

C1: Software Testing Concepts

1. Recall the principles of software testing.

The principles of software testing include:

1. Testing shows the presence of defects, not their absence.
 - a. Identify the defects, not the absence of defects. (We can't prove that the product has no defects or it is perfect)
 - b. To fix the defects found before releasing the products to clients.
2. Exhaustive testing is impossible.
 - a. It is impossible to test out all inputs and their combinations.
 - b. Use an efficient way to test out what we want to test using a short time.
 - c. Example: Test quantity text field with range 1-1000. By using the right technique, we can just use 3 data to test out (invalid data less than 1, valid data within 1-1000, invalid data more than 1000).
3. Early testing.
 - a. Testing activities should start as early as possible in the software life cycle.
 - b. Example: Review the user requirement and system requirement in documentation.
 - c. Type of testing: static testing (before development), dynamic testing (done on development code)
4. Defect clustering.
 - a. We cannot predict the happening of defects.
 - b. But one defect may cause another subsequent defects to happen. So, we need to increase more testing on the other relevant parts.
5. The pesticide paradox.
 - a. Modify or change the test case to explore new defects when the test case is no longer find out the desired result
 - b. repeatedly executing the same test case will not be able to find out new result or defects. It may cause wasting of resources.
6. Testing is context-dependent.
 - a. Different testing ways need to be done in different context (mobile app, web) even though they are the same company or name.
 - b. Different contexts may have different environments and different defects.
7. Absence-of-errors fallacy.
 - a. Even though the system does not have defects found or no failure found, the system will still not be a useful system if it did not fulfill users' requirements.
 - b. It is a useful system only if the users consider that the system is useful and it has fulfilled their requirements.

(pg20-27)

2. Software testing is unavoidable when developing the software. Some said software testing is an "umbrella activity". Analyse the reason behind.

Software testing is considered an "umbrella activity" because it is integrated into every phase of the software development life cycle. It starts from the requirement analysis phase and continues through design, coding, and implementation, ensuring that each phase meets its objectives and that the final product is of high quality. Testing activities are not isolated but are continuous and pervasive throughout the development process.

(pg9-10, 28-30)

- Software testing is a must in the whole development of the software.
- In each stage in software development, the software testing will be executed as an umbrella activity. Any outcome from each stage, they need to be tested.
- To ensure all the tasks are done in the right way, prevent failures.

3. Early testing is one of the principles of software testing, throughout the software development life cycle, identify what artifacts will be tested after the Analysis, Design, Coding and Implementation stage.

After the Analysis stage: Requirements and specifications.

After the Design stage: Design documents, architecture, and models.

After the Coding stage: Source code, unit tests, and integration tests.

After the Implementation stage: System tests, acceptance tests, and deployment tests.

(pg22-23, 28-30)

- Artifacts are the milestones for each stage (purpose).
- Analysis stage
 - Artifact: Requirement Document / Software Requirement Specification (SRS)
 - Review whether all the requirements are there and they are correct.
- Design stage
 - Artifact: Software Design Document / Software Design Architecture Document (database design, UI design, prototype are parts of them)
 - Test whether all the requirements are converted into a design document correctly.
- Coding stage
 - Artifact: Source code
 - Test whether the function is working
 - Test whether the modules are working when all the codes are combined together (compiler)
- Implementation stage
 - Artifact: Complete developed system
 - Test the performance, usability, efficiency of the system and so on
 - Test functional or non-functional requirements

4. Why is testing important?

Testing is important because it helps identify and correct defects, ensures that the software meets the requirements, verifies that the system is fit for purpose, and provides a quality product that satisfies user needs. It also helps prevent failures that could lead to significant financial loss or harm.

(pg9-10)

- Analyzing a program or its documentation to prevent failures
 - Test the earlier stage either the documentation or program which are not yet a system to prevent failures.
- To check if the system meets the requirements and be executed successfully in the intended environment.
 - To ensure that the system is working fine in the actual environment, not only on your own computer.
- To check if the system is "Fit for purpose".

- To ensure that the system is functional and works for any objective mentioned by the clients.
- Stable
- To check if the system does what it is expected to do.
- To identify and correct defects
 - This is the main purpose of testing as we can't promise that the system is perfect.
 - To identify the hidden defects and fix them.
- To check whether the users need are satisfied.
- To avoid user detecting.
 - Don't let the end user detect the defects.
- Provide quality product
 - The more the testing has been done, the more defects you found, the more defects you fixed, the higher the quality of the final product is.

5. Different systems cannot have the same way of testing? Do you agree? Justify your points.

Yes, different systems require different testing approaches because testing is context-dependent. The type of system, its usage environment, and the associated risks dictate the testing strategy. For example, safety-critical systems like medical devices require rigorous testing compared to e-commerce applications, which may focus more on usability and performance.

(pg25-26)

- Yes, I agree.
- As one of the principles of testing said: "Testing is context dependent", each of the systems has unique context, requirements and purposes.
- When conducting the testing, we must follow this principle.
- For example, e-commerce systems and online banking systems have different domains. / Different versions of the same system / Different platforms used for the system (mobile application, web application).

6. Explain the relationship between **Error, Defect and Failure**.

An error is a human mistake that leads to incorrect results. A defect (or fault) is an issue in the software code caused by an error. A failure occurs when the software does not perform as expected due to a defect.

(pg13-16)

- Error / Software Error
 - Can be done by humans (developers, designers, etc), humans may make mistakes.
 - When developing the code, humans may make mistakes such as syntax errors.
 - It is an incorrect internal state which is caused by humans.
- Defect / Software Defect
 - Bug, defect or internal error.
 - It is inside the program.
 - The mistakes made by humans will turn into bugs in the program and affect what the system should do.
- Failure / Software Failure
 - The bugs in the program will turn into a software failure.
 - Failure indicates that the system cannot fulfill the user requirements.

Not all software errors will turn into software faults. And, not all software faults will turn into software failure. It means not all human errors will turn into software defects and not all software defects will turn into software failure. But we cannot guarantee that which one will not or will be turned into the subsequent and more serious situation because we cannot know what they have done in every part of software development.

7. Why some defects in the software will not turn into failure. Explain with one example.

Some defects do not turn into failures because they may not be executed or the specific conditions required to trigger the defect do not occur. For example, a defect in a rarely used feature might never cause a failure if that feature is never used.

(pg17)

- The defects are not executed yet or not happening yet so it would not turn into failure but it still might turn into failure in the future. (hiding effect)
- The defects might have a low severity effect on the system. For example, spelling mistakes, incorrect alignment or image display problems. They are observable by the users but it would not stop them from using the system.

8. When performing testing, some tests have a higher priority than others. What are the criteria to be considered in prioritizing the tests?

The criteria for prioritizing tests include:

Risk: Tests that cover high-risk areas of the application should be prioritized.

Impact: Tests that cover critical functionalities that have a significant impact on the user or business should be prioritized.

Frequency of Use: Features that are used more frequently should be tested first.

Regulatory Requirements: Tests that ensure compliance with regulatory standards should be prioritized.

Dependencies: Tests that other tests depend on should be executed first.

(pg28-30)

- Importance of function: Which function is more important
- Criticality of function: High critical functionalities and modules will be tested first
- Precondition: Consider the precondition test cases or function first, ensure that the precondition test cases have passed before continuing the subsequent functionalities. (e.g. login function is a must before any other functions such as create post function, search function)

C2: Level of Testing

1. Discuss the consideration / preparation should be taken when performing integration test.

- Prepare sufficient test data to see how the integrated system can work with the test data.
- Make sure that the individual module testing has been conducted before integrating them. The integration testing focuses on the system integrity.
- Setup the integration environment (software, hardware, etc).
- Consider the approaches to be used.
- Prepare dummy code if there is a need to integrate a not fully completed individual module for testing purposes.

2. What are the advantages and disadvantages of incremental and non-incremental approaches? Which approach is better? Justify your answer.

Advantages of incremental approaches

- Every module will be tested repeatedly each time integration. Less bugs, higher quality can be caused.
- Meet one module with another module earlier. Can do integration testing earlier to identify the potential defects at earlier stages. Easy to conduct debugging.
- Requires less work to prepare drivers or stubs.

Disadvantages of incremental approaches

- More testing cause more hardware costs.
- Does not support parallel testing.

Advantages of non-incremental approaches

- Support parallel testing on different modules.

Disadvantages of non-incremental approaches

- Each module can only undergo 2 times of testing exposure (1 for individual testing, 1 for integration testing)
- Less machine time causes less hardware costs.
- Cannot identify the defects earlier as integration testing will only be conducted at later stages. Debugging will be difficult to conduct.
- Requires more work to prepare drivers and stubs.

