Unit I : Basics of Software Testing and Testing methods.
(Weightage - 14 marks)

Q1 Define Software Testing. (2M)

⇒ Software Testing is defined as performing validation and verification of software.

In simpler terms, it's the process of running program to find errors.

Software Testing is process of executin a program with the intent of finding errors.

Testing is process of uncovering Errors as possible.

Testing can show the presence of Bugs & is a support function that helps developers look good by finding their mistakes before anyone else.

Q2 Roles / objectives of Software Testing. (2M)

⇒ 1) Finding defects which may get created by programmer while developing and the software.
2) To prevent Defects.
3) Gathering Confidence and providing information about level of quality.
4) To make sure that the end results meet the bussiness and user requirements.
5) To gain confidence of customers by providing them quality product.
6) Testing should be Conduct with some goal or reason.

7) To ensures that it satifies the BRS (Business and Requirement Specification) and SRS (System Requirement Specification)
8) Testing should be applied all through software Lifecycle.

Q3 Define Following:
i) Error ii) Bug iii) Fault iv) Defect v) Failure.

⇒ Error: A human action that produces an incorrect result
An error is an mistake, misconception or misunderstanding on part of software developer.
In category of developer we include software engineers, programmers, analyst.

Bug: A Bug is result of coding error.
An error found in development environment before product is shipped to customer

Fault: The state of software caused by error or bug. An incorrect step, process or data definition in computer system.

Failure: Deviation of the software from it's expected result is known as Failure. It is event.

Defect: A defect is an error or bug in application which are created.
A programmer/Developer while designing and building the software can make mistakes and error.
This mistakes or errors indicate that there are flaws/faults in software. these are defects.

Q4 Define Verification and validation. (2m)

→ Verification

Verification is the process of checking the software achieves its goal without any bugs.
Verification is static Testing.
In Simpler words, Verification checks the software meets the requirements set at start of development phase.
It involves check document, design, code and program without running.
Are we building the product right?

validation

validation is process of checking whether the software product is up to mark or other words product has high level requirements.
validation is Dynamic Testing.
It is process of checking the validation of product. i.e
Are you building right product?

Q5 Compare verification and validation (5m)

⇒

parameters	verification	validation
Define	Verification is the process of checking that software achieves its goal without any bugs.	Validation is the process of checking whether s/w product is up to mark.
Type	Verification is Static Testing	Validation is Dynamic Testing.
Execution	It does not include execution of code.	It includes execution of code.
Timing	It comes before validation.	It comes after verification.
Error focus	Verification is prevention of errors	Validation is detection of errors.
Purpose	It checks the s/w conforms specification or not	It check s/w meet customer expection or not.
Focus	It includes Checking documents/design, codes and program.	It includes testing and validating the actual product
Methods	Static testing/walk through/Inspection/ Review	Dynamic testing, End users, White & Black Box Unit & Integration
Testing		

Q6 Define Static and Dynamic Testing. (2M).

⇒ Static Testing

In static testing, testing and identification of defects is carried out without executing the code.

This testing is done in verification process.

This testing consist of static analysis and reviewing of documents.

For example, (Reviewing, walkthrough, inspection etc...)

Dynamic Testing

In this testing the software code is executed for showing the result of tests performed.

Dynamic testing is done in validation process. i.e. unit testing, integration testing, system testing etc...

Q7 Compare Static Testing & Dynamic Testing. (2M)

Static Testing	Dynamic Testing
Static testing is completed without executing program	Dynamic testing is completed with execution of program
This testing is executed in verification stage.	This testing is executed in validation stage.
Executed before compilation.	After compilation
prevent defects	find & fix the defects
Cost is less	high cost.

