives includes source documents, standards, checklists,
(c) The soul responsibility of reviewer is to review author's document and try to find as many defects as present in the document.

Q.7(d) What are the purposes of Test Monitoring? [5]

(A) **Test monitoring** – a test management (the planning, estimating, monitoring and control of test activities carried out by a test manager) task that deals with the activities related to periodically checking the status of a test project. Reports are prepared that compare the actuals to that which was planned.

The purpose of test monitoring is as follows :

- Give the test team and the test manager feedback on how the testing work is going, allowing opportunities to guide and improve the testing and the project.
- Provide the project team with visibility about the test results.
- Measure the status of testing, test coverage and test items against the exit criteria to determine whether the test work is done.
- Gather data for use in estimating future test efforts.

Q.7(e) Postal rates of 'light letters' are 25p up to 10g, 35p up to 50g plus an extra 10p for each additional 25g up to 100g. Design test case for weight of the letters using equivalence partitioning. [5]

(A) Postal rates

light letters 25p – 10 g
35p – 50 g
\+ 10 + 25 g
⋮
100 g

0–10 gm	11 gm – 50 gm	51 gm – 75 gm	76 gm – 100 gm
25	35	45	55

Test Cases

	Input gm	Remark	Output
1)	20	Valid	Charge 35 p
2)	8	valid	Charge 25 p
3)	56	Valid	Charge 45 p
4)	80	Valid	Charge 55 p
5)	110	Invalid	enter valid data

Q.7(f) Define test scripts. Explain briefly advanced scripting techniques for test execution tools. [5]

(A) Test Scripts

Test script are written in scripting languages. Scripting languages are programming language.

Test execution tools use a scripting language. The advantage is that tests can repeat actions for different data values. They can take different router depending on the outcome of a test and they can be called from other scripts giving some structure to the set of tests.

Characteristics are :

1) Capturing test inputs while tests are executed manually.
2) Storing an expected result in the form of a screen or object to compare to, the next time the test in run.
3) Executing tests from stored scripts and optionally data files accessed by the script.
4) Dynamic comparison.
5) Ability to initialize post–execution comparison.
6) Logging results of test run.
7) Making or filtering of subsets of actual & expected results.
8) Measuring timings for tests.
9) Synchronizing inputs with the application under test.
10) Sending summary results.

❑ ❑ ❑ ❑ ❑