**Lesson 9 - Test design techniques overview. White box techniques**

Level 1

**Make a comparison of static and dynamic testing techniques. Give the advantages and**

**possible limitations when using each of them.**

|  | **Static Testing Technique** | **Dynamic Testing Technique** |
| --- | --- | --- |
| **Description** | Static testing techniques are those where the code is not executed | Dynamic testing involves running the software and executing test cases to evaluate its behavior, performance, and functionality. |
| **Advantage 1** | Early Detection of Issues: Static testing allows for the early identification of defects in the software development process. | Detects Runtime Errors: Dynamic testing can identify runtime errors, such as logic errors, crashes, and performance issues, which are impossible to find through static analysis alone. |
| **Advantage 2** | Improves Code Quality: By reviewing code and design documents, static testing helps improve the quality and maintainability of the codebase. It can lead to better coding standards and practices. | Verifies Functionality: It ensures that the software functions as expected under various conditions, helping to validate that it meets the specified requirements. |
| **Advantage 3** | No Test Case Preparation: static testing doesn't require creating test cases, making it less time-consuming and more cost-effective. | Automation Potential: Many dynamic testing activities, like unit testing and automated test scripts, can be automated, allowing for repeatable and efficient testing. |
| **Limitation 1** | Limited Coverage: Static testing cannot uncover runtime errors, integration issues, or performance problems since it doesn't involve code execution. | Cost and Time-Intensive: Dynamic testing typically consumes more resources, including time and human effort, especially when running a wide range of test cases. It's often conducted later in the development cycle, leading to higher defect resolution costs. |
| **Limitation 2** | Subjective Nature: The effectiveness of static testing heavily depends on the skills and experience of the individuals involved in the review, which can introduce subjectivity into the process. | Incomplete Test Coverage: Achieving complete test coverage can be challenging, and it's possible to miss specific scenarios, leading to untested areas of the software. |
| **Limitation 3** | Doesn't Ensure Functionality: Static testing may not guarantee that the software functions correctly; it primarily focuses on finding defects in the code and design. | Late Issue Detection: Dynamic testing detects issues after the code has been developed, which can be costly and require significant rework. |
| **Conclusion** | Static testing is valuable for early defect detection, improved code quality, and cost savings. | Dynamic testing is essential for verifying functionality and finding runtime errors. |

Level 2

**1. The following statement is about decision coverage:**

**When the code has a single 'IF' condition and no loops (LOOP) or switches (CASE), any test we run will result in 50% decision coverage.**

**Which option is true about this statement?**

**a. Correct. Any test case provides 100% coverage of statements, thus**

**covering 50% of solutions.**

**b. Correct. The result of any IF condition test will be either true or false.**

**c. Incorrect. A single test case can guarantee 25% coverage of the solutions**

**in this case.**

**d. Incorrect because it is too general a statement. We cannot know if it is**

**correct as it depends on the software being tested.**

**Answer:**

**b. Correct. The result of any IF condition test will be either true or false.**

**2. There’s the following pseudocode: Switch PC on -> Start MS Word -> IF MS Word**

**starts THEN -> Write a poem -> Close MS Word.**

**How many test cases will it take to test its functionality?**

**a. 1 for operator coverage, 2 for decision coverage**

**b. 1 for operator coverage, 1 for decision coverage**

**c. 2 for operator coverage, 2 for decision coverage**

**d. 2 for operator coverage, 1 for decision coverage**

**Answer:**

**c. 2 for operator coverage, 2 for decision coverage**

**a. 1 for operator coverage, 2 for decision coverage**

**3. How many tests are needed to check code statements:**

**Read P**

**Read Q**

**If P+Q>100 THEN**

**Print “Large”**

**ENDIF**

**If P>50 THEN**

**Print “P Large”**

**ENDIF**

**Answer: 1 Test Case**

Level 3

We continue working on a startup for a cat photo-sharing app.

There’s the following algorithm:

Ask what kind of pet the user has.

If the user answers that they have a cat, then ask what breed it is: "short-haired or

long-haired?"

If the user answers "long-haired," then ask: "Would you like to get the contacts of the

nearest groomer?"

If the user answers "yes," then say: "Give me the address of the nearest cat grooming

salon."

else

Say: "Suggest a shop with fur care products."

end

else

Say, "Suggest a pet shop."

end

If the user has no cat

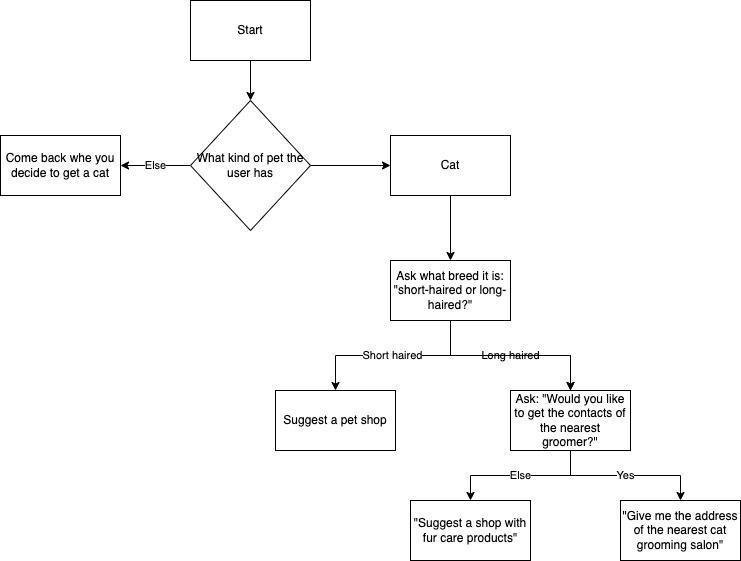
Say, "Come back when you decide to get a cat."

End

Assignment:

1. Draw an algorithm diagram (in a tool of your choice, for example, in the built-in

Google Docs editor, Figjam, or other.)



**2.** **What is the minimum set of test cases needed to make sure that all questions**

**have been asked, all combinations have been passed, and all answers have been**

**obtained?**

Possible paths:

The user has a cat, the breed is long-haired, and they want the nearest groomer.

The user has a cat; the breed is long-haired, but they don't want the nearest groomer.

The user has a cat breed that is short-haired.

The user has no cat.

For this scenario, I would have a minimum of 4 test cases, as follows:

1. The user has a cat, the breed is long-haired, and they want the nearest groomer (Covering Path 1).
2. The user has a cat; the breed is long-haired, but they don't want the nearest groomer (Covering Path 2).
3. The user has a cat, and the breed is short-haired (Covering Path 3).
4. The user has no cat (Covering Path 4).