Incremental approach is better because it can effectively explore defects at early stages and ease debugging.

3. Compare Unit Test with Integration Test.

	Unit Test	Integration Test
What to test	To test individual function / operation / component	To test the combined functionalities / operations / components in any level
When to conduct	Before integration	After integration
Purpose	To test independent units / parts to check the functions within them	To see the compatibility of the combined units

Requirement of testing tool	Can test by using compiler / editor tools (more simple)	Require testing tools (more complex)
-----------------------------	---	--------------------------------------

4. In some cases, Integration Test is performed but Unit Test is ignored. Explain the reasons.

- Focus on the result returned by the completed system so the unit test is ignored. The outcome of the unit test is unimportant. (may be used when the unit is too simple to test and can't give an accurate result)
- Dependency of the system. We cannot test the unit as itself without integrating with the other units. (e.g. the system can work only if the admin site and customer site are integrated together, the functions in customer site may only work depending on what the operations have been done in admin site)
- Reuse component. The component has been developed in the previous project and just reused in the current project. Thus, no unit testing will be needed, it just needs integration testing.

5. Differentiate Regression Test with Re-Test.

Regression test

- Executed once there is any changes or modification on the source code.
- To compare with the previous version to ensure that there is no any defects caused after modification done.
- Might modify the test cases and requirements.

Re-test

- Repeat the same testing without any changes made
- To ensure the module is completely correct.
- Not modify any test case and requirements.

6. What is the difference between Driver and Stub?

Stub

- Replacing lower level module
- To be called by the higher level
- Used by bottom-up strategy

Driver

- Replacing higher level module
- To call the lower level module
- Used by top-down strategy

7. Give an example of an artifact that to be tested/reviewed for each stage in a V – Model. Discuss what the quality attributes are (e.g. Correctness, Consistency, Reliability, Security ...etc.) to be tested for the artifacts.

- SRS: based on client request and no conflict (consistency), completeness, correctness
- Design document: design graphs must be consistent with client requirement (consistency)
- Source code: Make sure source code is secure (security), efficient, reliable

C3: Static Testing

1. Describe Static Testing.
 - Conducted before development of software
 - Test and analyze requirements, design document and other documentation without testing any source code

2. List and describe the Review Processes.
 - Planning
 - Decide what document with different version to review
 - Decide what review technique to conduct
 - Form a review team
 - Kick off
 - Secretary will share the information to the team members in meeting.
 - Requirement document will be shared with the reviewers.
 - Individual preparation
 - Study and identify whether there are any defects in the documents.
 - Create comment and questions.
 - Review meeting
 - Review leader or moderator will lead the review meeting.
 - Control the meeting to make sure the meeting can be completed within specified time.
 - Summarize the recommendation on the SRS.
 - Rework
 - Project manager decides whether to accept the final recommendation or suggest something else.
 - Author will rework on the documentation based on the changes needed.
 - Follow up
 - To check whether the rework has been done correctly or not.

3. Describe three advantages of Static Testing over Dynamic Testing?
 - Can easily identify and fix the defects at earlier stages before source code is created.
 - Save time and cost as the defects can be fixed immediately on documentation which wouldn't cause much time and cost.
 - Offer a clear and better understanding of the project so the development time required can be shortened.

4. Who are the participants in a Review and what are their roles?
 - Project manager
 - Accept the final recommendation given or give a new suggestion on how to rework
 - Follow up of the rework
 - Secretary
 - Share information of the meeting to all the members
 - Share document to the reviewers
 - Author
 - Do the rework based on suggestion or recommendation
 - Review leader
 - Lead the review meeting
 - Make the final recommendation
 - Follow up of the reworks
 - Reviewers

- Review the documents
5. Imagine that you are the chairperson of a formal review, if there is conflict happening among the reviewers, how do you solve the conflicts between the reviewers?
- Request all the reviewers to express their understanding and opinion
 - Conduct a voting
 - Ask the reviewers to provide the pros and cons of their suggestion and justify their answer based on their opinion
 - Review leader decide his own recommendation himself or choose the suitable recommendation given by reviewers.
6. List and explain the types of Review.
- Code review
 - Evaluate the code structure with tester
 - Check on the standard and procedures
 - Desk checking
 - Check and correct the defects found in his code by himself.
 - Pair programming
 - One person do the code, another person will check the code
 - More effective as the author of the code might not be able to find out his mistakes in his own code. Another person can easily find out the mistakes and express it out
 - Code walkthrough
 - Has one time of meeting, discuss about the code with programmers
 - Without involves execution of code
 - Mentally figure out how the code works and test cases are being conducted
 - Technical review
 - Involves technical people of the project
 - Formal inspection
 - Involves few round of meetings, reworks and follow up
 - Programmer will explain their code
 - If there is anything to change, they will need to apply the changes and do follow up
7. Analyse the following pseudo codes/codes and perform desk checking on the changes of variable's value:

Test Data for number is 2, 3, 1, 8, 10

```
BEGIN
Count = 0
Total = 0
REPEAT
Count = count + 1
Get number
Total = Total + number
UNTIL count = 5
Display Total
END
```

Count	Number	Total
0	0	0

1	2	2
2	3	5
3	1	6
4	8	14
5	10	24

C4: Dynamic Testing

1. Differentiate Dynamic Test with Static Test.

Dynamic test

- Testing by executing the source code.
- For example, white-box testing (analyze the back-end source code to check how the code is running) and black-box testing (conducting testing without understanding how the source code works).

Static test

- Testing without running the source code
- Review the documentation.

2. Briefly explain **Black Box Testing**.

- Conduct testing without understanding how the source code works.
- Usually conducted by testers and end-users.
- Not requiring programming techniques and knowledge.
- Assess the non-functional and functional requirements.

3. An e-commerce system is built to handle the trading of computer accessories. The system processes 200000 transactions each week. The system is required to produce an analysis report according to the standard provided. Suggest the types of tests that are important for the system.

- Volume test
 - To ensure that 200000 transactions can be processed by the system each week.
- Stress test
 - Purposely set the load beyond the limitation stated which is more than 200000 transactions.
 - To identify the breaking point can be handled by the system before it is broken down.
 - Prepare for future load increments.
- Functional test
 - To test the functionalities of the system along with the increment of the data processed.

4. Relate the types of testing to the following keywords:

- a. Down time

Recoverability test

- system is able to recover itself when error occurs or the system is not working.

Stress test

- To know the breaking point of the system to know when the system will down.

- b. Response Time

Performance test

c. User interface

Usability test

- to check whether users can easily learn and use the system for their purpose efficiently.

d. Workload

Load test

- To see whether the workload can be handled.

Stress test

- To confirm maximum workload can be handled before the system is down.

Performance test

- To see how fast the system can process the workload.

4. Compare **Volume Test, Stress Test, Load Test, and Performance Test**

Volume test

- deal with huge amount of data
- To check whether the system can handle the large amount of data along with time without breaking down the system.

Stress test

- Similar to load tests, but increase the load to make them become a stress.
- To identify the breaking point / maximum limitation of the system can handle before breaking down.

Load test

- deal with transactions, number of data, amount of devices and users and network bandwidth.
- To check whether the system can handle the given load without breaking down the system.

Performance test

- Deal with the response time, how fast the system can respond to its work.
- Check how the system perform, how fast is the response time. The shorter the response time, the shorter the waiting time.

5. In what situation would an Exploratory Test can be applied? Explain your answer.

6. Differentiate **Alpha Testing** and **Beta Testing**.

Alpha testing

- Type of user acceptance testing
- Conducted at software development site
- Some end-users will be invited to the development site to conduct the alpha testing.
- Internal staff of the software company will be invited to conduct the test except developers if no end-users are invited.
- Programmers will observe the testing to collect the feedback from the users. Based on the feedback given, changes will be made.

Beta testing

- Type of user acceptance testing
- System will be installed in the actual working environment.
- As known as a field test.
- Actual end-users will use the system without observation of developers.
- Developers will still collect the feedback but the changes will not be immediately made.

7. What is Error Guessing?

- Experienced testers guess which part of the system might contain errors.
- It is not a formal testing (experience based testing) as it is based on the experiences of the experienced testers.
- Experienced testers will provide the information before conducting the formal testing.

8. A web application is developed to determine the loyalty types of a club member. If the member stays in the club less than 12 months, the member will be awarded as Normal grade, VIP grade will be awarded to whom has become member between 1 to 3 years. VVIP grade is for those members who stay more than 3 years.

Decide and apply appropriate Black Box Testing techniques to determine the number of test cases. You must show the detailed working.