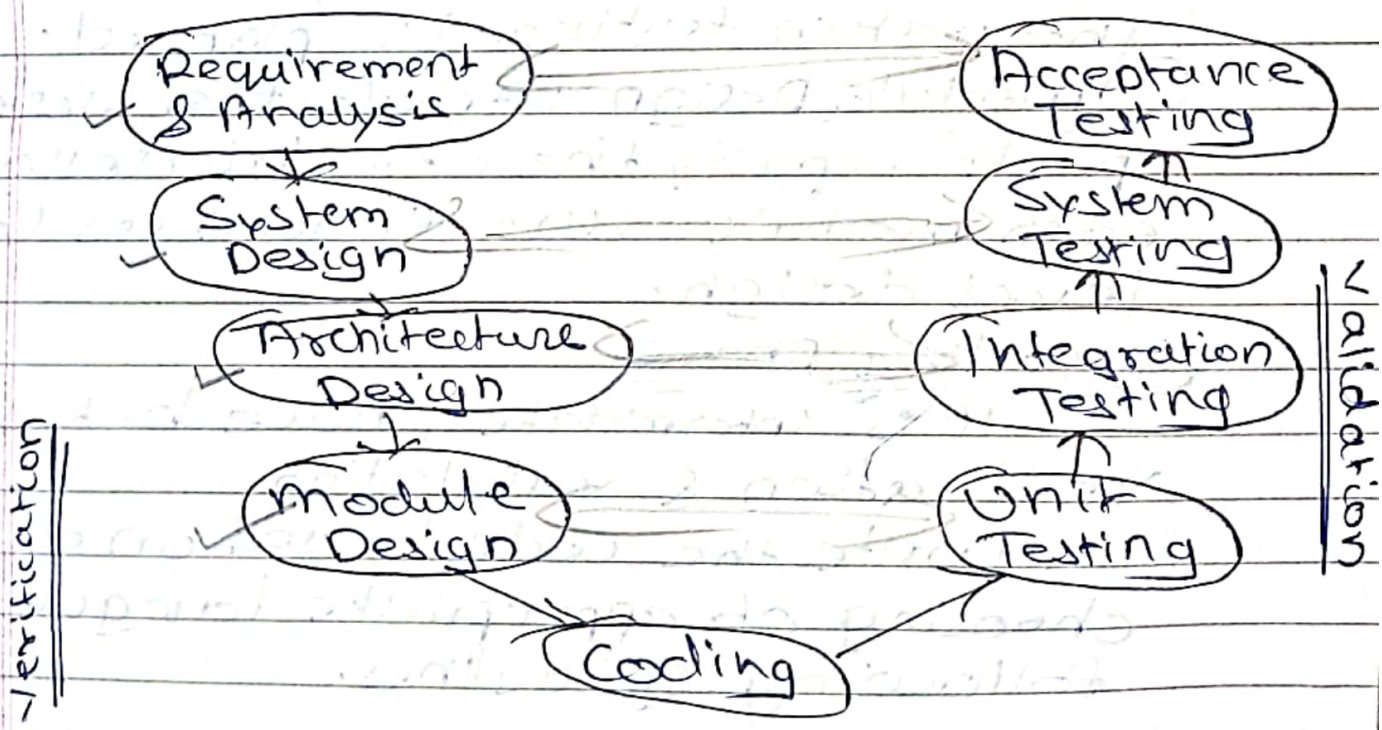
Q8 Define or compare Software Quality Assurance and Software Quality Control. (5 M)

	Software Quality Assurance	Software Quality Control
Definition	Quality Assurance is known as QA and focuses on preventing defects.	Quality control is known as QC and focuses on identifying defect.
process	Quality Assurance is preventive process. (prevention).	Quality control is reactive process
method	Quality Assurance that the approaches, techniques methods and processes are designed for projects are implement correctly	Quality control that approaches, techniques, methods and processes are following properly
monitor & verify	QA monitor and verify that processes used to manage and create the deliverable have been followed and are operative.	QC activities monitor and verify that project delivers meet the defined quality standards.
Timing	Complete before QC	After the QA

V. Imp

Q9 V-Model or V-Shaped Model in Detailed. (6 marks).

⇒ - V-Model is also known as Verification & validation model.
- Extension of waterfall model.
- Testing is Associated with every phase of life cycle.)
- Verification phase (Requirement Analysis, System Design, Architecture Design, Module Design).
- Validation phase (Unit Testing, Integration Testing, Acceptance testing).



Verification Phases

① **Requirement Analysis**: Understand what the customer needs and expects. Communicate with customer to gathered detailed documents.

② **System Design**: Analyze and interpret the overall system based on user requirements. System engineers study the user requirements documents to design system

③ **Architecture Design**: Choosing and design system architecture, create list of modules, their functions, relationship, dependencies, databases, tables etc... Integration testing is planned.

④ **Module Design**: Break the system into parts. specify the detailed design for each module, also known as low-level design.

⑤ **Coding phase**

It is intermidate phase between verification & validation.

write the code based on design, choosing of apporpriate language following guidelines.

Validation phases

① Unit Testing: Testing individual parts of code.

② Integration Testing: Test how different parts work together of syste

(3) System Testing: Complete system test.
(4) Acceptance testing: Test system in real use. environment.

Q9 Advantages
Time saving, Good understanding for begining, Every component is testable.

Q10 When to start and stop Testing of software (Entry and Exit criteria) (4m)
⇒ When to Start Testing? (Entry criteria).
It depends on the process and the associated stakeholders of project.
Before entrying in the phase of STLC. the project should meet some conditions are known as criteria.
Some conditions are
1) Documentation should be ready.
2) Test cases should be maintained.
3) Requirements should be done at current phase & analysis.
Includes: Software Tester, Developer, manager, End user.

