SET 1

1. Refactoring is not one of the stages of TDD.
   o True
   o False

2. TDD required a developer to write the test cases before writing the actual production code.
   o True
   o False

3. TDD is basically unit testing.
   o True
   o False

4. TDD can help a developer improve his initial design of a method by forcing him to come up with situations in which the test case can fail
   o True
   o False

5. TDD does not help in detection of bugs at an early developmental stages of SDLC.
   o True
   o False

6. There are _Two_____ types of TDD.

7. TDD has been discovered or reinvented by __________

8. TDD involves automating of test cases according to the need of the user

o True

o False

9.Give an example of a situation where TDD will not work.(Hint: Think of situations where entire production code is required before starting the testing phase)

One situation where Test-Driven Development (TDD) might not work well is when the development requires a significant amount of upfront planning and design before any code can be written or tested. Here's an example:

Suppose you are developing a complex system with intricate interactions between various components, and the requirements are not clear-cut. In such a scenario, it may be challenging to write meaningful tests upfront because the system's behaviour and interactions are not well understood yet. Additionally, implementing the entire system or significant portions of it before writing tests may be necessary to gain insights into the system's behaviour and requirements.

For instance, if you're building a highly specialized algorithm or a novel machine learning model where the behaviour cannot be easily defined beforehand, it might be more practical to implement the core functionality first and then write tests to verify its correctness and performance.

In cases like these, a more iterative approach to development, where code and tests are written incrementally based on emerging understanding and requirements, might be more suitable than strictly adhering to the TDD methodology.

10. ATDD is also called __ Acceptance Test-Driven Development. ________

SET 2

1.What does BDD stand for

- Behaviour Driven Deployment
- Behaviour Driven Development
- Basic Development and Deployment
- Behaviour Driven Discussion

2.BDD is about the collaboration between teams

- True
- False

3.What are the 3 Practices of BDD

- Discovery, Integration, Development
- Discovery, Formulation, Development
- Discovery, Formulation, Automation

4.What is the sequence of the 3 Practices of BDD

- Discovery, Automation, Formulation
- Discovery, Formulation, Automation
- Formulation, Discovery, Automation

5.What is the outcome of the Discovery phase .Documented Examples

The outcome of the Discovery phase in BDD is documented examples. These examples, often written in a natural language format using tools like Gherkin syntax, serve as concrete illustrations of the desired behaviour of the software system. They help stakeholders, including developers, testers, and business representatives, to gain a shared understanding of the requirements and expectations for the features being developed. These examples form the basis for further development and automation in the subsequent phases of BDD.

6.What is the outcome of the Formulation phase

- Requirement Document
- Documented Examples of system behaviour

7.What is the outcome of the Automation phase

- Implemented Code + Automation Tests
- Written Documentation of Examples

8.What is Cucumber

- A tool to facilitate BDD
- Helps in converting documented examples to automation tests
- All of the above

9.Using Cucumber means you are doing BDD

- False
- True

10.If you want to implement BDD, you have to use Cucumber

- No
- Yes

SET 3

1.The process of BDD starts with

- Discovery
- Formulation
- Automation

2.Why we call the Discovery meeting as the **"3 Amigos meeting"**

- Only 3 people should attend this meeting
- Minimum 3 people should attend

• To have a different perspective from at least 3 different teams to discuss and create a common understanding

3.Who are the “3 Amigos”

• Top 3 developers of the team

• Top 3 testers of the team

• Product Owner, Developer & Tester

4.What is a USER STORY

• A story about a user

• General description of the software feature or change from an end-user perspective

5.Who writes a USER STORY

• Product Owner

• Software Engineer

• Test Engineer

6.What is the purpose of the Discovery meetings

• To explain the user story and explore and discuss the system behaviour

• To sit together and watch the Discovery channel

7.In Discovery Meetings User Stories can be updated based on new rules or features discovered during the meeting

• Yes

• No

8.What is the sequence of the 3 Practices of BDD

- Discovery, Automation, Formulation
- Discovery, Formulation, Automation
- Formulation, Discovery, Automation

9.If you are not doing Discovery meetings, you are not doing BDD

- True
- False

10.What is the outcome of the Discovery phase

- Common agreed understanding of the behaviour of the system
- Documented Examples

SET 4

1 . ___________ is an iterative development methodology, where both development and testing activities are concurrent.

