CASE STUDY

Software Testing and Quality Assurance

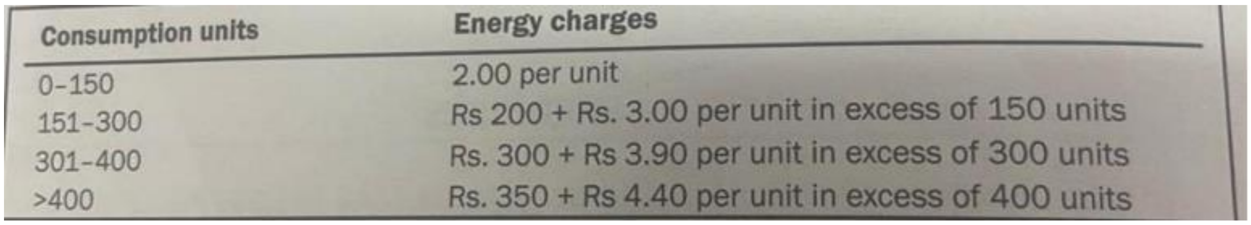
Abstract
Identify the best suitable black box testing technique and also frame the code for the mentioned problem statement and find the best suitable white box testing technique.

Syeda Reeha Quasar
14114802719

Case Study

Aim:

The BSE Electrical Company charges its domestic consumers using the following slab:

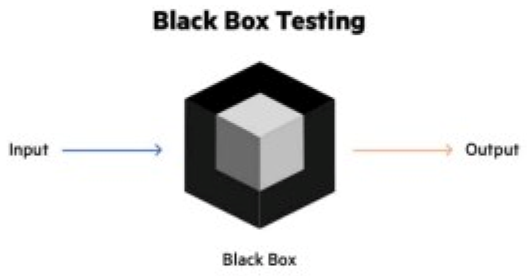


Consumption units	Energy charges
0-150	2.00 per unit
151-300	Rs 200 + Rs. 3.00 per unit in excess of 150 units
301-400	Rs. 300 + Rs 3.90 per unit in excess of 300 units
>400	Rs. 350 + Rs 4.40 per unit in excess of 400 units

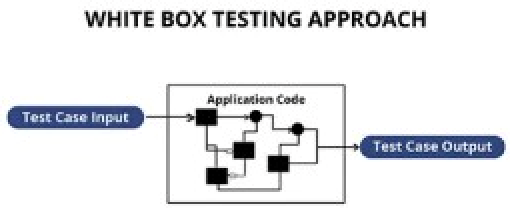
Identify the best suitable black box testing technique and also frame the code for the abovementioned problem statement and find the best suitable white box testing technique.

Theory:

Black Box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. Only the external design and structure are tested.



White Box Testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. Implementation and impact of the code are tested.



Black Box Testing	White Box Testing

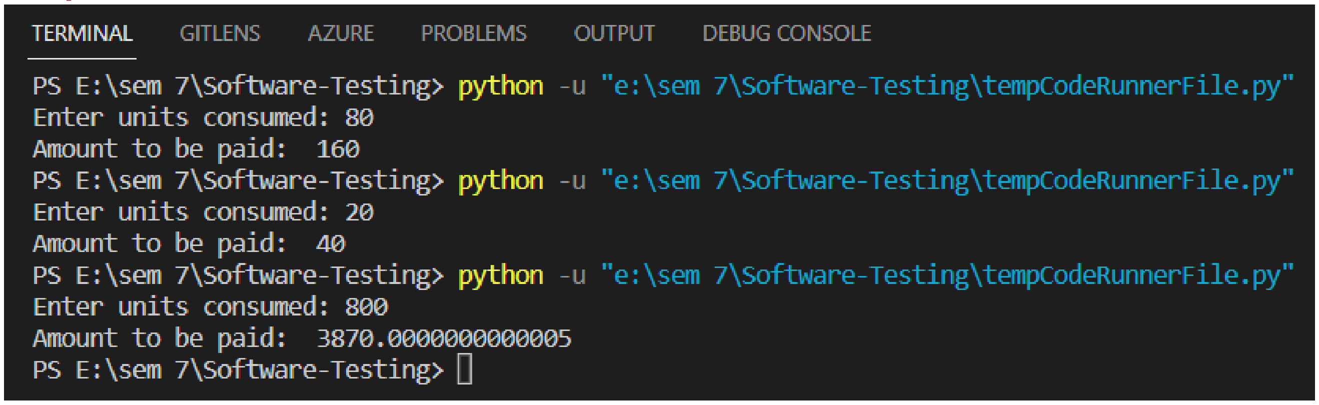
The Black Box Test is a test that only considers the external behavior of the system; the internal workings of the software is not taken into account.	The White Box Test is a method used to test a software taking into consideration its internal functioning.
It is carried out by testers.	It is carried out by software developers.
This method is used in System Testing or Acceptance Testing.	This method is used in Unit Testing or Integration Testing.
It is the least time consuming.	It is most time consuming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is also known as data-driven testing, functional testing, and closed box testing.	It is also known as clear box testing, code-based testing, structural testing, and transparent testing.
Black Box Test is not considered for algorithm testing.	White Box Test is well suitable for algorithm testing.

