CODING

- Good software development organizations normally require their programmers to follow some well-defined and standard style of coding called coding standards.
- Most software development organizations follow their own coding standards.
- The purpose of standard style of coding is the following:
  - ✓ uniform look of the codes written by different engineers
  - ✓ easy to code understanding
  - ✓ encourages good programming practices

CODE REVIEW

What is Code Review (S-2018)

- Code review for a model is carried out after all the syntax errors have been eliminated.
- This is the cost-effective strategies for reduction in coding errors and to produce high quality code.
- Two types code review techniques are code inspection and code walk through.

CODE WALK THROUGH

Explain Code Walkthrough(S-2017,S-2016,W-2015,S-2015) OR

Explain : (a) Code walk through (b) Code inspection(S-2017,W-2016)

- Code walk through is a code analysis technique.
- In this technique carried out after a module has been coded, successfully compiled and all syntax errors eliminated.
- A few members of the development team are given the code few days before the walk through meeting to read and understand code.
- Each member selects some test cases and simulates execution of the code by hand (i.e. trace execution through each statement and function execution).
- The main objectives of the walk through are to find the algorithmic and logical errors in the code.
- The members note down their findings to discuss these in a walk through meeting where the coder of the module is present.
- Some of the guidelines for this technique:
  - ✓ It should consist of between three to seven members.
  - ✓ Discussion should focus on find the errors and not on how to fix the errors.
  - ✓ Managers should not attend the walk through meetings.

CODE INSPECTION

Explain code inspection.(S-2017,S-2016,S-2015)

- The aim of code inspection is to discover common types of errors caused due to improper programming.
- During code inspection the code is examined for the presence of certain kinds of errors.
- Error will be discovered by looking for these kinds of mistakes in the code.
- Coding standards is also checked during code inspection.
- Good software development companies collect different types of errors commonly committed by their engineers and identify the type of errors most frequently committed.
- List of commonly committed errors can be used during code inspection to look out for possible errors.
- Following is a list of some errors which can be checked during code inspection:
  - ✓ Use of uninitialized variables
  - ✓ No terminating loops
  - ✓ Incompatible assignments
  - ✓ Array indices out of bounds
  - ✓ Improper storage allocation and deallocation
  - ✓ Mismatches between actual and formal parameter in procedure calls
  - ✓ Use of incorrect logical operators or incorrect precedence among operators

SOFTWARE DOCUMENTATION

Define Software Documentation. Explain Internal Documentation.(S-2018,W-2017,S-2017,S-2016,S-2015) OR Explain Test Documentation(S-2018,W-2017,S-2017,W-2016,W-2015,S-2015)

- When various kinds of software products are developed then not only the executable files and the source code are developed but also various kinds of documents such as users' manual, software requirements specification (SRS) documents, design documents, test documents, installation manual, etc are also developed as part of any software engineering process.
- Good documents enhance understandability and maintainability of a software product.
- Documents help the users in effectively using of the system.
- Good documents help in effectively handling the manpower.
- Even when an engineer leaves the organization, and a new engineer comes in, he can build up the required knowledge easily.
- Different types of software documents can be classified into the following:
  - ✓ Internal documentation
  - ✓ External documentation

INTERNAL DOCUMENTATION

- Documentation which focuses on the information that is used to determine the software code is known as internal documentation.
- It describes the data structures, algorithms, and control flow in the programs.
- It includes header comment blocks and program comments.
- Header comment blocks are useful in identifying the purpose of the code along with details such as how the functions are used in the program.
- Since software code is updated and revised several times, it is important to keep a record of the code information so that internal documentation reflects the changes made to the software code.
- Internal documentation should explain how each code section relates to user requirements in the software. Generally, internal documentation includes the following information.
  - ✓ Name, type, and purpose of each variable and data structure used in the code
  - ✓ Brief description of algorithms, logic, and error-handling techniques
  - ✓ Information about the required input and expected output of the program
  - ✓ Information on the up gradations in the program.

EXTERNAL DOCUMENTATION

- Documentation which focuses on general description of the software code and is not concerned with its detail is known as external documentation.
- It includes information such as function of code, name of the software developer who has written the code, dependency of code on programs, and format of the output produced by the software.
- Generally, external documentation describing the design of the program.
- It consists of information such as description of the problem along with the program written to solve it.
- It describes the approach used to solve the problem, operational requirements of the program, and user interface components.
- For the purpose of readability and proper understanding, the detailed description is given by figures and illustrations that how one component is related to another.
- External documentation explains why a particular solution is chosen and implemented in the software.
- It includes formulas, conditions, and references from where the documentation is derived.
- External documentation makes the user aware of the errors that occur while running the software code. For example, if an array of five numbers is used, it should be mentioned in the external documentation that the limit of the array is five.