A)Agile

B)Apache

C)Espresso

D)Bugzilla

2 . Agile Testing covers all the levels of testing and all types of testing.

A)True

B)False

C)Can not say

D)—

3 . ___________ is an Agile development method that emphasizes on team-centric approach.

A)Xp

B)Scrum

C)FDD

D)Crystal

4 . What are advantage's of Agile testing?

A)Saves time

B)Regular feedback from end users

C)requires less planning and creates less documentation

D)All of the above

5 . FDD stands for?

A)Feature Driven Design

B)Feature Driven Difference

C)Feature Driven Development

D)Feature Design Development

6 . Which testing is based on the expected behaviour of the software being developed?

A)FDD

B)TDD

C)BDD

D)ATDD

7 . The product backlog is prioritized by__________

A)Scrum master

B)Scrum team

C)product owner

D)All of the above

8 . What is the Agile?

A)Iterative

B)Sequential

C)Incremental

D)Both (A) and (C)

9 . How do tools like FIT and Fitness compliment Agile?

A)Unit Test

B)Automated Acceptance Tests

C)Code coverage metrics

D)Continuous Integration

10 . In Agile Testing Practices, which of them has two team members work together at the same keyboard?

A)Pairing

B)Mind Mapping

C)Incremental Test Design

D)None of the above

SET 5

1) White box texting is also known as Structural testing__
2) White box texting is dynamic ,static, or infeasible – Dynamic testing
3) A _Negative_____ test is when an invalid input is put and error are received
4) __Automation______ tool is involved in automation of regression test

5) _________ test cases are when random test cases are performed and equivalence partioning is applied to those test cases

- Random
- Static
- Semi-random
- Equivalent random

6) ___________ is normally used to evaluate a product to determine its suitability for intended use and to identify discrepencies

- Supervisor
- Technical
- Decision
- Design

7) ________ insures the original fault has been removed

- Regression testing
- Re-testing

• Negative testing

8) Static analysis cannot

• Enforce coding

• Detect memory leaks

• Help in code understanding

SET 6

1. Not a Behaviour-Driven Development tools
   - o Concordion
   - o JBehave
   - o SpecFlow
   - o Concordat

2. Check if all the steps have the step definition before execute
   - o DryRun
   - o Monochrome
   - o Strict
   - o Format

3. Which is not a Cucumber Report
   - o Json
   - o Progress
   - o Html
   - o Rerun
   - o None of the above

4. ________________is the actual code implementation of the feature mentioned in feature file.
   - o Feature File
   - o Step definition
   - o Test Runner

o None of the above

5. What does Cucumber do?
   o Cucumber tests web page safety o Cucumber tests other software
   o Cucumber test proxy status
   o Cucumber tests software for phishing malware

6. In which programming language was Cucumber originally written?
   o Mercury
   o Selenium
   o Diamond
   o Ruby

7. What language does Cucumber use?
   o Veggie
   o Salad
   o Gherkin
   o Ginger

8. What does Gherkin do?
   o Checks reliability of computer codes
   o Translates computer codes for people to understand them
   o Writes code in a computer-readable way
   o Provides simple documentation of the code under test

9. Which sign is used in Gherkin's syntax?
   o #
   o $
   o %
   o &

10. Which file extension do all Gherkin files have?
   o .closure
   o .feature
   o .doc
   o com

11. What is Cucumber?
   o A tool
   o A plugin
   o A tab
   o A vegetable?

12. _____________is a process of developing software based on behavioral specification of software units.
   o BDD
   o TDD
   o ATDD
   o None of the above

13. ____________is a software development technique where automated tests are written before the code.

o TDD

o BDD

o ATDD

o Hooks

14. This person is most concerned with the scope of the application. This involves translating user stories into a series of features.
   o Product Owner
   o Tester o Developer
   o Project Manager