- Equivalence Partitioning

Equivalence Partitioning

Parameter	Equivalence Partitioning	Representative	Expected Output
Duration, X	vEC1: $0 \leq X < 12$	10	Normal grade
	vEC2: $12 \leq X \leq 36$	20	VIP grade
	vEC3: $37 \leq X \leq 120$	100	VVIP grade

Parameter	Equivalence Partitioning	Representative	Expected Output
Duration, X	ivEC1: $X < 0$	-1	Error message: Duration cannot be in

			negative
	ivEC2: X > 120	200	Error message: Duration in exceeded the max allowed years

3 positive test cases (valid data) and 2 negative test cases (invalid data)

Boundary Value Analysis

Equivalence Partitioning	Invalid (min-)	Valid (min, min+, max-, max)	Invalid (max+)
vEC1: $0 \leq X < 12$	-1	0, 1, 10, 11	12
vEC2: $12 \leq X \leq 36$	11	12, 13, 35, 36	37
vEC3: $37 \leq X \leq 120$	36	37, 38, 119, 120	121

4 positive test cases (valid data) and 2 negative test cases (invalid data) for each partitioning.

9. A special machine is used to check and categorise the sumo wrestler contestant, there are three classes that a man can choose to join. The machine will check on the weight of the contestant. For the Heavyweight class, the man must have at least 254 lbs and above. For Middleweight, the weight of the wrestler must be between 188 lbs and 253 lbs. If a man's weight is 187 lbs and below, he can join the Lightweight class.

- Using Equivalence Class Partitioning, create a table to determine the equivalence class, representative value, parameter and expected result.
- Identify and list the test case using Boundary Value Analysis.

Equivalence Class Partitioning

Parameter	Equivalence Partitioning	Representative	Expected Output
Weight, X	vEC1: $100 \leq X \leq 187$	120	Lightweight class
	vEC2: $188 \leq X \leq 253$	210	Middleweight class
	vEC3: $254 \leq X \leq 661$	280	Heavyweight class
	ivEC1: $X < 100$	90	Error message: You didn't reach the required minimum weight.
	ivEC2: $X > 600$	210	Error message: You have exceeded the max allowed weight.

3 positive test cases (valid data) and 2 negative test cases (invalid data)

Boundary Value Analysis

Equivalence Partitioning	Invalid (min-)	Valid (min, min+, max-, max)	Invalid (max+)
vEC1: $100 \leq X \leq 187$	99	100, 101, 186, 187	188
vEC2: $188 \leq X \leq 253$	187	188, 189, 252, 253	254
vEC3: $254 \leq X \leq 600$	253	254, 255, 599, 600	601

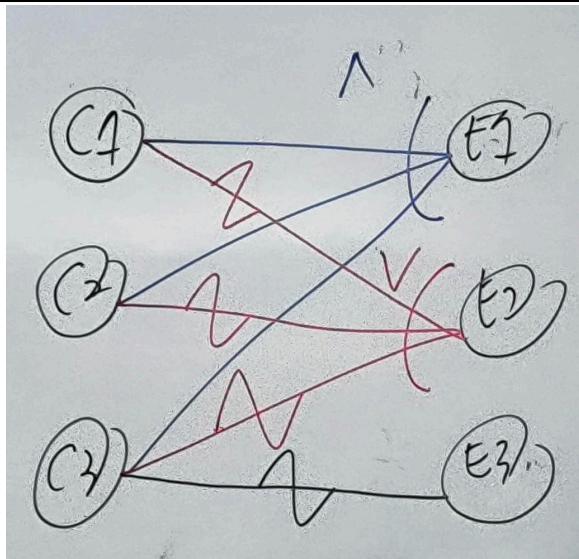
4 positive test cases (valid data) and 2 negative test cases (invalid data) for each partitioning.

10. In order for a student to graduate from a Bachelor Degree, the student must clear all payments before the final examination. The CGPA of the student to pass is at least 2.67 point and the student must fill in the feedback form before graduation. If the student does not fill in the feedback form, the student will be prohibited from graduating and the student will be requested to pay 50 dollar for additional charges.

- Construct the Cause Effect Graph.
- From the Cause Effect Graph, create a Decision Table.

C1: Clear all payment
 C2: CGPA ≥ 2.67 point
 C3: Fill in feedback form

E1: Graduate from a Bachelor Degree
 E2: Prohibited from graduating
 E3: Pay additional 50 dollar



Conditions	TC1	TC2	TC3	TC4	TC5
C1	T	F	T	T	-
C2	T	T	F	T	-
C3	T	T	T	F	F
Actions					

E1	T	-	-	-	-
E2	-	T	T	T	-
E3	-	-	-	-	T

11. AGV cinema in Malaysia has given free tickets or 50% discount to the public for the celebration of Independence Day. The cinema is using a ticket printing system to print the ticket. In order for the public who is eligible for the promotion, the person must show his Identity Card at the ticket counter. For those people who come along with his son/daughter, maximum 4 free tickets are given to each family. If the person does not come along with his son/daughter but with his wife/partner, a 50% discount is given to each ticket. For those elderly people whose age is 60 and above, free tickets will be given to them with the presence of their identity card.

- Construct the Cause Effect Graph.
- From the Cause Effect Graph, create a Decision Table.

C1: Show Identity Card at ticket counter

C2: Come along with his son/daughter

C3: Come along with his wife/partner

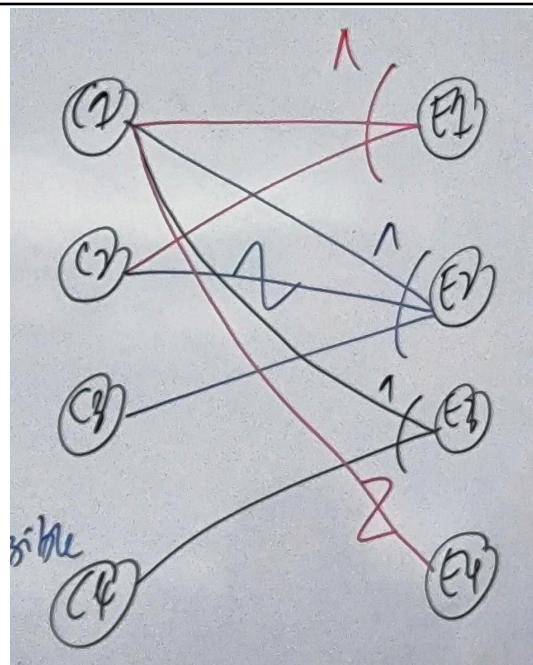
C4: Age is 60 and above

E1: Maximum 4 free tickets are given to each family

E2: 50% discount is given to each ticket

E3: Free tickets will be given

E4: Not eligible



Conditions	TC1	TC2	TC3	TC4
C1	T	T	T	F
C2	T	F	-	-
C3	-	T	-	-

C4	-	-	T	-
Actions				
E1	T	-	-	-
E2	-	T	-	-
E3	-	-	T	-
E4	-	-	-	T

Situation 2:

The “Print Number” is software that reads a series of characters (one by one) and displays messages either “Odd” or “Even”.

- The character must be a digit (number).
- If the number modulo by 2 is zero, “Even” message is printed.
- If the number modulo by 2 is not zero, “Odd” message is printed.
- If the character is not digit (number), message “Non Number” printed.

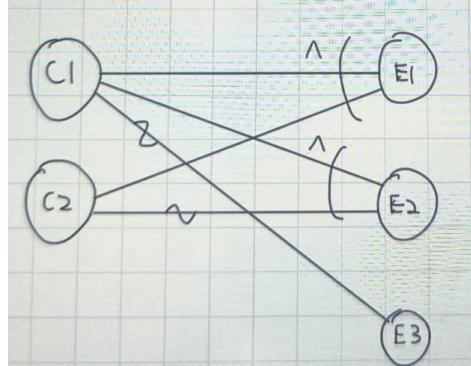
C1: The character is a digit

C2: The number modulo by 2 is zero

E1: “Even” message is printed

E2: “Odd” message is printed

E3: “Non Number” message is printed



Conditions	TC1	TC2	TC3	TC4
C1				
C2				
Actions				
E1				
E2				

E3				
----	--	--	--	--

Q2. Equivalence Partitioning

Create Equivalent Classes and Representative (Key) Values for the following situations (A and B)

Situation A:

A program calculates Christmas bonuses for employees depending on the affiliation to the company:

More than 3 years = 50% bonus

More than 5 years = 80% bonus

More than 8 years = 100% bonus

Equivalence Class Partitioning

Parameter	Equivalence Partitioning	Representative	Expected Output
Year of services, X	vEC1: $3 < X \leq 5$	4	50% bonus
	vEC2: $5 < X \leq 8$	7	80% bonus
	vEC3: $8 < X \leq 20$	9	100% bonus
	ivEC1: $X < 3$	2	Error message: The minimum year of services must be more than 3 years.
	ivEC2: $X > 20$	25	Error message: The maximum year of service must not exceed 20 years.

