**PRACTICAL NO 5**

**Description: Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.

**AIM: To study of Black Box Testing in software testing.**

**Theory:**

**Black Box Testing**

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

**Types of Black Box Testing**

There are many types of Black Box Testing but the following are the prominent ones:

* **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.
* **Non-functional testing**- This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
* **Regression testing**- [Regression Testing](https://www.guru99.com/regression-testing.html) is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

**Black Box Testing Techniques**

Following are the prominent[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)amongst the many used in Black box Testing

* **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
* **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
* **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

**Equivalence Class Partitioning**

Equivalence Partitioning or Equivalence Class Partitioning is type of black box testing technique which can be applied to all levels of software testing like unit, integration, system, etc. In this technique, input data units are divided into equivalent partitions that can be used to derive test cases which reduces time required for testing because of small number of test cases.

* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

**Boundary Value Analysis**

For the most part, errors are observed in the extreme ends of the input values, so these extreme values like start/end or lower/upper values are called Boundary values and analysis of these Boundary values is called “Boundary Value Analysis”. It is also sometimes known as ‘Range Checking’. Boundary value analysis is another black box test design technique and it is used to find the errors at boundaries of input domain rather than finding those errors in the centre of input. Equivalence Partitioning and Boundary value analysis are linked to each other and can be used together at all levels of testing. Based on the edges of the equivalence classes, test cases can then be derived. Each boundary has a valid boundary value and an invalid boundary value. Test cases are designed based on the both valid and invalid boundary values. Typically, we choose one test case from each boundary. Finding defects using Boundary value analysis test design technique is very effective and it can be used at all test levels. You can select multiple test cases from valid and invalid input domains based on your needs or previous experience but remember you do have to select at least one test case from each input domain.

**Cause Effect Graphing and Decision table technique**

**Cause Effect Graphing**

Cause-effect graph comes under the black box testing technique which underlines the relationship between a given result and all the factors affecting the result. It is used to write dynamic test cases. The dynamic test cases are used when code works dynamically based on user input. For example, while using email account, on entering valid email, the system accepts it but, when you enter invalid email, it throws an error message. In this technique, the input conditions are assigned with causes and the result of these input conditions with effects. Cause-Effect graph technique is based on a collection of requirements and used to determine minimum possible test cases which can cover a maximum test area of the software. The main advantage of cause-effect graph testing is, it reduces the time of test execution and cost. This technique aims to reduce the number of test cases but still covers all necessary test cases with maximum coverage to achieve the desired application quality. Cause-Effect graph technique converts the requirements specification into a logical relationship between the input and output conditions by using logical operators like AND, OR and NOT.

**Notations used in the Cause-Effect Graph**

* **AND  
  OR**
* **NOT  
  MUTUALLY EXCLUSIVE**

**Decision table technique**

Decision table technique is one of the widely used case design techniques for black box testing. This is a systematic approach where various input combinations and their respective system behaviour are captured in a tabular form. That's why it is also known as a cause-effect table. This technique is used to pick the test cases in a systematic manner; it saves the testing time and gives good coverage to the testing area of the software application. Decision table technique is appropriate for the functions that have a logical relationship between two and more than two inputs. This technique is related to the correct combination of inputs and determines the result of various combinations of input. To design the test cases by decision table technique, we need to consider conditions as input and actions as output.

**Use case testing**

The use case is functional testing of the black box testing used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end. It is a graphic demonstration of business needs, which describe how the end-user will cooperate with the software or the application. The use cases provide us all the possible techniques of how the end-user uses the application.

**Advantage of Use Case Technique**

The use case technique gives us some features which help us to create an application.

Following are the benefits of using the use case technique while we are developing the product:

* The use case is used to take the functional needs of the system.
* These are the classification of steps, which describe the connections between the user and its actions.
* It starts from an elementary view where the system is created first and primarily used for its users.
* It is used to determine the complete analyses, which help us to achieve the complication, and then it focuses on the one detailed feature at a time.

**State Transition**

State Transition Testing is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test (AUT). State transition testing helps to analyse behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behaviour. It is the model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before, is a finite state system. There are 4 main components of the State Transition Model as below:

* **States** that the software might get
* **Transition** from one state to another
* **Events** that origin a transition like closing a file or withdrawing money
* **Actions** that result from a transition (an error message or being given the cash.)