15. What's the subdivision of Gherkin's features?
   o Steps
   o Pages
   o Scenarios
   o Books

16. How many parts does a Gherkin feature have?
   o 3
   o 5

o 2
o 10

17. Which one of these Formatter Plugins Cucumber doesn't use to provide output?
o Www
o JSON
o HTML
o JUnit

18. Which languages does Cucumber not support?
o C
o Ruby
o Net
o Python

19. What software is needed to run a Cucumber Web Test?
o Ruby and its Development Kit
o Cucumber
o IDE like Active State
o All of the above

20. What are the advantages of Cucumber?
o Cucumber acceptance tests are written from user's the perspective.
o Team Collaboration (Product owners, business analysts, developers, testers are all involved in writing scenarios)
o Tests are written in plain-text English, so people with less technical skills can also write scenarios.
o All of the above

21)Mention at least two differences between TDD and BDD

Two key differences between Test-Driven Development (TDD) and Behaviour-Driven Development (BDD) are:

Focus on Stakeholders:

TDD primarily focuses on developers writing tests to drive the implementation of individual units of code. It emphasizes the technical aspects of testing and is more developer-centric.

BDD, on the other hand, involves collaboration among stakeholders, including developers, testers, product owners, and business representatives. It emphasizes writing tests in a natural language format that reflects the behaviour of the system from the perspective of end-users and stakeholders.

Language and Scope of Tests:

In TDD, tests are typically written in the programming language of the development environment and focus on testing the behaviour of individual methods or functions within the codebase.

In BDD, tests are written in a human-readable format, often using a Given-When-Then structure, known as Gherkin syntax. These tests describe the behaviour of the system from the perspective of users or stakeholders and focus on testing the system's behaviour as a whole, typically through end-to-end scenarios.

1. **write a simple string calculator using TDD principles.(CALCI)**
   - **The function should return 0 for an empty string.**
   - **Write the function to handle an empty string and return 0.**
   - **The function should add numbers separated by commas.**

1. Define the Calculator function: - Input: String containing numbers separated by commas - Output: Integer representing the sum of the numbers

2. Write a test for an empty string input: - Input: "" - Expected Output: 0

3. Implement the Calculator function to handle an empty string: - If input string is empty, return 0

4. Write a test for adding numbers separated by commas: - Input: "1,2,3" - Expected Output: 6 (1 + 2 + 3)

5. Implement the Calculator function to add numbers separated by commas: - Split the input string by commas - Convert each part to an integer - Sum the integers and return the result

6. Write additional tests to cover edge cases and different input scenarios: - Input: "5" Expected Output: 5 - Input: "10,20,30,40" Expected Output: 100

7. Implement error handling for invalid input (optional): - If input contains non-numeric characters or invalid separators, return an error message or raise an exception

2. write a password validator function using TDD:(VALIDATOR)

• The function should return "Password cannot be empty" for an empty string

• (Initial): Define the function with a placeholder message

• The function should return "Password must be at least 8 characters long" for a password less than 8 characters.

• Update the function to check password length and return the appropriate message if it's less than 8 characters

1. Define the PasswordValidator function: - Input: String containing the password - Output: String representing the validation message

2. Write a test for an empty string input: - Input: "" - Expected Output: "Password cannot be empty" 3. Implement the PasswordValidator function to handle an empty string: - If input string is empty, return "Password cannot be empty"

4. Write a test for a password less than 8 characters: - Input: "1234567" - Expected Output: "Password must be at least 8 characters long"

5. Implement the PasswordValidator function to check password length: - If length of input string is less than 8 characters, return "Password must be at least 8 characters long"

6. Write additional tests to cover edge cases and different input scenarios: - Input: "password" Expected Output: "Password is valid" - Input: "superstrongpassword" Expected Output: "Password is valid"

7. Implement additional validation rules as needed: - Check for special characters, uppercase letters, etc. - Return appropriate messages for different validation criteria.

8. Refactor the code to improve readability, efficiency, and maintainability while ensuring that tests continue to pass