Scoring Indicators (Version B)

COURSE NAME : SOFTWARE TESTING

COURSE CODE : 6131 C **QID : 2102240142**

PART A

I. Answer all the following questions in one word or sentence.

(9 x 1 = 9 Marks)

Max. marks

Q.No.	Scoring Indicators	Split score	Sub Total	Total score
	PART A			24
I.1	o Reduced maintenance cost o Improved software testing process o Bug prevention (Any two)		1	
I.2	Failure means the inability of a system or component to perform a required function according to its specification.		1	
I.3	4.n + 1		1	
I.4	An FSM (Finite State Machine) is a behavioural model whose outcome depends upon both previous and current inputs		1	
I.5	Function testing		1	
I.6	Stubs and drivers.		1	
I.7	A system under test (SUT) is said to regress if : o A modified component fails o A new component causes failure in the unchanged components by generating side effects.		1	
I.8	Test Data generator Test Case generator		1	
I.9	WinRunner, SilkTest, LoadRunner, Jmeter, Test Director (Any two)		1	
				24

	PART B			
II.1	**OPEN** When the test leader approves that the bug is genuine ,it's state becomes open. **ASSIGN** If the bug is valid ,a developer is assigned the job to fix it and the state of the bug now is 'assign'. **DEFERRED** If the priority of reported bug is not high or there is not sufficient time to set it or the bug does not have any adverse effect on the software ,then the bug is changed to deferred state which implies the bug is expected to be fixed in next releases.			
II.2	• Testing Tactics are the ways to perform various types of testing. This can be done in two ways: Manual testing and Automated testing. o Complete and exhaustive testing is not possible o Our effort should be for effective testing o The main objectives of effective test case design are: Coverage of testing domain Detection of maximum number of bugs The software testing techniques can be categorized into two parts : Static testing & Dynamic Testing Static Testing Dynamic Testing • Dynamic testing is further divided in to two parts : black-box testing & white-box testing			
II.3	Need of White-box Testing • White-box testing is the initial stage of testing of modules and black-box testing is the second stage. Black-box testing cannot be executed until the code is produced and checked using white-box testing techniques. So white-box testing is essential. • There are categories of bugs that can be revealed by white-box testing and not by black-box	1 2		

	testing. So it is necessary. • Design phase errors will be reflected in code. So white-box testing must be done for code verification (unit verification) • White-box testing explores all logical paths which are commonly as well as rarely executing. • White-box testing helps to detect typographical errors which are not covered by blac-box Testing. (Any 3)			
II.4	Detect the extreme values, Min, Min+, Max, Max- and nominal value. To understand. Design test cases. No. of test cases = 4x1 + 1 = 5			
II.5	State graphs of larger problems may not be easy to understand. So they are converted into state graphs. The following conventions are used for state table. • Each row of the table represents a state • Each column corresponds to an input event • Intersection of columns and rows specified the next state (transition) and the output, if any.			
II.6	1. Author/Owner/Producer : A programmer or designer responsible for producing the program or document and fixing the discovered defects. 2. Inspector : A peer member of the team who is not a manager or supervisor and responsible to find errors, omissions and inconsistencies in programs and documents 3. Moderator: A team member who manages the whole inspection process 4. Recorder : a member who records all the results of the inspection meeting			
II.7	Guidelines for top-down integration: • The module which is ready to be integrated, will be			

	integrated and tested first i.e. use the availability of modules. • Critical sections of the module will be added and tested as early as possible • I/O modules are added as early as possible so that all interface errors will be detected earlier			
II.8	Objectives of Regression Testing 1. It tests to check that the bug has been addressed • After bug-fixing, run the same test that was executed when the problem was first found. If program fails, bug has not been fixed correctly and no need to do any regression testing 2. It finds other related bugs • Regression tests are necessary to validate that the system does not have any related bugs. 3. It tests to check the effect on other parts of the program			
II.9	1. Diversity and Complexity : • Web applications interact with many components that run on diverse hardware and software platforms • They are written in diverse languages and they are based on different programming approaches such as procedural, OO, and hybrid languages such as JSP • The client side includes browsers, HTML, embedded scripting languages and applets • The server side includes CGI, JSPs, Java Servlets and .NET technologies • They all interact with diverse back-end engines found on web server or other servers 2. Dynamic Environment 3. Very short development time 4. Continuous evolution 5. Compatibility and Interoperability			
II.10	Static testing tools consists of tools for the following : i. Control flow analysis			

	ii. Data use analysis iii. Interface analysis iv. Path analysis b. Dynamic testing tools support the following: Dynamic testing activities They are called program monitors. Short description of program monitors.			
	PART C			42
III	Goals of software testing can be classified broadly into 3 categories • Short-term or immediate goals : • Long-term goals : • Post-implementation goals Short-term or immediate goals : These goals are the immediate results after performing testing. o Bug Discovery : Bug discovery refers to find errors at any stage of software development. The goal is to discover more bugs at early stages itself. • Long-term goals : These goals affect the product quality in the long run. o Quality : o Reliability is the major factor to achieve quality. The confidence in reliability increases quality o Customer Satisfaction : It is the prime concern of testing All specified and unspecified requirements must be tested for customer satisfaction A complete testing process achieves reliability, reliability enhances quality, and quality in turn increases customer satisfaction . o Risk Management : Risk is the probability that undesirable events will occur in a			

	system • Post-implementation goals These goals are important after the product is released o Reduced maintenance cost o Improved software testing process o Bug prevention			
IV	Software Testing Life Cycle (STLC) • Software testing process is divided into a well-defined sequence of steps known as software Testing Life Cycle (STLC). • STLC allows testers to involve at early stages of development which has significant benefit in the project cost and schedule • STLC helps management in measuring specific milestones. • STLC consists of four phases : Test Planning , Test Design, Test Execution & Post-Execution/ test review (Figure- 2 marks, Brief explanation of phases – 4 marks)			
V	Basis path testing is the oldest structural testing technique • It is based on control structure of the program • Basis path testing is the technique of selecting the paths through the program. Applications of Path Testing 1) Thorough Test / More Coverage • Path testing provides us best code coverage, leading to a thorough testing. Basis path set provides the number of test cases to be executed for full coverage. 2) Unit Testing • Path testing is mainly used for structural testing of a module. Path testing uncovers errors in decision outcomes and prepare each module for integration	1 6		