Source Code:

```python
cu = int(input("Enter units consumed: "))
amt = 0.0
if cu <= 150 and cu > 0:
    amt = cu*2
elif cu >= 151 and cu <= 300:
    amt = 200 + 3 * cu
elif cu >= 301 and cu <= 400:
    amt = 300 + 3.90 * cu
elif cu > 400:
    amt = 350 + 4.40 * cu

print("Amount to be paid: ", amt)
```

Output:



```
PS E:\sem 7\Software-Testing> python -u "e:\sem 7\Software-Testing\tempCodeRunnerFile.py"
Enter units consumed: 80
Amount to be paid:  160
PS E:\sem 7\Software-Testing> python -u "e:\sem 7\Software-Testing\tempCodeRunnerFile.py"
Enter units consumed: 20
Amount to be paid:  40
PS E:\sem 7\Software-Testing> python -u "e:\sem 7\Software-Testing\tempCodeRunnerFile.py"
Enter units consumed: 800
Amount to be paid:  3870.0000000000005
PS E:\sem 7\Software-Testing> 
```

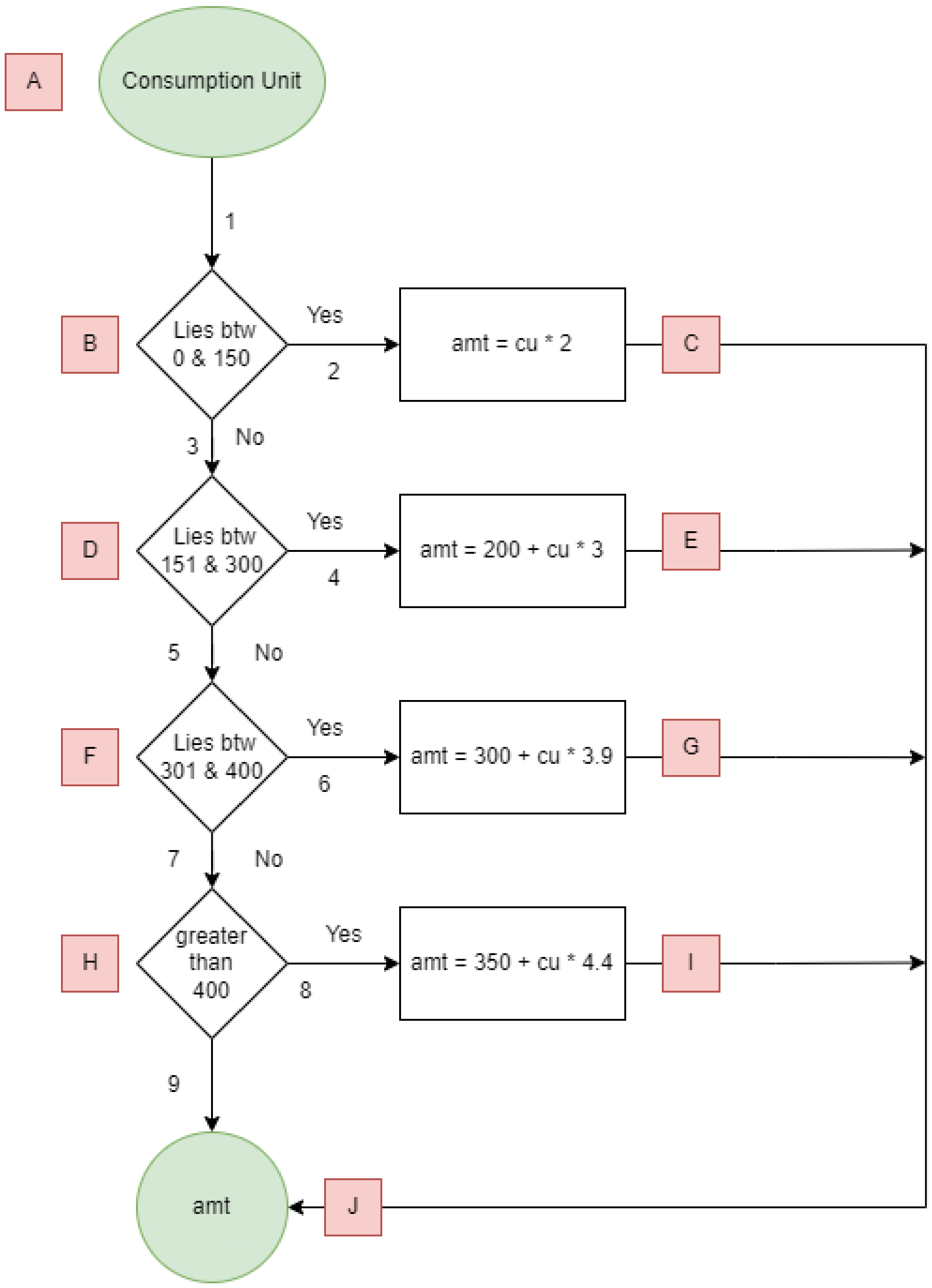
White Box Testing:

Branch analysis is a white box testing technique that involves examining the individual branches or paths of execution within a program or system and ensuring that they are executed correctly. In branch analysis, the tester creates test cases that exercise as many different branches as possible, with the goal of identifying any defects or errors that may occur when certain branches are taken.

To perform branch analysis, the tester typically needs to have detailed knowledge of the internal structure and implementation of the system being tested, including the source code and any logical or conditional statements it contains. The tester may use tools such as code coverage tools or static analysis tools to help identify and analyze the different branches within the system.

Branch analysis is often used in combination with other white box testing techniques, such as unit testing or integration testing, to ensure that the system is functioning correctly at the component level and that all possible paths of execution are tested. It can be an effective way to uncover defects that may not be detected by other testing techniques, but it can also be time-consuming and may require significant resources to perform thoroughly.

Case	Path	Branch coverage
C	A1-B2-CJ	3
E	A1-B3-D4-EJ	4
G	A1-B3-D5-F6-GJ	5
H	A1-B3-D5-F7-H8-IJ	6

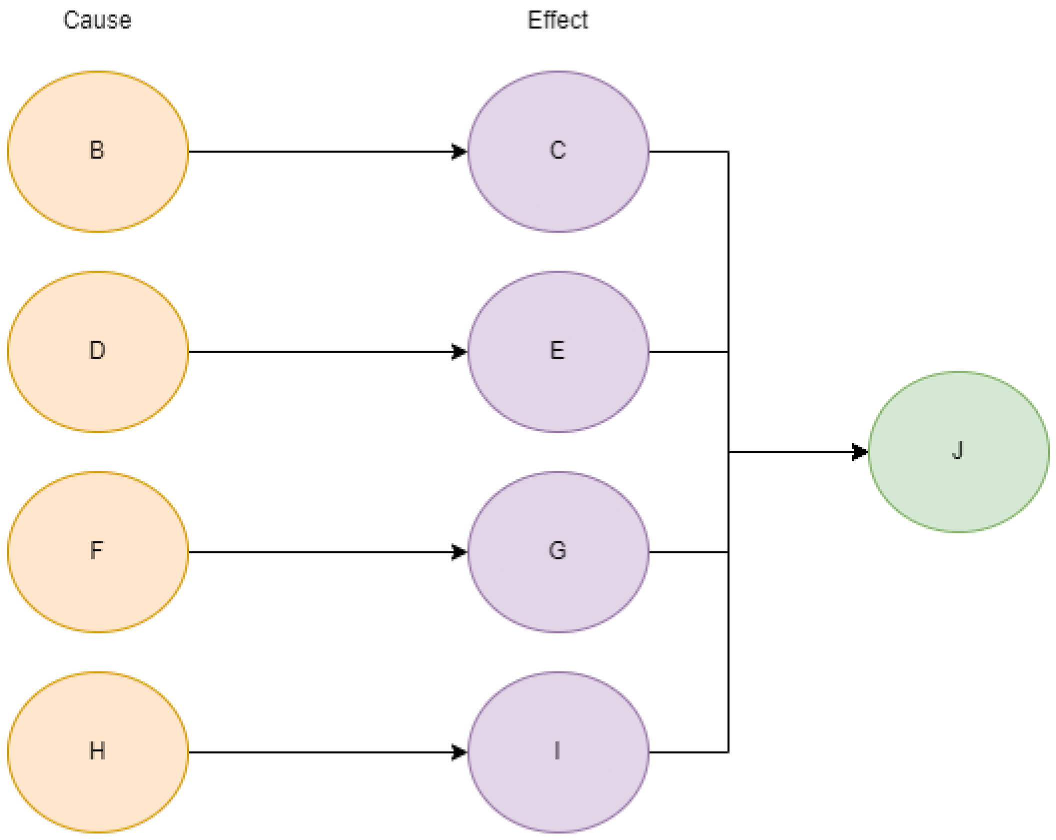


Black Box Testing:

Cause and effect testing is a black box testing technique that involves identifying the possible causes of a problem or defect within a system, and then creating test cases to determine whether those causes are indeed the root of the problem. This technique is based on the idea that every problem or defect has a specific cause, and that by identifying and testing the potential causes, it is possible to determine the true cause of the problem and fix it.

To perform cause and effect testing, the tester typically does not need to have detailed knowledge of the internal structure or implementation of the system being tested. Instead, the tester focuses on the input and output behavior of the system and the relationships between different inputs and outputs. The tester may use various tools and techniques, such as fault injection, boundary value analysis, or equivalence partitioning, to identify and test the possible causes of the problem.

