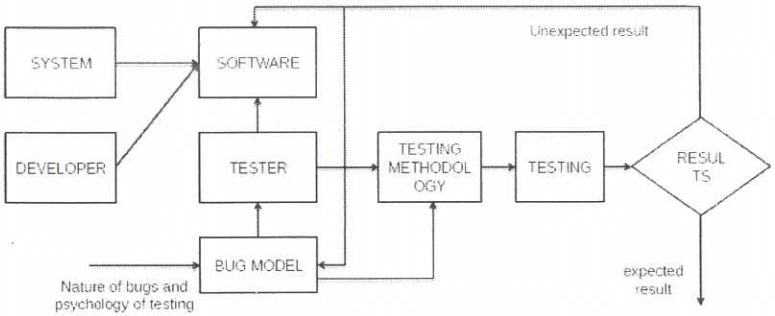
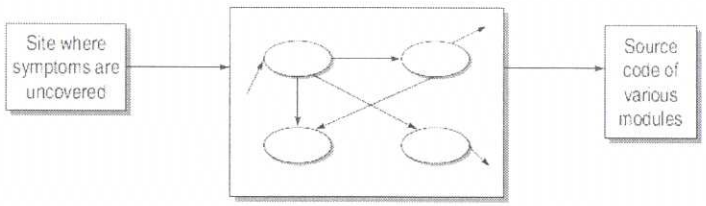
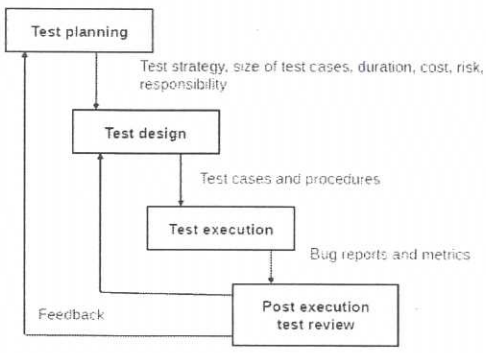
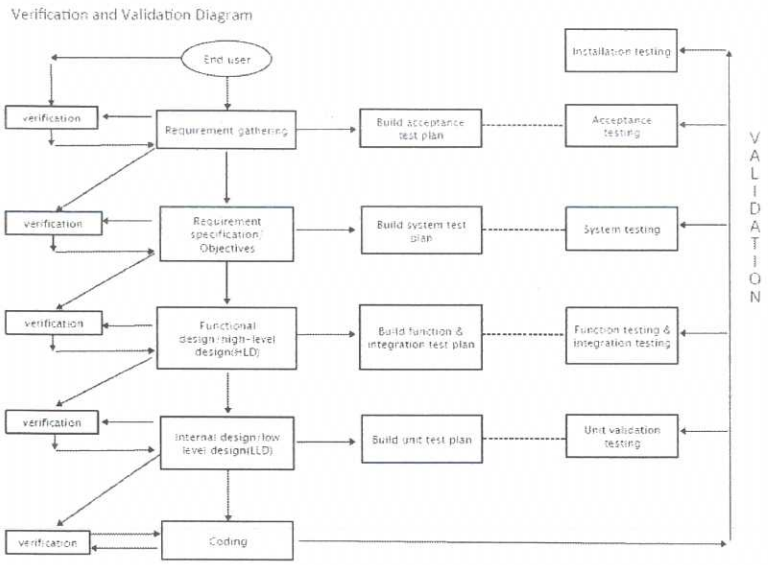
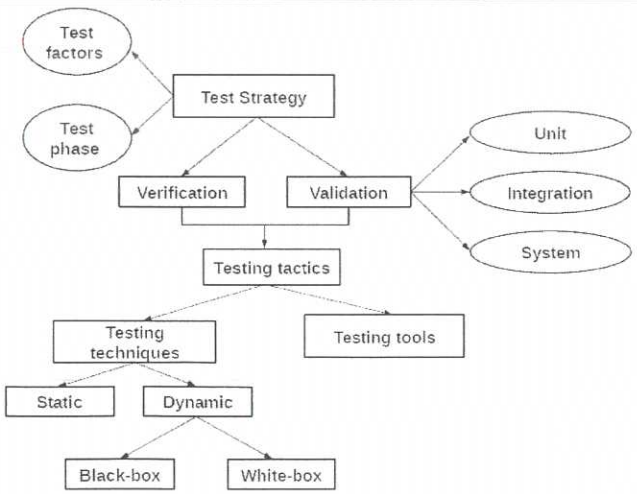
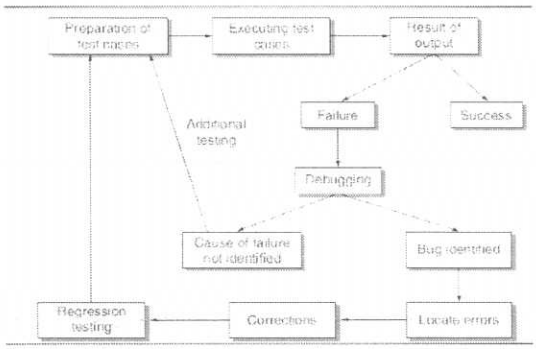
SCHEME OF VALUATION
(Scoring Indicators)