When to Stop Testing? (Exit criteria)
The criteria where testing should meet current STLC phase, criteria and while going to next phase.
It is difficult to determine when to stop testing as testing is the

never ending process and no one can claim the software is 100% tested.

following are some conditions :-
- Testing Deadline
- Completion of test case execution.
- management decision.
- completion of function and code coverage to a certain point.

Q) List types of white-Box testing, Describe any two types of white-box testing with neat diagram. (4m).

⇒ Types of white-Box testing
1) Inspections
2) walk throughs
3) Technical Reviews
4) Functional Testing
5) Code Coverage Testing.
6) Code Complexity Testing.

Q Describe Code Complexity testing (4m)

⇒ Code Complexity testing, also known as Cyclomatic Complexity testing.
It is used to measure how complex the program is.
It helps to identify parts of the code that might be more prone to error or harder to maintain.

Input C for Ab + c (a+b)

Cyclomatic complexity is calculated using number of decision points in a program.

Higher complexity means more paths through code which leads to defects.

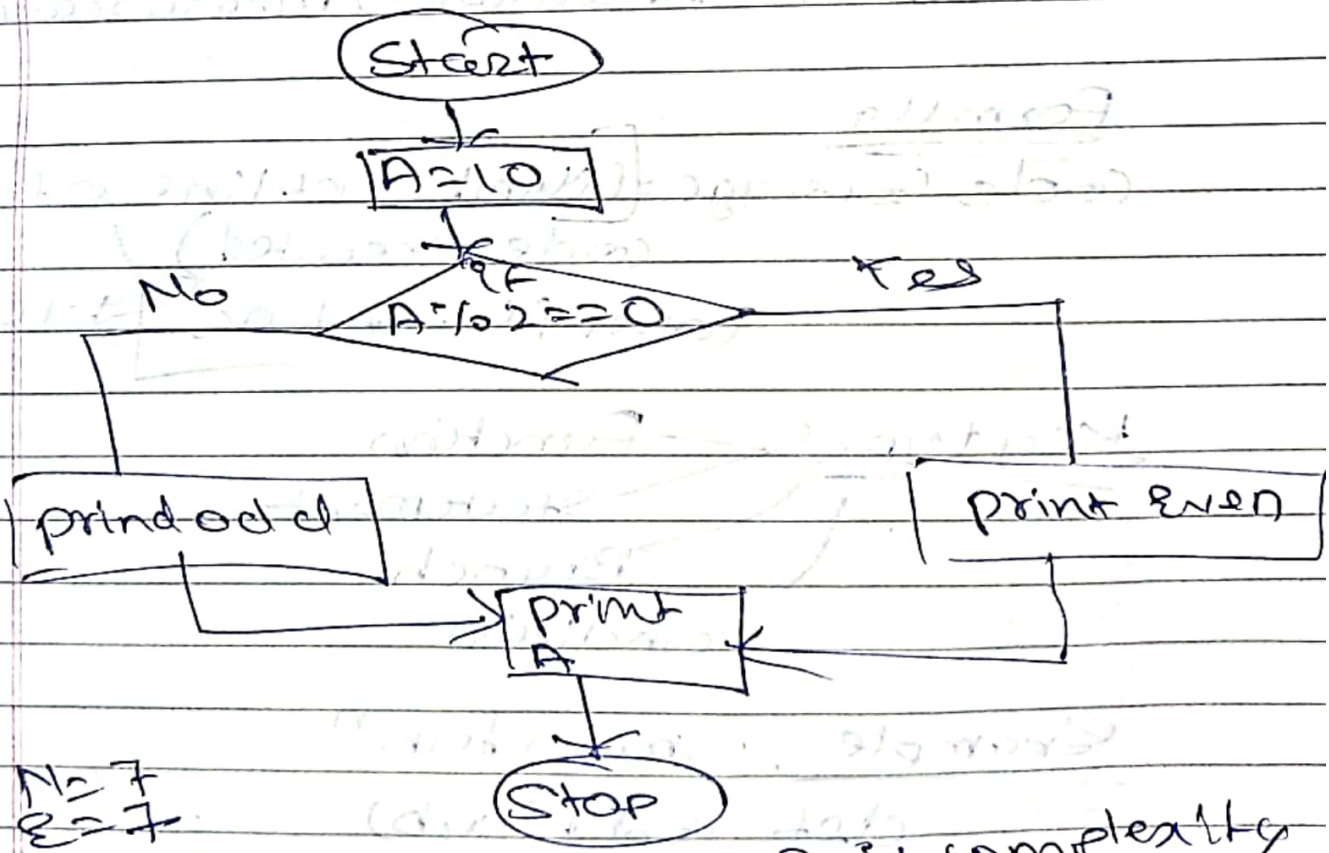
Complexity formula

$M = E - N + 2$

M = Cyclomatic complexity
E = no. of edges
N = no. of Nodes

Or

$M = P + 1$ (P is predicate nodes).



