# [Feb 11, 2024 CTFL_Syll2018 Exam Brain Dumps - Study Notes and Theory [Q190-Q206


[Feb 11, 2024] CTFL_Syll2018 Exam Brain Dumps - Study Notes and Theory
Pass ISQI CTFL_Syll2018 Test Practice Test Questions Exam Dumps

**Q190.** The flow graph below shows the logic of a program for which 100% statement coverage and 100% decision coverage is required on exit from component testing. [K4] The following test cases have been run:

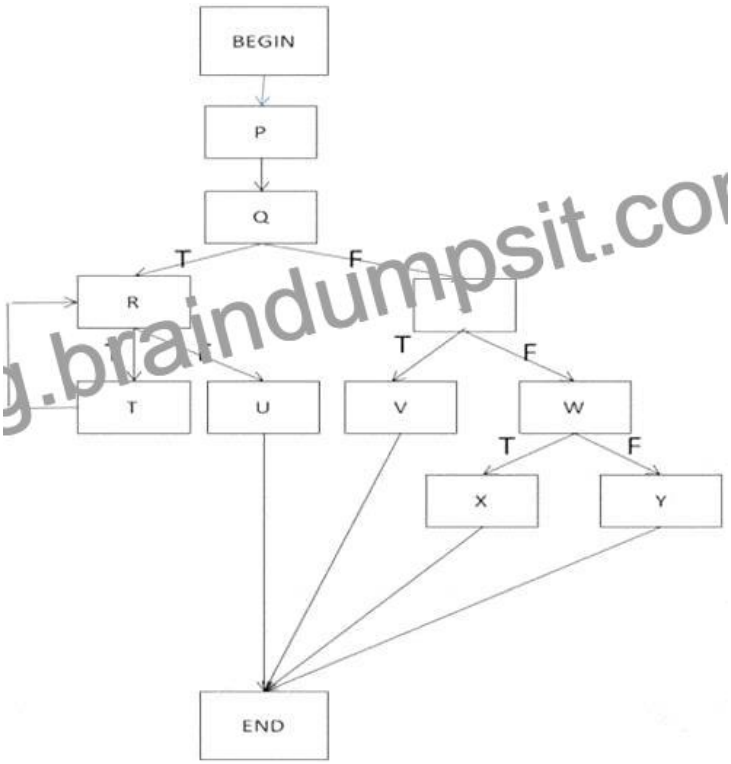
Test Case 1 covering path P,Q,R,U

Test Case 2 covering path P,Q,S,V

Test Case 3 covering path P,Q,S,W,X

Test case 4 covering path P,Q,S,W,Y

Refer to the exhibit






* Statement coverage is 100%; decision coverage is 100%
* Statement coverage is less than 100%; decision coverage is 100%.
* Statement coverage is 100%; decision coverage is less than 100%
* Statement coverage and decision coverage are both less than 100%

Statement coverage and decision coverage are both less than 100%. Statement coverage is a criterion that requires every executable statement in the source code to be executed at least once by a test suite1. Decision coverage is a criterion that requires every decision (or branch) in the source code to take both true and false outcomes at least once by a test suite1. The flow graph below shows the logic of a program for which 100% statement coverage and 100% decision coverage is required on exit from component testing. The flow graph has 9 nodes (A, B, C, D, E, F, G, H, I) and 10 edges (1, 2, 3, 4, 5, 6, 7, 8, 9, 10). The edges represent executable statements and the nodes represent decisions or branches.