Revision: 2015 Course Title: Software Testing		Course Code: 6136		
Qst No	**Scoring Indicators**	**Split up Score**	**Sub Total**	**Total**
I (1)	1. Reduced maintenance cost: • If a software is properly tested, the bugs and failures of that software will decreases, ie, the maintenance cost can be save for us. 2. Improved software testing process: • A testing process one project may not be successful and there may be scope for improvement. • Post implementation results can be analysed to find out snags in the present testing process.	1 * 2	2	2
I (2)	Boundary value checking Robustness testing method Worst-case testing method Robust worst-case testing method	4 * 0.5	2	2
I (3)	To check that the bug has been addressed To find other related bugs To test that the fixes have not created any other problems elsewhere To check the effect on other parts of the program	4 * 0.5	2	2
I (4)	CUT, Cgreen, Emma and Findbugs WinRunner, SilkTest, LoadRunner, Jmeter, and TestDirector (any four)	4 * 0.5	2	2
I (5)	To determine the exact nature of failure with the help of symptoms identified, locate the bugs and errors and finally correct it	2	2	2
II (1)	 Software and software model Bug model Testing methodology and testing	2 1 1 2	6	6
II (2)	Test plan Test case design Test execution	2 * 3	6	6

II (3)	Divide or partition the input domain based on a common feature Identification of Equivalent Classes Design test cases	3 * 2	6	6
II (4)	1. Decomposition based integration Non-incremental integration testing, Incremental integration testing 2. Call graph based integration Pair-wise integration, Neighborhood integration 3. Path based integration	3 * 2	6	6
II (5)	Reduction of testing effort, Reduces the testers involvement in executing tests, Facilitates regression testing, Avoids human mistakes, Reduces overall cost of software, Simulated testing, Internal testing, Test enablers, Test case design	6 * 1	6	6
II (6)	1. Static testing tools: Control flow analysis, Data use analysis, Interface analysis, Path analysis 2. Dynamic testing tools	2 * 3	6	6
II (7)	Steps for the backtracking process 1. Observe the symptom of failures 2. Trace the source code backwards 3. Isolate the module 4. Logical backtracking in the isolated module 	4 * 1 2	6	6
III (a)	 Test planning & Test design Preparing list of items to be tested Mapping items to be tested, Selection of test case design techniques Creating test cases & test data Setting up the test environment and supporting tools, Creating test procedure	2 1 1 1 1 1	8	8

	specification Test Execution & Test Review	1		
III (b)	 1. Verification Activities Verification of Requirement and Objectives Verification of High-Level Design Verification of Low-Level Design Verification of Coding (Unit Verification) 2. Validation Activities Validation Test Plan Validation Test Execution	3 2 2	7	7
IV	 Software Testing Strategy Testing tactics Testing techniques Testing tools	3 3 4 2	15	15

V(a)	Method to represent the information in a tabular method. Complex combinations of input conditions and resulting actions 1. Formation of Decision table Condition stub, Action stub, Condition entry, Action entry 2. Test case design using decision table 3. Expanding immaterial cases in decision table	1 3 2 2	8	8
V(b)	The process of attempting to detect discrepancies between the functional specifications of a software and its actual behaviour 1. Test planning 2. Partitioning / Functional decomposition 3. Requirement definition 4. Test case design 5. Traceability matrix formation 6. Test case execution	1 1 1 1 1 1 1	7	7
VI(a)	Considers the program code and test cases are designed based on the logic of the program 1. Statement coverage 2. Decision or branch coverage 3. Condition coverage 4. Decision/condition coverage 5. Multiple condition coverage	1 2 2 1 1 1	8	8
VI(b)	Inspection process is carried out by a group of peers. First inspect the product at the individual level. Discuss the potential defects of the product observed in a formal meeting. 1. Planning – plan for inspection meeting 2. Overview – Define purpose of inspection meeting 3. Individual preparation – Team members identify potential errors 4. Inspection meeting – Item is inspected in inspection meeting 5. Rework – Bugs which are reported in inspection meeting are fixed 6. Follow-up – Planning and overview	1 1 1 1 1 1 1	7	7
VII(a)	1. Match the tool to its appropriate use 2. Select the tool to its appropriate SDLC phase 3. Select the tool to the skill of the tester 4. Select a tool which is affordable 5. Determine how many tools are required for testing the system 6. Select the tool after examining the schedule of testing	1.5 1.5 1.5 1.5 1 1	8	8
VII(b)	1. Method-level testing 2. Class-level testing 3. Cluster-level testing 4. System-level testing	1 3 * 2	7	7
VIII(a)	1. Consider building a tool instead of buying one 2. Test the tool on an application prototype 3. Not all the tests should be automated 4. Select the tools according to organizational needs 5. Use proven test-script development techniques 6. Automate the regression tests whenever feasible	1 6 * 1	7	7

VIII(b)	1. Diversity and complexity 2. Dynamic environment 3. Very short development time 4. Compatibility and interoperability 5. Continuous evolution	1.5 1.5 1.5 1.5 1	7	7
IX(a)	To determine exact nature of failure with the help of symptoms identified, locate the bugs and errors, and correct it. 1. Check the result of the output 2. Analyze the failure occurred 3. Identify the cause of failures 4. Trace the actual location of the error 5. Correct the errors 6. Regression test to validate the corrections 	5 3	8	8
IX(b)	1. Fresh thinking leads to good debugging 2. Don't isolate the bug from your colleagues 3. Don't attempt code modifications in the first attempt 4. Additional test cases are a must if you don't get the symptom or clues to solve the problems 5. Regression testing is a must after debugging 6. Design should be referred before fixing the error	1 6 * 1	7	7
X(a)	1. Debugging with memory dump 2. Debugging with watch points: Output statements, Breakpoint execution, Single stepping, Step-into, Step-over	3 5	8	8
X(b)	To speed-en up the debugging process To track down, isolate and remove bugs Illustrate the dynamic nature of a program Understand a program as well as to find and fix the bugs Give fine control over the program under test 1. Kernel Debugger 2. Basic machine level debugger 3. In-circuit emulator 4. Interpretive programming environment debugger	3 4	7	7