TESTING

- Testing a program consists of providing the program with a set of test inputs (or test cases) and observing if the program behaves as expected.
- If the program fails to behave as expected, then the conditions under which failure occurs are noted for later debugging and correction.
- Some commonly used terms associated with testing are:
  - ✓ **Error:** It is a mistake committed by the development team during any of the development phases. An error is also called fault, a bug, or a defect.
  - ✓ **Failure**: This is a form of an error (or defect or bug). Due to Error program can failure.
  - ✓ **Test case**: This is the triplet [I,S,O], where I is the data input to the system, S is the state of the system at which the data is input, and O is the expected output of the system.
  - ✓ **Test suite**: This is the set of all test cases with which a given software product is to be tested.

Verification and validation

Differentiate between verification and validation.(W-2017,W-2015,S-2015)

- ✓ Verification is the process of determining the output of one phase.
- ✓ Validation is the process of determining fully developed system conforms to its requirements.
- ✓ Thus while verification is concerned with phase containment of errors, the aim of validation is that the final product be error free.

Verification	Validation
Verification is a static practice of verifying documents, design, code and program.	**Validation** is a dynamic mechanism of validating and testing the actual product.
It does not involve executing the code	It always involves executing the code
It is human based checking of documents and files.	It is computer based execution of program
Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	**Validation** uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc
It generally comes first-done before validation	It generally follows after **verification**

Testing activities

- **Test suit design:** The set of test cases using which a program is to be tested is designed.

- **Running test cases and checking the result to detect failures:** Each test case is run and the results are compared with the expected results. A mismatch between the actual result and expected result indicates a failure. The test cases for which the system fails are noted down for later debugging.
- **Debugging:** Debugging is carried out to identify the statements that are in error. Failure is analyzed to locate the errors.
- **Error correction:** Error is located in previous activity the code is changed to correct the error.

UNIT TESTING

Explain Unit testing (S-2018,W-2017,S-2017,W-2016,W-2015,S-2015)

- Unit testing is undertaken after a module has been coded and successfully reviewed.
- Unit testing (or module testing) is the testing of different units (or modules) of a system in separately.
- When developer is coding the software it may happen that the dependent modules are not completed for testing, in such cases developers use stubs and drivers to simulate the called (stub) and caller (driver) units. Unit testing requires stubs and drivers, stub represent the called unit and driver represent the calling unit.

Stub

- Assume you have 3 modules, Module A, Module B and module C. Module A is ready and we need to test it, but module A calls functions from Module B and C which are not ready, so developer will write a dummy module which simulates B and C and returns values to module A. This dummy module code is known as stub.

Driver

- Now suppose you have modules B and C ready but module A which calls functions from module B and C is not ready so developer will write a dummy piece of code for module A which will return values to module B and C. This dummy piece of code is known as driver.

BLACK BOX TESTING

Explain Black Box Testing(W-2017,W-2016,S-2016)

- In the black-box testing, test cases are designed from an examination of the input/output values only and no knowledge of design or code is required. The following are the two main approaches to designing black box test cases.
  - ✓ Equivalence class portioning
  - ✓ Boundary value analysis

Equivalence Class Partitioning

- In this approach, the domain of input values to a program is partitioned into a set of equivalence classes. This partitioning is done such that the behavior of the program is similar for every input data belonging to the same equivalence class.
- In this approach testing the code with any one value belonging to an equivalence class is as good as testing the software with any other value belonging to that equivalence class.
- Equivalence classes can be designed by examining the input data and output data. The following are some general guidelines for designing the equivalence classes:
  - ✓ If the input data values to a system can be specified by a range of values, then one valid and two invalid equivalence classes should be defined. For example if the equivalence class is the set of integers in the range 1 to 10 (e.g. [1, 10]) then the invalid equivalence classes are [-∞, 0], [11, +∞]
  - ✓ If the input data assumes values from a set of discrete members of some domain, then one equivalence class for valid input values and another equivalence class for invalid input values should be defined. For example if the valid equivalence classes are {A,B,C} then the invalid equivalence class is U-{A,B,C} where U is the universe of possible input values.
- In short it is the process of taking all possible test cases and placing them into classes. One test value is picked from each class while testing.
- For example, **Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:**
  1. One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 then result is going to be same. So one test case for valid input data should be sufficient.
  2. Input data with any value less than 1 (Lower Limit) to represent invalid input class.
  3. Input data with any value greater than 1000 (Upper Limit) to represent invalid input class.
- So using equivalence partitioning you has categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.