Boundary Value Analysis

Equivalence Partitioning	Invalid (min-)	Valid (min, min+, max-, max)	Invalid (max+)
vEC1: $3 < X \leq 5$	3	4, 5, 4, 5	6
vEC2: $5 < X \leq 8$	5	6, 7, 7, 8	9
vEC3: $8 < X \leq 20$	8	9, 10, 19, 20	21

Situation B:

A savings account in a bank has a different rate of interest depending on the balance in the account. For example, 3% rate of interest is given if the balance in the account is in the range of \$0 to \$100, 5% rate of interest is given if the balance in the account is in the range of \$100 to \$1000, and 7% rate of interest is given if the balance in the account is \$1000 and above,

Parameter	Equivalence Partitioning	Representative	Expected Output
-----------	--------------------------	----------------	-----------------

Balance, X	vEC1: $1 \leq X \leq 99$	50	3% rate of interest
	vEC2: $100 \leq X \leq 999$	650	5% rate of interest
	vEC3: $X \geq 1000$	5000	7% rate of interest
	ivEC1: $X < 1$	0	Error message: The minimum balance must be at least 1.
	ivEC2: $X > 1\ 000\ 000\ 000$	1 000 500 000	Error message: The maximum balance should not exceed 1 000 000 000.

Q3. Boundary Value Analysis

Write the test cases using Boundary Value Analysis for Situation C and D.

Situation C:

A software accepts valid User Name and Password field to work on that tool, and accepts minimum 8 characters and maximum 12 characters.

Valid range 8-12, Invalid range 7 or less than 7 and Invalid range 13 or more than 13.

Situation D:

Test cases for the application whose input box accepts numbers between 1-1000.

Valid range 1-1000, Invalid range 0 and Invalid range 1001 or more.

Boundary Value Analysis

Equivalence Partitioning	Invalid (min-)	Valid (min, min+, max-, max)	Invalid (max+)
vEC1: $7 < X \leq 12$	7	8, 9, 11, 12	13
vEC2: $1 \leq X \leq 1000$	0	1, 2, 999, 1000	1001

C5: White Box Testing

1. Explain what Control Flow Graph (CFG) is.
 - Representative of a program control flow or a program workflow or a program structure.
 - Translate the code into a graphical representation of a program structure.
2. List and describe the three (3) primitives of Control Flow Graph (CFG).
 - Decision
 - Program point at which the control can diverge (if and case statement).
 - e.g. if-else, switch, whether to enter loop or not
 - decision is created when one node has multiple edges.
 - Junction
 - Program point where the control flow can merge.
 - End loop / end if
 - Process block
 - A sequence of program statements uninterrupted by either decisions or junctions. (straight-line code)
 - One node is connected to another node via one edge.
3. Explain the following:
 - a. Statement Coverage
 - Cover all the nodes under the true path only without checking the false path.
 - The testing is running based on how the function is supposed to perform.
 - b. Branch Coverage
 - Cover all the edges where all true and false paths will be gone through.
 - c. Path Coverage
 - Test all the possible paths that might happen at least one time.
4. What is Cyclomatic Complexity?
 - A quantitative measure of logical complexity of a program.
 - It is calculated using the formula: $V(G) = E - N + 2$
 - It is applied on path coverage.
 - The result determines the complexity of the program (how many parts to be tested).
 - The result is always positive.
5. Explain the three steps to identify and define test cases.
 - Draw a corresponding control flow graph based on the source code.
 - Determine Cyclomatic complexity based on the control flow graph.
 - Determine independent paths based on the cyclomatic complexity.
 - Prepare test cases based on the independent paths.
6. Describe the four kinds of Loop Testing.
 - Simple
 - Has pre or post condition to check (for loop, while loop, do-while loop)
 - Normal loop
 - Nested
 - There may be at least one internal loop inside an external loop.
 - Concatenated

- Combination of dependent (result depends on each other) or independent (result not depend on each others, just a simple loop) loops
- Unstructured
 - Cannot be tested

7. What are the challenges in White Box testing? Using your own words, list and explain four challenges in White Box testing.

- There are requirements of programming languages for understanding the source code.
- White box testing can only test on the functionalities while black box testing is able to test on non-functionalities such as usability and performance.
- White box testing will never be ended as it is required to execute once the program is updated.
- It is time-consuming when converting the source code into control flow graph, calculating the complexity and testing all the functionalities.

8.

```

x=0;
choice = input.next.charAt (0); // accept input from keyboard
switch (choice)
{
    case '1' :
        System.out.println("Addition !");
        x = x+1;
        break;
    case '2':
        System.out.println("Subtraction!");
        x = x-1;
        break;
    case '3' :
        System.out.println("Multiplication !");
        x = x * 1;
        break;
    default :
        System.out.println("Please enter a valid choice !");
}
return 0;

```

a. Draw the Control Flow Graph for the above codes.

line 1, 2 = process block
 line 3 - decision
 line 6 - 8, 10 - 12, 14 - 16, 18 = process block
 line 19 = junction
 line 20 = process block

(no need label cases)

b. Calculate the Cyclomatic Complexity. You must show the working steps.

$$\begin{aligned}
 V(G) &= E - N + 2 \\
 V(G) &= 10 - 8 + 2 \\
 V(G) &= 4
 \end{aligned}$$

c. Identify the Basis Path. (independent path)

- Path 1: 1 - 2 - 3 - 7 - 8

- Path 2: 1 - 2 - 4 - 7 - 8
- Path 3: 1 - 2 - 5 - 7 - 8
- Path 4: 1 - 2 - 6 - 7 - 8

d. Identify the path for Branch Coverage.

- Path 1: 1 - 2 - 3 - 7 - 8
- Path 2: 1 - 2 - 4 - 7 - 8
- Path 3: 1 - 2 - 5 - 7 - 8
- Path 4: 1 - 2 - 6 - 7 - 8

* For statement coverage, only run the true paths only.

9.

```

x=true;
do
{
    gender = input.next.charAt (0); // accept input from keyboard

    if(Character.isDigit(gender) != True) //check whether the character is numeric digit
    {
        switch (gender)
        {
            case 'M' :
                System.out.println("Hello, Gentleman !");
                x = false;
                break;
            case 'F' :
                System.out.println("Hello, Lady!");
                x = false;
                break;
            default :
                System.out.println("Please enter a valid gender !");
        }
    }
}while(x);
return 0;

```

a. Draw the Control Flow Graph for the above codes.

b. Calculate the Cyclomatic Complexity. You must show the working steps.

$$V(G) = E - N + 2$$

$$V(G) = 14 - 11 + 2$$

$$V(G) = 5$$

c. Identify the Basis Path.

- Path 1: 1 - 2 - 3 - 9 - 10 - 11
- Path 2: 1 - 2 - 3 - 9 - 10 - 2 - 3 - 4 - 5 - 8 - 9 - 10 - 11
- Path 3: 1 - 2 - 3 - 4 - 5 - 8 - 9 - 10 - 11
- Path 4: 1 - 2 - 3 - 4 - 6 - 8 - 9 - 10 - 11
- Path 5: 1 - 2 - 3 - 4 - 7 - 8 - 9 - 10 - 2 - 3 - 4 - 6 - 8 - 9 - 10 - 11

* For Path 1, it would never happen logically, but we have to test it to make sure that it will not happen.

- d. Identify the path for Branch Coverage.

10.

```

x = true;
while (x)
{
    num = input.nextInt(); // accept input from keyboard
    if (num <=0)
        System.out.println("Invalid number !");
    balance = num % 2;
    switch (balance)
    {
        case 1 :
            System.out.println("You have entered an odd number !");
            x = false;
            break;
        default :
            if(balance ==2)
                System.out.println("You have entered an even number !");
            else
                System.out.println("Please enter a valid integer number !");
    }
}
return 0;

```

- a. Draw the Control Flow Graph for the above codes.
b. Calculate the Cyclomatic Complexity. You must show the working steps.

$$V(G) = E - N + 2$$

$$V(G) = 18 - 15 + 2$$

$$V(G) = 5$$

- c. Identify the Basis Path.
- Path 1: 1 - 2 - 15
 - Path 2: 1 - 2 - 3 - 5 - 6 - 7 - 8 - 13 - 14 - 2 - 15
 - Path 3:
 - Path 4:
 - Path 5:
- d. Identify the path for Statement Coverage.