!flow graph

The following test cases have been run:

* Test Case 1 covering path A-B-C-D-E-F-G-H-I

* Test Case 2 covering path A-B-C-D-E-F-G-H

* Test Case 3 covering path A-B-C-D-E-F

* Test case 4 covering path A-B-C-D-E

To calculate the statement coverage and decision coverage achieved by these test cases, we can use the following formulas:

* Statement coverage = (Number of statements executed / Total number of statements) x 100%

* Decision coverage = (Number of decisions exercised / Total number of decisions) x 100% In this case, the number of statements executed by the test cases is 8 (edges 1, 2, 3, 4, 5, 6, 7, and 9). The total number of statements in the flow graph is 10 (edges 1 to 10). Therefore, the statement coverage achieved by the test cases is:

* Statement coverage = (8 / 10) x 100% = 80%

The number of decisions exercised by the test cases is also 8 (nodes A, B, C, D, E, F, G, and H). The total number of decisions in the flow graph is also 10 (nodes A to J). Therefore, the decision coverage achieved by the test cases is:

* Decision coverage = (8 / 10) x 100% = 80%

Therefore, statement coverage and decision coverage are both less than 100%. To achieve 100% statement coverage and decision coverage, two more test cases are needed to cover edges 8 and 10 and nodes I and J.

**Q191.** What factors should be considered to determine whether enough testing has been performed?

(i)The exit criteria.

(ii)The budget.

(iii)How big the test team is.

(iv)The product’s risk profile.

(v)How good the testing tools are.

(vi)Sufficient details of the system status to allow decisions
* i and ii and iv and vi
* i and ii and iii and vi
* ii and iii and iv and v
* i and ii and v and vi

Explanation

These are some of the factors that should be considered to determine whether enough testing has been performed. The exit criteria define the specific goals and requirements for testing completion. The budget limits the resources and time available for testing. The product’s risk profile indicates the areas of highest priority and impact for testing. Sufficient details of the system status allow decisions on whether further testing is needed or not. The other options are not relevant factors for determining enough testing.

**Q192.** Load testing tool checks for:
* presence of bugs in user interface
* the number of testers needed in order to achieve “dead line”
* correct functional behavior of the system under test
* time response and resource utilization

**Q193.** Which of the following characteristics is most likely to promote effective software testing? [K1]
* Independence from the production process
* A belief that programmers always make mistakes
* Knowledge of the number of defects typically found in a program
* Confidence that the next stage will find defects missed at this stage

**Q194.** Which of the following BEST defines static techniques? [K1]
* Executing the software work product
* Manually examining the code or project documentation
* Automated analysis of the code or project documentation
* Manual examination and automated analysis of code or project documentation

Explanation

The statement that best defines static techniques is D. Manual examination and automated analysis of code or project documentation. Static techniques are techniques that analyze the code or other software artifacts (such as requirements, specifications, designs, models, test cases, etc.) without executing them, and provide information about their quality, defects, complexity, maintainability, etc. Static techniques can be performed manually or automatically by using tools or methods such as reviews, inspections, walkthroughs, checklists, standards, guidelines, static analysis tools, code metrics tools, etc. Static techniques can help to improve the quality and consistency of code or other software artifacts throughout the software development life cycle and reduce the cost and effort of testing and debugging. A detailed explanation of static techniques can be found in

[A Study Guide to the ISTQB Foundation Level 2018 Syllabus], pages 27-34.

**Q195.** Which of the following are part of ISTQB code of ethics?

I. Certified software testers shall advance the integrity and reputation of the profession consistent with the public interest II. Certified software tester shall always sign a NDA (Non Disclosure Agreement) in presence of customer data III. Certified software testers shall maintain integrity and independence in their professional judgment IV Certified software testers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest
* I, II, III
* II, III, IV
* I, II, V
* I, III, IV

**Q196.** Decision table testing is being performed on transactions in a bank’s ATM (Automated teller Machine) system. Two test cases have already been generated for rules 1 and 4, which are shown below:

| | RULES | R1 | R4 |
|---|---|---|---|
| Conditions | Account has been entered | T | F |
| | Amount has been entered | T | F |
| Actions | Process the transaction | T | F |
| | Generate an error message | F | T |