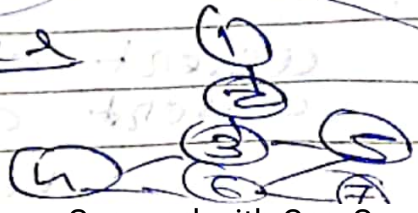
N = 7
E = 7

$M = E - N + 2$
$= 7 - 7 + 2$

$M = 2$

∴ 2 is complexity

Red



2) Code Coverage Testing

Code Coverage Testing is determining how much code is being tested.

The main purpose of code coverage testing is to check if your tests are covering all parts of code.

It helps in identifying risk and reduce % of untested areas to find.

Working

Write tests → Run test → Measure coverage

Formula

$$\text{Code Coverage} = \left(\frac{\text{Number of line of code executed}}{\text{Total No. of LOC}}\right) * 100$$

Key types ← Function
- Statement
- Branch
- Condition

Example : Small fun^n

```
def add(a,b)
    return a+b
```

```
assert add(1,2) == 3
assert add(-1,-1) == -2
```

100% coverage

3) Functional Testing

Functional testing in white box testing focuses on verifying specific functions within code work as expected, but with knowledge of structure of code.

The purpose is to test individual functions or components while being aware of how code is written.

Unlike Black box testing, where internal structure is known, White-Box functional testing allows testers to create tests based to actual code.

Works

Understand code → write test → Run test → Evaluate Result

Example

add() & sub()
are check individually.

Q12 Describe Boundary value analysis with suitable example. (4m)

⇒ Boundary value analysis is a testing method that focuses on checking "edges" or "boundaries" of input ranges.
The idea is that errors are most likely to happen at edges than middle.

Working

Inputs are divided into groups (call partitions) where each group should behave similarly.
Instead of testing just any value within group. BVA tests the value right at the edges of group.
Both minimum and maximum (smallest & largest value).

Types of Boundaries
- Valid (start, end, min & max allowed)
- Invalid (one step beyond above & below)

Example

Input accepted 1 to 100

↳ Valid Boundary : 1 & 100

↳ Invalid Boundary : 0 & 101

Expected

Q13 Describle EQUIVALENCE Partitioning with Example.

⇒ Equivalence partitioning is a software testing technique that divides the input data of software application into different group or "partitions".

The idea is that all inputs within partitions should behave same way, so you only need to test one input from each group, rather than every possible input.

Working

The input data is divided into partition where every value of partition should be treated the same by software.

Instead of testing all the values you pick up represent the value from each partition of test.

Types
- Valid (Accepts)
- Invalid (reject or handle)

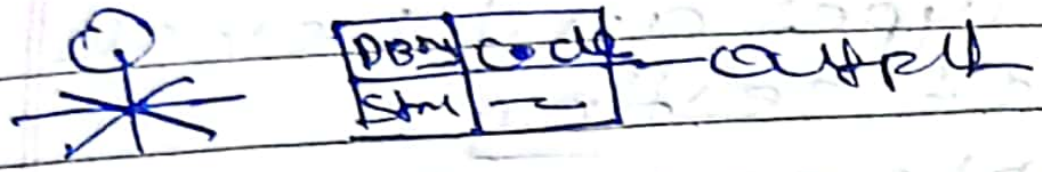
Example

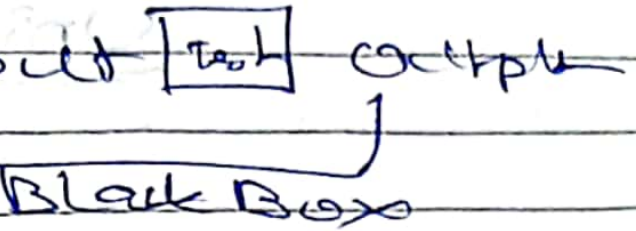
Input age = 18 - 60

- Valid ⇒ Any age from 18 to 60 (25, 30, 35)
- Invalid
  - Age < 18
  - Age > 60
  - Nonnumeric abc

Q14 Compare Black-Box & White-Box Testing. (4M).

Parameters	White-Box	Black-Box
Definition	Testing in which internal structure need to be known is white Box testing.	Testing in which no internal structure needed known as Black-Box testing
Name	Also known as Clear Box testing	Also known as closed-Box testing
performed by	mostly done by Developers	mostly done by testers
Knowledge	Knowledge of implementation is mandatory.	No knowledge of implementation is required
Time	more time consuming	Less time consuming
Known as	Also known as Structural testing	Also known as Functional Testing
programming knowledge	required	Not required.
level applicable	higher-levels	lower levels.
Algorithm testing	Suited	Not suited.





End

Pooja Sure.