Question 1

Based on the following codes:

```

x == 8;
count=0;
while (x >=0)
{
    x = x - 2;
    if ( x <= 0)
        break;
    else
        count = count++;
}

```

```
printf("%d", count);
```

- i. Draw the Control Flow Graph for the following codes
- ii. Calculate the number of basis paths using Cyclomatic Complexity and list ALL the paths.

Question 2

Analyze the codes shown in Figure 1.

```
switch (citizens)
    case Malaysia
        accessfee = 0.00
    case Singapore
        if (sex == 'M')
            accessfee = 20.00
        else
            accessfee = 18.00
    case Others
        accessfee = 1.5 * age / 3
endswitch
```

Figure 1

- i. Prepare the control flow graph by using the program shown in Figure 1 above .
- ii. Calculate the Cyclomatic Complexity from the control flow graph created.
- iii. List the basis path based on the result.

Question 3

```
1.     passingMarks = 40;
2.     Scanner input = new Scanner(System.in);
3.     System.out.println("Input marks scored by you");
4.     marksObtained = input.nextInt();
5.     if (marksObtained >= passingMarks)
6.     {
7.         if (marksObtained > 90)
8.             grade = 'A';
9.         else if (marksObtained > 75)
10.            grade = 'B';
11.        else if (marksObtained > 60)
12.            grade = 'C';
13.        else
14.            grade = 'D';
15.
16.        System.out.println("You passed the exam and your grade is " + grade);
17.    }
```

1. List the path(s) of Statement coverage.
2. List the path of Branch Coverage.

Question 4

Analyze the following Pseudo-Code:

READ X;

```

READ Y;
IF X > 0 THEN
    INCREASE X WITH 10
    Y = Y DIVIDED BY X
PRINT X
PRINT Y

```

- i. Draw the Control Flow Graph based on the pseudo-code.
- ii. Calculate the Cyclomatic Complexity.
- iii. Identify the basis path
- iv. Identify the path for Statement Coverage.
- v. Identify the path for Branch Coverage.

Question 5

Study the following codes:

```

if (v[0]<v[1]) {
    tmp = v[0];
    v[1] = v[1];
    v[1] = tmp;
}
if (v[1]<v[2]) {
    tmp = v[0];
    v[1] = v[2];
    v[2] = tmp;
}
return v;

```

- i. Create the control flow graph.
- ii. Identify the path using the following coverage criteria:
 1. Node (statement) coverage,
 2. Edge (branch) coverage,
 3. Path coverage

C6: Software Testing Life Cycle - Part 1

4. During requirement analysis, what quality factors are focused?
 - Correctness
 - Make sure the requirement which is given in SRS is correct based on what end users said.
 - Judged based on standards.
 - Completeness
 - Make sure no missing requirement happens.
 - Consistency
 - Make sure that there is no internal or external contradiction among the requirements.
 - Testability
 - Every requirement which we analyze must be able to be tested.
 - If not, we cannot convince users that the requirement can be achieved.
 - Feasibility
 - The given requirements are able to be tested based on the availability of budget, schedules, technology, infrastructure and other resources.
 - Traceability
 - Make sure that every requirement is able to be traced back.
5. Explain **Test Entry Criteria**, **Test Continuation Criteria**, and **Test Exit Criteria** of a test. Give an example for each of them.
 - Test Entry Criteria
 - What are the criterias we need to prepare or fulfil before we can enter into the testing base to start the testing.
 - E.g.
 - Necessary documentation, design and requirements information
 - Is the system ready for delivery?
 - Are the supporting utilities, accessories and prerequisites available?
 - Is the system at the appropriate level of quality?
 - Passing a smoke test conducted by the development team before moving the software to the testing team to start testing. To confirm that they have completed the development, the software is stable enough, provided with all the functionalities.
 - Is the test environment ready? (lab, hardware, software and system administration support)
 - Test Continuation Criteria
 - Decide either we are going to continue testing or put the testing on hold until the system is stable and appropriate for testing.
 - E.g.
 - Test environment must remain stable. (if the system keep on crashing during testing, we have to stop the testing)
 - Bug backlog must be manageable. (if there are too many bugs found until causing unable to continue testing system)
 - The tests for the most part unblocked (by large bugs)
 - Installable and stable test releases must be delivered regularly and properly
 - Change to the system under test must be known and controlled
 - Test Exit Criteria
 - The criterias to be met to prove that the testing has been completed.

- E.g.
 - All test cases are executed at least once
 - Min 95% of test cases pass
 - Max 5% of test cases fails

6. Explain and give an example for

- Requirement failure
- Non-requirement failure
- Waiver requested
- External failure
- Test failure
- Requirement failure
 - Defects or the bugs which can be identified during software execution happen, related to functional requirement / functionality)
- Non-requirement failure
 - Defects related to the non-functional requirements, affecting quality of system (performance, efficiency, reliability of system, etc)
- Waiver requested
 - Bugs that will not significantly affect the customers' and users' experiences of quality.
- External failure
 - The bugs which is arised from external factors or beyond the control of the system under test.
 - E.g. Network, Third-party service
- Test failure
 - Defects happen during testing and it is caused by the testers.
 - Not the development team is solving the problems, the testing team is solving it.
 - Might be due to the testing environment, testing tools, etc.

7. Discuss the tasks dependency in test estimation.

- Task dependency is the relationship between two tasks.
- Help us to do the right estimation in schedule by setting the timeline.
- E.g.
 - Finish-to-Start
 - Start-to-Start
 - Finish-to-Finish
 - Start-to-Finish

8. Identify the test planning activities.

- Test planning activities are activities that we do during testing.
- Identify and define test items to be tested, test levels, test strategies, test entry and exit criteria.
- Manage resources to allocate resources for tests.
- Control, integrate and coordinate test activities, identify the number of test activities)
- Clearly describe the details of test (test specification description) (test steps, test data)
- Risk management (project risk, product risk)

9. What are the factors of influence in test planning?

- Infrastructure for test
 - identify the availability of infrastructure (e.g. testing tools, test harness (driver and stub), procedures, standards)
- Software
 - Can the software be tested?
 - Different test plan will be needed when dealing with completed software development and almost done software development software.
- Development process
 - Maturity of the development process will affect how we should plan the testing.
 - E.g. Waterfall model (completed the development of whole system before start testing), incremental model (deliver modules by modules)
- Human factor
 - What type of testers do we have? (qualification, skills, junior or senior)

10. Explain what Test Environment (Test bed) is?

- Configured or setup environment to carry out the test cases, including hardware, software, testing tools, network, data or other special devices.
- The test environment will mimic the production environment (production environment is totally different from the development environment)

11. What is Test Effort?

- Test effort refers to how much work is required for the software testers to contribute to the project.
- E.g.
 - The number of test cases created by one person per day
 - The number of test cases executed by one person per day.
 - The effort needed to create a test environment.
 - The effort needed to train test engineers on the project.

C7: Software Testing Life Cycle - Part 2

1. Compare Test Plan with Test Case.

- Similarity
 - Software testing related documentation that is prepared before test execution is started.
- Test Plan
 - General documentation of entire planning of a testing (what to test, when to test, how to test, who will be testing the software)
- Test Case
 - A detailed document which describes step by step how the testers should test the test item which has been mentioned in the test plan.

2. What are the characteristics of good test cases?

- Accurate: Exacts the purpose.
- Economical: Each test case must be necessary to what you need to execute to prevent resource wastage.
- Traceable: All requirements must be unique to easier for being traced.
- Repeatable: Allows testers to repeat the testing over and over.
- Reusable: Test cases can be reused in new projects.

3. Compare Positive Test Case with Negative Test Case.

- Positive Test Case
 - Using valid data to do the testing.
 - Do the testing without breaking any rules in a normal environment.
 - The aim is to test the success of the functionalities.
 - Passed if it successfully performs the functions based on the valid data.
- Negative Test Case
 - Using invalid data to do the testing.
 - Conduct the testing by breaking some rules of how the system has been decided.
 - The aim is to test the error handling of the functionalities.
 - Passed if it successfully showed the error messages and vice versa.

4. Discuss the purpose of Test Monitoring.

- Track the testing works as we carry out, we cannot assume that all the tests will be completed smoothly.
- Give feedback to the test team and manager on the ongoing test, so we can know what is happening and determine how to improve the testing.
- Provide the test results to the project team.
- Measure the status of testing, test coverage and make sure that the monitoring test's test exit criteria determine whether the test is done.
- Gather the data how well we have done in the testing and use it in future testing.

5. When to start writing test cases?

- In the early stage of the software development lifecycle, we already have to do preparation for the test cases.

- When we have the analyzed requirements (has been confirmed with users), we can start writing the test cases.