Given the following additional test cases:

| | Test Case | DT1 | DT2 | DT3 | DT4 |
|---|---|---|---|---|---|
| Inputs | Account | Checking | Not entered | Not entered | Savings |
| | Amount | $500 | Not entered | $100 | Not entered |
| Expected Result | Process the transaction | T | F | F | F |
| | Generate an error message | F | T | T | T |

Which two of the additional test cases would achieve fill coverage of the full decision table (when combined with the test cases that have already been generated for rules a and 4?
* DT1, DT2
* DT2, DT3

* DT3, DT4
* DT1, DT4

Explanation

The correct answer is D, as DT1 and DT4 would achieve full coverage of the full decision table when combined with the test cases that have already been generated for rules 1 and 4. Decision table testing is a black-box test technique that uses a table to show combinations of inputs and their associated outputs1. In this case, the inputs are the account type, the amount entered, and the expected result. The outputs are the actions taken by the ATM system. The decision table has eight rules, each corresponding to a possible combination of inputs and outputs. The test cases that have already been generated for rules 1 and 4 are:

TC1: Account = Checking, Amount = $600, Expected Result = Process the transaction TC4: Account = Savings, Amount = Not entered, Expected Result = Generate an error message The additional test cases are:

DT1: Account = Checking, Amount = Not entered, Expected Result = Generate an error message DT2: Account = Savings, Amount = $100, Expected Result = Process the transaction DT3: Account = Checking, Amount = $100, Expected Result = Process the transaction DT4: Account = Savings, Amount = $600, Expected Result = Generate an error message To achieve full coverage of the full decision table, two more test cases are needed that cover the remaining six rules. DT1 and DT4 cover these rules, as shown in the table below:

| Rule | Account | Amount | Expected Result | Test Case |
|---|---|---|---|---|
| 1 | Checking | $600 | Process the transaction | TC1 |
| 2 | Checking | Not entered | Generate an error message | DT1 |
| 3 | Checking | $100 | Process the transaction | DT3 |
| 4 | Savings | Not entered | Generate an error message | TC4 |
| 5 | Savings | $100 | Process the transaction | DT2 |
| 6 | Savings | $600 | Generate an error message | DT4 |
| 7 | Not entered | $600 | Generate an error message | DT1 |
| 8 | Not entered | Not entered | Generate an error message | DT1 |



Therefore, option D is the correct answer. Options A, B, and C are incorrect, as they do not cover all the rules in the decision table. References: 1, Section 4.2.5

**Q197.** Which of the following correctly states a limitation in the use of static analysis tools? [K1]

* Static analysis tools can be applied to new code but cannot be applied to existing code
* Static analysis tools can be used to enforce coding standards
* Static analysis tools always generate large numbers of warning messages when applied to new code, even if the code meets coding standards

* Static analysis tools do not generate warning messages when applied to existing code

A limitation in the use of static analysis tools is C. Static analysis tools always generate large numbers of warning messages when applied to new code, even if the code meets coding standards. Static analysis tools are tools that analyze the code or other software artifacts without executing them, and report any defects, errors, vulnerabilities, or violations of coding standards or conventions. Static analysis tools can help to improve the quality and maintainability of the code or other software artifacts. However, static analysis tools also have some limitations, such as generating large numbers of warning messages when applied to new code, even if the code meets coding standards or conventions. This can make it difficult for developers or testers to identify and prioritize the most important or critical issues to fix or address. A detailed explanation of static analysis tools can be found in A Study Guide to the ISTQB Foundation Level 2018 Syllabus, pages 109-1101.

**Q198.** Which of the following software development models BEST exemplifies a model that does NOT support the principle of early testing?

* The Waterfall model
* The V-model
* The Incremental development model
* The Iterative development model

Explanation

According to the syllabus, a software development life cycle (SDLC) model is a conceptual framework that describes the phases and activities involved in developing a software product. Different SDLC models have different advantages and disadvantages depending on the project context and objectives. The principle of early testing states that testing activities should start as early as possible in the software development life cycle and be focused on defined objectives. Early testing helps to prevent defects, reduce rework, lower costs, and improve quality. The answer A is correct because it best exemplifies a model that does not support the principle of early testing. The waterfall model is a sequential SDLC model that divides the development process into distinct phases, such as requirements analysis, design, implementation, testing, and maintenance.