Boundary value analysis

- It's widely recognized that input values at the extreme ends of input domain cause more errors in system. More application errors occur at the boundaries of input domain.
- Boundary value analysis testing technique is used to identify errors at boundaries rather than finding those exist in center of input domain.
- Boundary value analysis is a next part of Equivalence partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.
- Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:

1. Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000.
2. Test data with values just below the extreme edges of input domains i.e. values 0 and 999.
3. Test data with values just above the extreme edges of input domain i.e. values 2 and 1001.

WHITE BOX TESTING

Explain White Box testing method in brief. (W-2017,W-2016,S-2016,W-2015,S-2015)

- In this method of testing the test cases are calculated based on analysis internal structure of the system based on Code coverage, branches coverage, paths coverage, condition Coverage etc.
- White box testing involves the testing by looking at the internal structure of the code & when you completely aware of the internal structure of the code then you can run your test cases & check whether the system meet requirements mentioned in the specification document.
- Based on derived test cases the user exercised the test cases by giving the input to the system and checking for expected outputs with actual output.

Statement coverage

- The statement coverage strategy aims to design test cases so that every statement in a program is executed at least once.
- The principal idea governing the statement coverage strategy is that unless a statement is executed, it is very hard to determine if an error exists in that statement.
- Unless a statement is executed, it is very difficult to observe whether it causes failure due to some illegal memory access, wrong result computation, etc.
- For example:
  If(a>b)
  printf(“a is greater”)
  else
  printf(“b is greater than”)
- Test cases for above example: {a=5,b=10}, {a=10,b=5}

Branch coverage

- In the branch coverage-based testing strategy, test cases are designed to make each branch condition to assume true and false values in turn.
- Branch testing is also known as edge testing as in this testing scheme, each edge of a program’s control flow graph is traversed at least once.

- It is obvious that branch testing guarantees statement coverage and thus is a stronger testing strategy compared to the statement coverage-based testing.
- For example, find maximum from three number:

```
If(a>b && a>c)
{ max=a; }
else if (b>c)
{ max=b; }
else
{ max=c; }
```

- Test cases for above example: {a=5,b=10,c=15}, {a=5,b=15,c=10}, {a=15, b=5, c=10}

Condition coverage

- In this structural testing, test cases are designed to make each component of a composite conditional expression to assume both true and false values.
- For example, in the conditional expression ((c1.and.c2).or.c3), the components c1, c2 and c3 are each made to assume both true and false values.
- Branch testing is probably the simplest condition testing strategy where only the compound conditions appearing in the different branch statements are made to assume the true and false values.
- Thus, condition testing is a stronger testing strategy than branch testing and branch testing is stronger testing strategy than the statement coverage-based testing.
- For a composite conditional expression of n components, for condition coverage, 2^n test cases are required. Thus, for condition coverage, the number of test cases increases exponentially with the number of component conditions.
- Therefore, a condition coverage-based testing technique is practical only if n (the number of conditions) is small.

Path coverage

- The path coverage-based testing strategy requires us to design test cases such that all linearly independent paths in the program are executed at least once.
- A linearly independent path can be defined in terms of the control flow graph (CFG) of a program.
- Path testing is used for module or unit testing.
- It requires complete knowledge of the program structure.

Compare Black box and White box testing(S-2017)

Black Box Testing	White Box Testing
Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program	White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.
This type of testing is carried out by testers	Generally, this type of testing is carried out by software developers
Implementation Knowledge is not required to carry out Black Box Testing	Implementation Knowledge is required to carry out White Box Testing
Programming Knowledge is not required to carry out Black Box Testing	Programming Knowledge is required to carry out White Box Testing
Testing is applicable on higher levels of testing like System Testing, Acceptance testing	Testing is applicable on lower level of testing like Unit Testing, Integration testing
Black box testing means functional test or external testing.	White box testing means structural test or interior testing
In Black Box testing is primarily concentrate on the functionality of the system under test	In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc
Black Box testing can be started based on Requirement Specifications documents	White Box testing can be started based on Detail Design documents.

2 Marks QUE-ANS

Explain Test Cases.(S-2018)

A Test Case is a set of actions executed to verify a particular feature or functionality of your software application.

Test Case 1: Check results on entering valid User Id & Password

Test Case 2: Check results on entering Invalid User ID & Password

Test Case 3: Check response when User ID is Empty & Login Button is pressed, and many more