6. What are the items found in a test case?

- Test case ID, Test case name
- Unit to test
- Assumptions
- Test data
- Steps to be executed
- Expected result
- Actual result
- Pass/Fail
- Comments

7. Discuss the effects of incomplete or incorrect information of a test case.

- Unable to fulfil the testing objectives that we want
- Unable to obtain the accurate result, affecting the percentage of the passed and failed test cases
- Unable to achieve test exit criterias
- Unable to identify all the hidden defects

8. Discuss the purpose of test readiness review.

- Test readiness refers to identifying whether the software is ready to move to the next testing phase.
- Conduct a review to confirm the software is ready to move to the next testing phase.
- We need to update the status to the management or client.
- The review can be formal or informal.
- Review the result of the previous testing phase and the preparation of the next testing phase. If both phases are successful, we will just move the software to the next testing phase.

9. Compare the three levels of test readiness review at application (software company) level.

- Development Test Readiness Review
 - Informal review
 - Conducted after completion of unit or module testing
- Project Test Readiness Review
 - Formal review
 - Review the result from the Software Integration Test (SIT) and the preparation of Functional Validation Test (FVT)
- Enterprise Test Readiness Review
 - Formal review
 - Review the result from the Functional Validation Test (FVT) and the preparation of Enterprise Integration Test and Performance Test.

10. Compare the two levels of test readiness review at enterprise (client) level.

- Acceptance Test Readiness Review
 - Formal test readiness review

- Review the result from Enterprise Integration Test and Performance Test and preparation of Enterprise Acceptance Test.
- Implementation Test Readiness Review
 - Formal test readiness review
 - Review the result from the Enterprise Acceptance Test and preparation to move the software into production.

C8: Software Testing Life Cycle - Part 3

1. Describe the different types of test data generators.
 - Random Test Data Generators
 - Not follow any requirement for any particular program
 - Randomly generate input
 - Cons: Quality of data is low
 - Goal Oriented
 - Generate data based on source code, convert to control flow graph
 - Identify basis path
 - Cons: Cannot smartly work, can only generate data for all the path (not specifically)
 - Pathwise
 - Generate data based on source code, convert to control flow graph
 - Generate basis path effectively and automatically, generate path based on each path
 - Intelligent
 - Analyze source code to generate test data
 - Pros: Generate test data quicker
 - Cons: complex
2. When selecting test data, there are some questions to be asked for the test data requirements. What are they?
 - What data we want to use?
 - Real data
 - day-to-day business operations
 - whatever has been stored daily
 - production data
 - Random generated data
 - Using data from fixed pool
 - Fixed pool: centralized database for testing purpose
 - What is the access we can get in real data?
 - Is it safe to use the data? (real data)
 - How big the real data is? (real data)
 - Which test data generator to use? (generate data)
 - How much data we want to generate? (generate data)
 - How much data we have from the fixed pool? (fixed pool)
 - Can the fixed pool data automatically fit or be suitable to use? Where are the data which is suitable and not suitable to be used? (fixed pool)
3. Explain the importance of test environment readiness review.
 - Test lead or manager check whether all the hardware, software, network and other facilities are prepared using checklist.
 - To ensure that all the facilities are setup and working properly before testing.
 - Test entry criteria is not achieved and testing cannot be started if test environment readiness review is not executed.
4. Analyse the challenges in using production data.
 - Need to deal with huge amount of data

- Spend too much resources and time to do data cleaning (separate junk data with actual data)
- Might not useful for newly developed system

5. Discuss the guidelines for test environment readiness.

- Stakeholder requirements collection and addressing.
- Centralized way of managing environment and test data requirements.
- Test environment management strategy.
- Multiple types or round or cycle of testing.

6. Analyse the effects of inappropriate setup of test environment.

- Software and hardware are not stable and are unable to continue testing.
- Time wastage as testing has to be on hold to check the test environment, so testing schedule will be delayed.
- Accuracy of test result will be affected.

C9: Test Management

1. Defects of software are described and handled by using defect classification and priority.
 - a. List and describe all the classes in defect classification
 - Showstopper (X)
 - Impact of defects is high severity until it disturbs the testing.
 - The testing can continue only after the defect has been resolved.
 - Most of the reason is the defects in the source code cannot fulfil the functionalities and non-functionalities where the testing team does not have an internal solution.
 - For example, broken hyperlink which will cause malfunction of functionality.
 - Critical (c)
 - Impact of the defect is high severity until it disturbs the testing but testing team has internal solution
 - Defects happens due to test environment issue such as configuration problem, hardware problem, etc.
 - Non-critical (N)
 - Defects not in the X and C, to be in N category.
 - Impact of defects is low severity where the testing, functionalities and non-functionalities would not be disturbed.
 - For example, GUI related defects such as spelling mistakes, alignment missing, picture not shown.
 - b. Provide one example for each class in (a).
 - c. List and describe all the priority levels in defect priority.
 - High
 - Defects that will stop the testing process.
 - Medium
 - Defects that may not affect the testing process.
 - But it needs to be solved as soon as possible.
 - Low
 - Can be fixed after more serious defects are fixed.
2. Define configuration management and explain why it is important.
 - Establish and maintain consistency of product's performance, functional and physical attributes with its requirements, design and operational information.
 - Help for storing, tracking and updating all system information daily.
3. Identify configuration management activities in software testing.
 - Identifying and defining the items in the system.
 - Controlling the change of these items throughout their lifecycle.
 - Recording and reporting the status of items and change requests.
 - Verifying the completeness and correctness of items.
4. What is the purpose of defect reporting?

- Testing team will report the defects found to the development team.
- Formal documentation which testers need to prepare to describe what defects they see and the priority of the defects clearly.
- So, the development team can understand the found defects and execute debugging.
- Acts as a communication medium between testing team and development team.

5. If a defect cannot be solved, suggest any professional action will be taken.

- Development team requests for a waiver and removes the module temporarily so the testing team can continue with the testing.
- The dummy data will be used for testing purposes of the next module.
- After the defects have been fixed, the module will be merged again with the testing.
- Deploy third-party more specialized and experienced developers to help fix the defects.

6. What is a test incident report?

- Any unexpected incident happened during testing is reported such as defects, system crashing, test environment break down, malfunction of testing tools.
- Defects are categorized into incidents but not only that.

C10: Software Quality Assurance (SQA)

1. Explain the differences between software quality control and software quality assurance. *
 - Software quality control
 - Check quality of the software
 - Do testing to identify the defects to fix the defects. (Software cannot be treated as a product which could be rejected)
 - Acts as validation focuses on the end product
 - To identify the defects so that they can fix the defects.
 - More to dynamic testing
 - Software quality assurance
 - Acts as verification focuses on the process
 - To prevent defects and failures
 - More to static testing
2. List the structures (categories and factors) for McCall's classic factors model. *
 - Product operation: the product can be operated
 - Product revision: the product has to be taken action such as maintenance
 - Maintainability:
 - Corrective Maintenance
 - Perfective Maintenance
 - Adaptive Maintenance
 - Product transition: software need to work with another software
 - When the software needs to work on different operating systems, there will be a transition.
 - Collaborate with third party software like payment gateway in online shopping platform.
3. What are the differences (in terms of structures) between McCall's classic factor models, Evans and Marciniak factor model and Deutsch and Willis factor model? List the differences.
 - Evans and Marciniak
 - Testability is removed.
 - Verifiability and Expandability are added.
 - Deutsch and Willis
 - Testability is removed.
 - Verifiability, Expandability, Safety, Manageability and Survivability are added.
4. The software requirement document for the tender for development of "Super-lab", a software system for managing a hospital laboratory, consists of chapters according to the required quality factors as follows: correctness, reliability, efficiency, integrity, usability, maintainability, flexibility, testability, portability, reusability and interoperability.