Each phase must be completed before the next phase can begin, and there is no overlap or iteration between phases. Testing is done only after the implementation phase, which means that defects are detected late in the development cycle and are more expensive and difficult to fix. The other answers are incorrect because they exemplify models that support the principle of early testing. The V-model is an extension of the waterfall model that emphasizes verification and validation activities at each phase of development. Testing is done in parallel with each corresponding development phase, which means that defects are detected early and feedback is provided to the developers. The incremental development model is an iterative SDLC model that divides the development process into smaller increments or iterations, each delivering a working software product or a subset of features. Testing is done at the end of each iteration, which means that defects are detected early and feedback is provided to the developers. The iterative development model is another iterative SDLC model that repeats the development process for each iteration, with each iteration producing an improved version of the software product or a subset of features. Testing is done throughout each iteration, which means that defects are detected early and feedback is provided to the developers.

References: Certified Tester Foundation Level Syllabus, Section 1.1.1, page 9-10.

**Q199.** Which of the following lists represents the correct sequence of the main activities of the fundamental test

process (leaving out the activity of control which should take place in parallel to all the other activities)?

* Planning, analysis and reporting, design and implementation, execution, test closure activities,

evaluating exit criteria.

* Planning, analysis, design and implementation, execution, logging, test closure activities, evaluating exit

criteria.

* Planning, analysis and design, execution, logging and reporting, regression testing

* Planning, analysis and design, implementation and execution, evaluation exit criteria and reporting, test

closure activities

**Q200.** Which of the following statements is the best explanation why software failures can be caused by environmental conditions?
* Factors like magnetism, radiation and even pollution can affect electronic devices and the performance of their embedded real time software
* Environmental conditions only affect the hardware – not the software
* If the hardware on which the software application is running under ambient temperature and humidity, no failures can be linked to environmental conditions
* Extreme heat and vibrations exerted on storage media can cause errors in algorithms and program flows

The statement that environmental conditions only affect the hardware – not the software is not a good explanation why software failures can be caused by environmental conditions. Environmental conditions can affect both the hardware and the software, as they can influence the performance, reliability, and functionality of the system. For example, factors like magnetism, radiation, pollution, heat, humidity, vibration, etc. can affect electronic devices and the embedded software that runs on them. Software failures caused by environmental conditions can have serious consequences, especially for safety-critical systems.

References: Certified Tester Foundation Level Syllabus, Section 1.2.2

**Q201.** Which of the following BEST describes a test summary report for executive-level employees?
* The report Is detailed and Includes specific information on defects and trends
* The report is detailed and includes a status summary of defects by priority or budget
* The report is high-level and includes specific information on defects and trends
* The report is high-level and includes a status summary of detects by priority or budget

The correct answer is D, as it describes a test summary report for executive-level employees. A test summary report is a document that summarizes the results and evaluation of testing activities for a specific activity or phase3. It may have different levels of detail and content depending on the intended audience and purpose3. A test summary report for executive-level employees is typically high-level and includes a status summary of defects by priority or budget3. This type of report provides a concise overview of the quality and progress of testing without going into too much detail or technical information3. Option A is incorrect, as it describes a test summary report for technical-level employees. A test summary report for technical-level employees is typically detailed and includes specific information on defects and trends3. This type of report provides a comprehensive analysis of the quality and progress of testing with relevant data and metrics3. Option B is incorrect, as it describes a test summary report that is neither suitable for executive-level nor technical-level employees. A test summary report that is detailed and includes a status summary of defects by priority or budget is too detailed for executive-level employees and too vague for technical-level employees3. Option C is incorrect, as it describes a test summary report that is neither suitable for executive-level nor technical-level employees. A test summary report that is high-level and includes specific information on defects and trends is too high-level for technical-level employees and too specific for executive-level employees3. References: 3, Section 2.7

**Q202.** What other details should be included in the following incident report when it is first submitted?

Date of Issue: 23/11/05

Severity: P1

Build: Version15.6

Details: Expected field to be limited to 15 chars, able to enter 27
* Suggested solution, priority and number of defects assigned to this developer.
* Status of the incident, degree of impact, Test Case Number.
* History, related defects and expected fix time.