	3) Integration Testing • Path testing analyses all the paths on the interface and explores all possible errors that can occur while calling other modules 4) Maintenance Testing • Path testing is also necessary with modified versions of the software. 5) Testing effort is proportional to Complexity of software • Path testing takes care of the complexity of the software and derives the number of tests to be carried out. 6) Basis path testing is concentrated on error-prone software • Cyclomatic complexity number (no. of test cases needed) signifies that the testing effort is only on the error-prone part of the software, thus minimizing the testing effort.ovide a basis set of execution paths through the program.			
VI	There are two steps to perform equivalence class testing 1. Identify equivalence classes 2. Design test cases • Step 1 : Identification of Equivalent classes 1. Grouping inputs for which behavior pattern of the module is similar. i. For example, in a module to determine absolute value for integers, we can form two classes : one is class for +ve integers & another for –ve integers. 2. Two types of classes can always be identified as discussed below: i. Valid Input Class: Considers only valid inputs to the program ii. Invalid Input class : Considers invalid inputs which generate errors or unexpected behavior. 3. Guidelines to form equivalent classes : • Step 2: Identifying Test Cases 1. Identify a unique number to each equivalent class 2. Design test cases covering as many of the uncovered valid equivalent classes as possible until all valid equivalent classes			

	have been covered 3. Design individual test cases to cover each invalid equivalent classes			
VII	Two steps in decision table based testing 1.Formation of decision table (3 marks) 2. Test case design using decision table. (4 marks)			
VIII	Test cases for structural testing are designed based on logic of the program. Every element of the logic must be covered by the test cases. Basic forms of logic coverage are : o Statement coverage o Decision or Branch coverage o Condition coverage o Decision/Condition coverage o Multiple Condition Coverage (Listing of logic coverage 2 marks) (Brief explanation of each 5 marks)			
IX	Call Graph-based integration • Usually integration testing performs structural testing. If we can draw the module calling graph, it is possible to perform behavioural testing at the integration level. • A call graph is a directed graph where nodes represent modules and directed edges means one module has called another module. The call graph can also be represented in adjacency matrix.There are two types of integration based on call graph 1. Pair-wise Integration • Consider only one pair of calling and called modules and thus make a set of pairs for all such module. 2. Neighbourhood Integration • In this method, number of test sessions can be reduced by			

	considering neighbourhood for a node. • The neighbourhood for a node is the immediate predecessor or immediate successor nodes. It can be a set of nodes that are one edge away from the given node.			
X	**Regression Testing produces quality software** • The creation, maintenance and execution of regression test suite helps to retain and improve the quality of software. • Due to following reasons, regression testing is very important: o It validates the parts of the software where changes occur o It validates the related parts of the software which may be affected by some changes. o It ensures the proper functioning of software as it was before modifications o It enhances the quality of software, as it reduces the risk of bugs Figure	5 2		
XI	Need of Integration testing : Integration testing can find out inconsistency between the modules such as improper call or return sequences. o Data can be lost across an interface o Combining modules may not give desired results always o Data types and their valid ranges may mismatch between the modules. Decomposition-based integration • This type of integration is based on the decomposition of design into functional components or modules. • Integrate all the modules together and then test it • Integrate the modules one by one and test them incrementally Integration testing methods are classified into two categories : Non-Incremental Integration Testing	3		

	Incremental testing (Brief explanation of both)	4		
XII	The function test must ensure : o Each component or business event performs in accordance to the specifications o Each component responds correctly to all conditions o Moves data correctly from one business event to the next o Each event is initiated in the order required to meet the business objective o Function testing can be performed after unit and integration testing o To keep a record of function testing, function coverage metric is used. An effective function test cycle have a defined set of processes and deliverables as shown below: Test Planning Requirement definition Test case design Traceability matrix formation Test Case Execution			
XIII	ISSUES ON OBJECTED ORIENTED TESTING 1. Basic unit of testing • Class is the natural unit for test case design • Other units are aggregation of classes such as class clusters and application systems • There are different types of classes : application-specific vs general-purpose, abstract classes, parameterized or template classes . This demands different test requirements • Testing an object (class instance) verify a class. When objects are used in applications, the entire system must be tested as a whole .			

	2. Implications of inheritance • The inherited features require retesting because these features are now in a new context of usage 3. Polymorphism • It is difficult to find out all polymorphic bindings which requires a separate test to validate all of them 4. White-box testing • Conventional white-box testing cannot be applied on testing a class. 5. Black-box testing • Retesting of inherited features require examination of class structure. This limits the use of black-box testing technique for OO software 6. Integration strategies • There is no clear hierarchy of methods and classes. So conventional methods of incremental integration testing cannot be adopted. Therefore it is required to devise new integration methods.			
XIV	Selenium is an open-source suite of tools and libraries that is used for browser automation. Selenium is an open-source, automated testing tool used to test web applications across various browsers. Selenium can only test web applications, Selenium is used to: It allows users to test their websites functionally on different browsers. Perform Cross browser testing to check if the website functions consistently across different browsers.			