In the following table you will find sections taken from the mentioned requirements document. For each section, fill in the name of the factor that best fits the requirement (choose only one factor per requirements section). *

No.	Section taken from software requirements document	Requirement Factor
-----	---	--------------------

1	The probability that the “Super-lab” software system will be found in a state of failure during peak hours (9 am to 4 pm) is required to be below 0.5%.	Product operation factor: Reliability
2	The “Super-lab” software system will enable direct transfer of laboratory results to those files of hospitalized patients managed by the “MD-File” software package.	Product transition factor: Interoperability
3	The “Super-lab” software system will include a module that prepares a detailed report of the patient’s laboratory test results during his or her current hospitalization. (This report will serve as an appendix to the family physician’s file.) The time required to obtain this printed report will be less than 60 seconds; the level of accuracy and completeness will be at least 99%.	Product operation factor: Correctness
4	The “Super-lab” software to be developed for hospital laboratory use may be adapted later for private laboratory use.	Product revision factor: Flexibility
5	The training of a laboratory technician, requiring no more than three days, will enable the technician to reach level C of “Super-lab” software usage. This means that he or she will be able to manage reception of 20 patients per hour.	Product operation factor: Usability
6	The “Super-lab” software system will record a detailed users’ log. In addition, the system will report attempts by unauthorized persons to obtain medical information from the laboratory test results database. The report will include the following information: network identification of the applying terminal, system code of the employee who requested that information, day and time of attempt, and type of attempt.	Product operation factor: Integrity
7	The “Super-lab” subsystem that deals with billing patients for their tests may eventually be used as a subsystem in the “Physiotherapy Center” software package.	Product transition factor: Reusability
8	The “Super-lab” software system will process all the monthly reports for the hospital departments’ management, the hospital management, and the hospital controller according to Appendix D of the development contract.	Product operation factor: Correctness
9	The software system should be able to serve 12 workstations and eight automatic testing machines with a single model AS20 server and a CS25 communication server that will be able to serve 25 communication lines. This hardware system should conform to all availability requirements as listed in Appendix C.	Product operation factor: Efficiency
10	The “Super-lab” software package developed for the Linux operating system should be compatible for applications in a Windows NT environment.	Product transition factor: Portability

5. List and explain the components of SQA system.

- Pre-project components
- Components of project life cycle activities assessment
- Components of infrastructure error prevention and improvement
- Components of software quality management
- Components of standardization, certification and SQA system assessment
- Human components
 - All other components must depend on human components.

6. List and explain the SQA activities in Software Testing *

- Activity 1: Plan the testing and SQA processes
 - Before starting testing, we must plan the testing correctly.
 - Produce test plan and test cases.
 - Ensure a good quality of the software
- Implementation of test-oriented management
 - Use correct methodology
 - For example, extreme programming gives importance in testing.
 - Test-driven development: prepare test cases first before implementation of the source code.
 - When there is no source code yet, the test case execution will fail.
 - Then, implement the source code to make the test case pass.
 - Pair programming
 - Two developers working on a single computer.
 - One developer write code, another one review the code
- Ensure suitable work environment for SQA team
 - Respect the testers.
 - Let them understand the system to be tested.
 - Make sure there is effective communication between the software testing team and developers team.
 - Expand their knowledge and improve the work of entire team
- Optimize the use of automated tests
 - Optimize the automation testing tools to run the tests.
 - Suitable for regression testing (instead of using manual testing, use automation testing tools)
- Employ code quality measurements
 - Quality objectives should be measurable, documented, reviewed and tracked.
 - Make sure we are testing the right quality.
- Report bugs effectively
 - Prepare the defects report and send it to the developer teams.
 - A good bug report must be prepared so no misunderstanding will happen between the software testing team and developer team.

7. Discuss the benefits of applying SQA activities. *

- Improve software quality
- Customer satisfaction on the end product and customer retention
- Each team member's work would be more standardized.

8. Discuss the challenges in applying SQA activities. *

- Employees may not adapt to the SQA activities workflow or reject to apply it.
- Lack of knowledge of people who are not familiar with the SQA activities.
- Hire more employees or experts and lead to a higher cost to pay them.
- Time consuming on deploying the hardware and software needed during SQA activities.

C11: Software Quality Metrics

1. Explain the difference between **process metric** and **product metric**.

- Process metrics
 - Related to software development process
 - For example,
- Product metrics
 - Related to software maintenance
 - For example,

2. Product metric focuses on customer services performance.

a. List types of services.

- Help Desk
 - Based on customer call
 - Call the help desk when they face any issues.
 - The issue will be listed in the failure report when the help desk is unable to resolve the problem.
- Corrective maintenance services
 - Based on failure report
 - The failed help desk service phone call will be listed in the failure report.

b. Explain the difference between the services in (a).

-

c. Give an example for the difference in (b).

-

d. What are the major metrics in product metric? List all of them.

- HD quality metrics
- HD productivity and effectiveness metrics
- Corrective maintenance quality metrics
- Corrective maintenance productivity and effectiveness metrics

3. HD services are vital for successful regular use of a software system.

a. Suggest situations where the HD service is a failure.

- When the HD service is unable to solve the problem.

b. What metrics can be applied for the failure situations mentioned in (a)?

- HD calls density metrics
- Severity of HD calls metrics
- Success of the HD services

4. A software development department applies two alternative measures, NCE and WCE, to the code errors detected in its software development projects. Three classes of error severity and their relative weights are also defined:

Error Severity Class	Relative Weight
Low Severity	1
Medium Severity	3
High Severity	10

- a. Draw the table as discussed in lecture, and calculate NCE and WCE when there are 59 low severity errors, 20 medium severity errors and 11 high severity errors: (density and severity)

Error Severity Class a	Calculation of NCE (number of errors) b	Calculation of WCE	
		Relative Weight c	Weighted errors $D = b \times c$
Low Severity	59	1	59
Medium Severity	20	3	60
High Severity	11	10	110
Total	90		229

- b. If KLOC = 40, calculate the CED and WCED.

$$CED = NCE/KLOC = 90/40 = 2.25 \text{ NCE per KLOC}$$

$$WCED = WCE/KLOC = 229/40 = 5.725 \text{ WCE per KLOC}$$

5. Consider the two requirements for the successful software quality metrics.

- General requirements
 - Relevant - based on what to measure
 - Valid - measures of required attribute
 - Reliable - produces similar results when applied in similar conditions
 - Comprehensive - applicable to a large variety of situations
 - Mutually exclusive
- Operative requirements
 - Easy and simple - easy to implement the data collection with minimal resources
 - Does not require independent data collection - must not require more data outside of the project
 - Immune to biased interventions - give accurate results

6. Explain the limitations when applying software quality metrics.

- Budget constraints

- allocation necessary resources for development of quality metrics system and its regular application.
- Human factors
- Data's validity

7. Explain the benefits when applying software quality metrics.

- Reduces the ambiguity
- Helps managers to identify, prioritize, track and communicate project issues.
- Accurately describe the status of software project processes and products.
- Early discovery and correction of technical and management problems.
- Optimize software project and product performance.
- Provides an effective rationale for selecting the best alternatives.

8. Define **Total Quality Management (TQM)**.

- Continuous improvement to maintain the quality of the products and processes.
- Determine how to achieve these goals and ensure customer satisfaction.
- Total: Customers, Employees, Supplier
- Quality: Products, Processes, Work, Company
- Management: Teamwork, Ability to learn, Responsibility
- A holistically focused quality management method and considers both corporate objectives and legal requirements at all levels of a company with the participation of all employees.
- As known as total productive maintenance.
- Provides good services and products for retaining long-term success through customer satisfaction.
- Understand what the users want.

9. List and explain the eight (8) **principles** of TQM.

- Focus on customer
 - Services that are given to customers must be based on what they want.
- Employee involvement
 - Every employees must collaborate and involving in the project to achieve long-term success.
- Process centered
 - Work process based on the specified standards.
- Integrated system
- Strategic approach
 - Must understand the step-by-step strategies.
- Fact-based decisions
 - Any decision must be made based on accurate facts and data, instead of assumptions.
- Communication
- Continuous improvement

C11(additional)

1.

- (ii) The number of code error and the relative weight for each error severity class are shown in **Table 1** below:

Severity Class	Relative Weight	Number of Code Errors (NCE)
Low	2	250
Medium	5	150
High	11	75

Table 1

By referring to the figures given in the **Table 1**, compute the Total *NCE*, the *Weighted NCE (WCE)* for each severity class, the Total *WCE*, the *Code Error Density (CED)*, and the *Weighted CED (WCED)* if the KLOC is 65. (5 marks)

Severity Class	Number of Code Errors (NCE)	Relative Weight	Weighted Errors
Low	250	2	500
Medium	150	5	750
High	75	11	825
Total	475		2075

$$\text{Total NCE} = 250 + 150 + 75 = 475$$

WCE

- Low = $250 * 2 = 500$
- Medium = $150 * 5 = 750$
- High = $75 * 11 = 825$

$$\text{Total WCE} = 500 + 750 + 825 = 2075$$

$$\text{CED} = 475/65 = 7.308$$

$$\text{WCED} = 2075/65 = 31.923$$

$$\text{Average Severity of Code Errors (ASCE)} = \text{WCE/NCE} = 2075/475 = 4.368$$

2. Calculate the **Development Errors Removal Effectiveness (DERE)** if the **total number of development errors** is 65 and **Total Number of Failure Detected** in a software of a year is 300.

$$\text{DERE} = \text{NDE} / (\text{NDE} + \text{NYF}) = 65/(65+300) = 0.178$$

3. The following table shows the development milestone of a software.

Milestone	Hours of working
M1	240
M2	360

M3	160
M4	200
M5	60

Assumed that the program size is 250 KLOC, and 20% of the codes are reused from previous project. The new project has produced 580 pages of documents and 120 pages are reused from previous project. Calculate DevP, Cre, DocRe.

$$\text{DevH} = 240 + 360 + 160 + 200 + 60 = 1020$$

$$\text{Development Productivity (DevP)} = \text{DevH/KLOC} = 1020/250 = 4.08$$

$$\text{ReKLOC} = 0.2 * 250 = 50$$

$$\text{Code Reuse (CRe)} = \text{ReKLOC/KLOC} = 50/250 = 0.2$$

$$\text{ReDoc} = 120$$

$$\text{Documentation Reuse (DocRe)} = 120/580 = 0.207$$

4. An online system is expected to operate 24 hours a year (365 days). However, the total time of the system that is not able to provide some services is 144 hours. From the 144 hours, 20 hours of total time where the system is not able to perform important e-commerce activities. In addition, a total of 15 hours that the system is totally failed.