* Line of code, number of defects found, time of day.

The other details that should be included in the incident report when it is first submitted are B. Status of the incident, degree of impact, Test Case Number. These details are important to provide information about the current state of the incident, the severity of its effect on the system or user, and the test case that detected the incident. Other details, such as suggested solution, priority, history, related defects, expected fix time, line of code, etc., can be added later during the incident management process. A detailed explanation of incident report can be found in A Study Guide to the ISTQB Foundation Level 2018 Syllabus, pages 99-1001.

**Q203.** Which of the following would NOT be a typical target of testing support tools?

* Automate activities that require significant resources when done manually
* Automate activities that cannot be executed manually
* Automate repetitive tasks
* Automating repetitive inspections

Testing support tools are software tools that assist testers in performing testing activities, such as test management, test design, test execution, test evaluation, and defect management1. According to the ISTQB syllabus, some typical targets of testing support tools are to automate activities that require significant resources when done manually, automate activities that cannot be executed manually, and automate repetitive tasks1. Automating repetitive inspections is not a typical target of testing support tools, as inspections are static testing techniques that involve human review and analysis of software artifacts1.

**Q204.** Which of the following defects-can

* Infinite loops
* Wrong business rules
* Syntax errors of the code
* Undefined variables

Explanation

Wrong business rules are defects that cannot be found by static analysis tools, because they are logical or functional errors that depend on the context and requirements of the software13. A static analysis tool can only detect syntactic or structural errors in the code or design of the software under test, but not whether they match the intended business rules or logic13. The other options are defects that can be found by static analysis tools. Option A is a defect that can be found by a static analysis tool, because it is a structural error that causes the code to loop indefinitely without terminating13. Option C is a defect that can be found by a static analysis tool, because it is a syntactic error that causes the code to fail to compile or run13. Option D is a defect that can be found by a static analysis tool, because it is a structural error that indicates a waste of memory or a possible logic flaw

**Q205.** Which of the following statements about use-case testing are most accurate?

(i)In a use-case diagram an actor represents a type of user.

(ii)Use-cases are the most common test basis for unit testing.

(iii)A use-case describes interactions between actors.

(iv)An actor is always a human user that interacts with the system.

(v)Test cases can be based on use-case scenarios.

(vi)Use-case testing will often identify gaps not found by testing individual components.

* ii, iii, iv, v
* i, iii, v, vi
* i, ii, iv, v
* iii, iv, v, vi

**Q206.** Which of the following options BEST explain the pesticide paradox principle of testing?
* If we do not regularly review and revise our tests, we’ll stop finding defects
* Repeatedly running a set of tests will ensure that a system is defect free
* Defects are, paradoxically, often contained in a small number of modules
* Testing, like spraying pesticide, is an effective bug / defect removal activity

The pesticide paradox principle of testing states that if we do not regularly review and revise our tests, we’ll stop finding defects, because the same tests will no longer be effective at detecting new or different defects in the software12. The other options do not explain the pesticide paradox principle of testing. Option B is false, because repeatedly running a set of tests will not ensure that a system is defect free, but rather that it meets the same quality criteria as before12. Option C is related to the defect clustering principle of testing, which states that defects are often contained in a small number of modules12. Option D is a metaphor that compares testing to spraying pesticide, but it does not explain the paradoxical effect of using the same tests over and over again