Cause and effect testing can be an effective way to quickly narrow down the possible causes of a problem and identify the root cause, but it may not be suitable for all types of problems or defects. It is important to carefully consider the goals and constraints of the testing process and choose the appropriate testing techniques accordingly.



Result:
The case was successfully studied, coded, executed and tested.

Viva Questions

Q1. Why choose Open Source Performance Test tool?
It is essential because it helps to test the following:

- It is executed at different levels such as system, integration, and unit level of software development.
- One primary goal of it is to verify the working of an application.
- It involves the identification of the operational flow of an application.

Q2 What exactly ad-hoc testing is?

Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

Q3. What are the draw backs of ad-hoc testing?

Following are some drawbacks:

1. Since ad-hoc testing is done without any planning and in unstructured way so recreation of bugs sometime becomes a big trouble.
2. The test scenarios executed during the ad-hoc testing are not documented so the tester has to keep all the scenarios in their mind which he/she might not be able to recollect in future.
3. Ad-hoc testing is very much dependent on the skilled tester who has thorough knowledge of the product it cannot be done by any new joiner of the team.

Q4. At what situation it is prompt to implement ad-hoc testing?

Ad hoc testing can be performed when there is limited time to do elaborative testing. Usually Adhoc testing is performed after the formal test execution. And if time permits, ad hoc testing can be done on the system. Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

Q5: What are the advantages of ad-hoc testing?

1. Ad-hoc testing gives freedom to the tester to apply their own new ways of testing the application which helps them to find out more number of defects compared to the formal testing process.
2. This type of testing can be done at anytime anywhere in the Software Development Life cycle (SDLC) without following any formal process.
3. This type of testing is not only limited to the testing team but this can be done by the developer while developing their module which helps them to code in a better way.