Calculate *Full Availability (FA)*, *Vital Availability (VitA)* and *Total Unavailability (TUA)* of the system.

$$\text{Number of hours software system is in service during one year (NYSerH)} = 24 * 365 = 8760$$

$$\text{Number of hours where at least one function is unavailable during one year (NYFH)} = 144$$

$$\text{Full Availability} = (\text{NYSerH} - \text{NYFH})/\text{NYSerH} = ((365 * 24) - 144) / (365 * 24) = 0.984$$

$$\text{Number of hours when at least one vital function is unavailable during one year (NYVitFH)} = 20$$

$$\text{Vital Availability (VitA)} = (\text{NYSerH} - \text{NYVitFH}) / \text{NYSerH} = (8760 - 20) / 8760 = 0.998$$

$$\text{Number of hours of total failure (all system functions failed) during one year (NYTFH)} = 15$$

$$\text{Total Unavailability (TUA)} = \text{NYTFH} / \text{NYSerH} = 15 / 8760 = 0.002$$

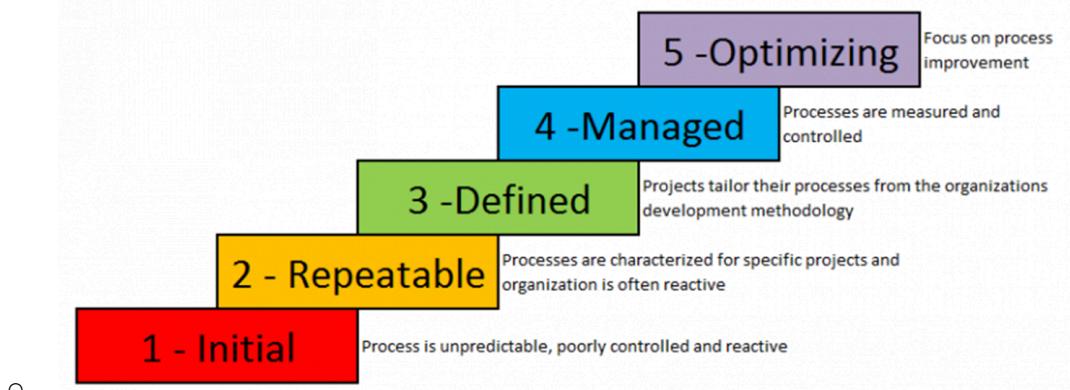
C12: Software Quality Assurance Standard

1. Discuss the benefits of using Software Quality Assurance (SQA) standards.
 - The ability to apply software development and maintenance methodologies and procedures of the highest professional level.
 - Better mutual understanding and coordination among development teams but especially between development and maintenance teams.
 - Greater cooperation between the software developer and external participants in the project.
 - Better understanding and cooperation between suppliers and customers, based on the adoption of known development and maintenance standards as part of the contract.
 2. Discuss the challenges faced in implementing SQA standard.
 - Human factors
 - Budget constraints
 3. Differentiate the SQA standard classifications, Quality Management Standard with Project Process Standard.
 - Quality management standards
 - Focus more on what the process needed to be done. (What process to perform)
 - Does not concern how you do it.
 - Includes certification and assessment methodologies.
 - Project process standards
 - Focuses on the methodologies for carrying out software development and maintenance projects. (How the process to be performed)
 4. Differentiate the two classifications of Quality Management Standard.
 - Certification standard (organization)
 - Company must comply with acceptable quality requirements.
 - This is achieved by certification granted by external body.
 - Serve as an agreed basis for customer and supplier evaluation.
 - Improve quality management system performance and enhance customer satisfaction
 - Assessment Standard (individual in organization)
 - Serve software development and maintenance organizations as a tool for self-assessment of their ability to carry out software development projects.
 - Serve as a tool for improvement of development and maintenance processes. The standard indicates directions for process improvements.
 - Help purchasing organizations determine the capabilities of potential suppliers.
 - Guide training of assessors by delineating qualifications and training program curricula.
 5. Illustrate the model of Capability Maturity Model (CMM) and Capability Maturity Model Integration (CMMI). *
- CMM is a methodology used to develop and refine an organization's software development process.
 - Acts as a benchmark to be fulfilled by projects for continuous process improvement.
 - Level 1: Initial
 - Level 2: Repeatable
 - Level 3: Defined
 - Level 4: Managed

- Level 5: Optimizing

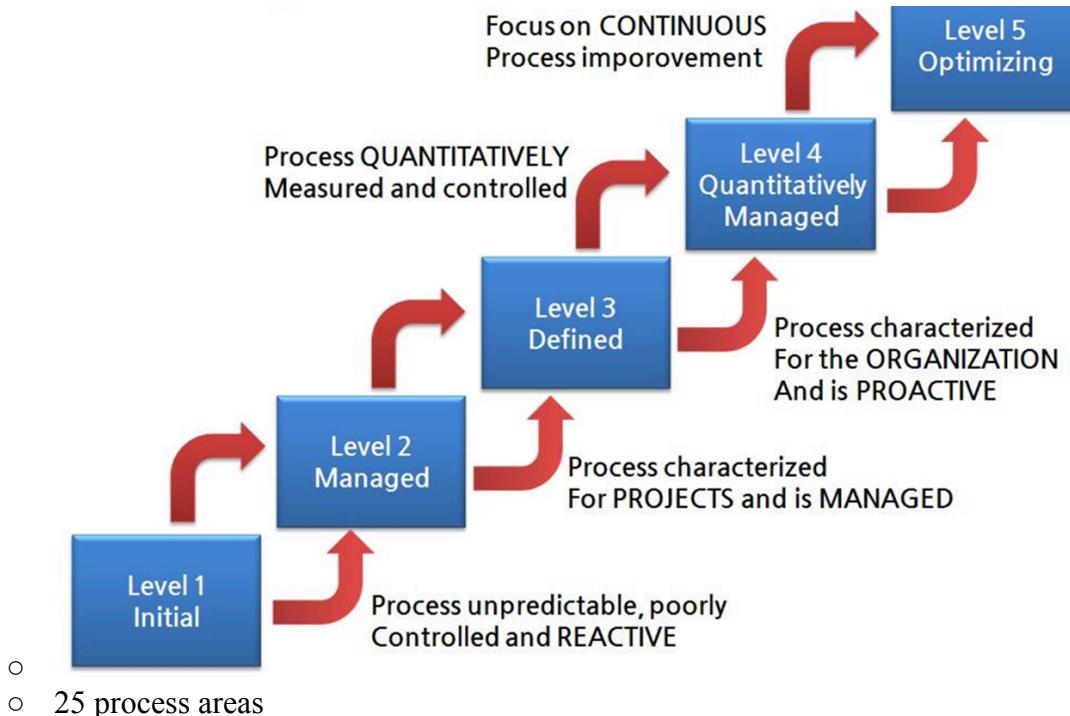
6. Draft a table to compare CMM with CMMI. *

- CMM



- 18 key process areas

- CMMI



- 25 process areas

7. Explain what are the disadvantages of CMM and how CMMI overcome it. *

- Disadvantages of CMM

- Lose perspective and forget that the real goal. Only focuses on the processes to be achieved. They don't consider whether the quality is implemented or not. So, it does not actually improve the process's quality.
- Does not specify a particular way of achieving the goals.
- Since the organization does not know the right way to do, CMM cannot be used for emergency fixing between the development of the project. Only suitable during the beginning of the project.
- 18 key process areas only focus on the improvement of management related activities but not with the context of software.

- How CMMI overcome it

- CMMI provides guidance on how to do it with the list of the processes to be done.

- Clearly defined procedures on how to implement the processes.
- Can improve the quality of the processes in the right way.