**Solve the given problems**

**A) Equivalence Class Partitioning**

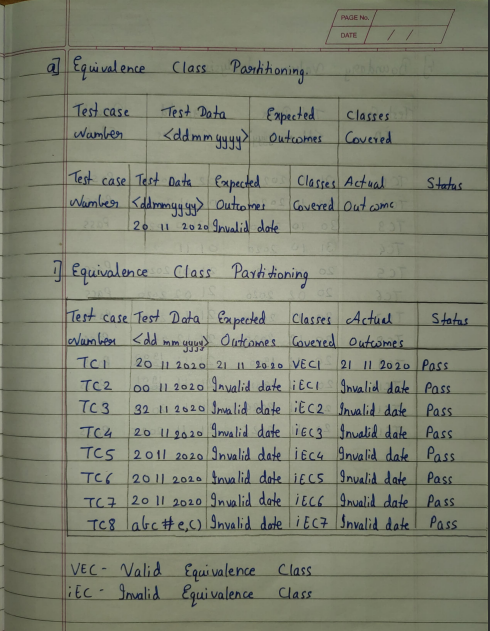
A program determines the next date in the calendar. Its input is entered in the form of <ddmmyyyy> with the following range:

1 ≤ mm ≤ 12

1 ≤ dd≤ 31

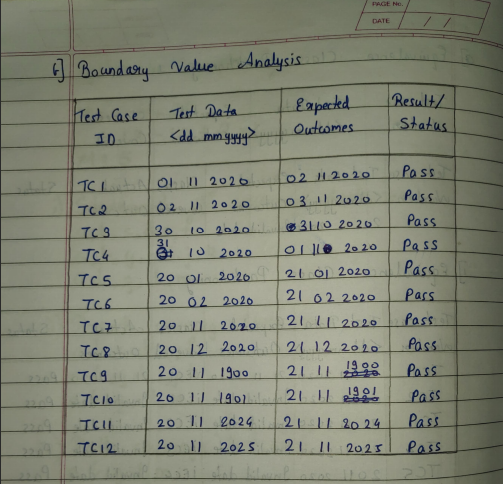
1900 ≤ yyyy ≤ 2025.

Its output would be the next date or an error message “invalid date”. Design test cases using equivalence class Partitioning method.



**B) Boundary Value Analysis:**

For the same problem statement given above, Design test cases using Boundary Value Analysis method.



**C) Cause Effect Graphing and Decision table technique:**

A university is admitting students in a professional course subject to the following conditions:

(a) Marks in Java >= 70

(b) Marks in C++ >= 60

(c) Marks in Data Structures >= 60

(d) Total in all three subjects >= 220 OR Total in Java and C++ >=150

If the aggregate marks of an eligible candidate are more than 240, he will be eligible for a scholarship course, otherwise he will be eligible for normal course.

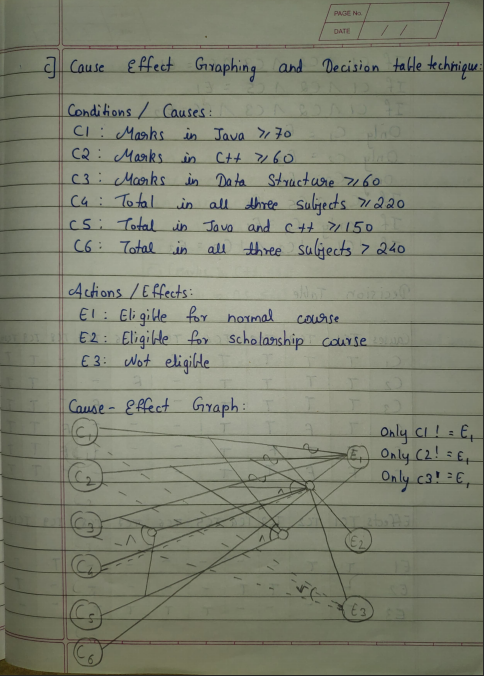
The program reads the marks in the three subjects and generates the following outputs:

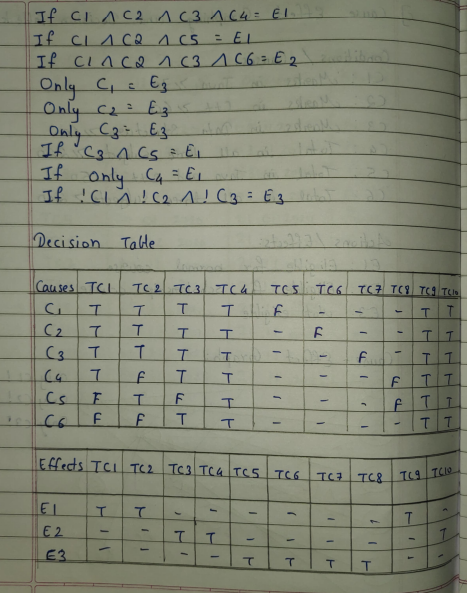
(i) Not Eligible

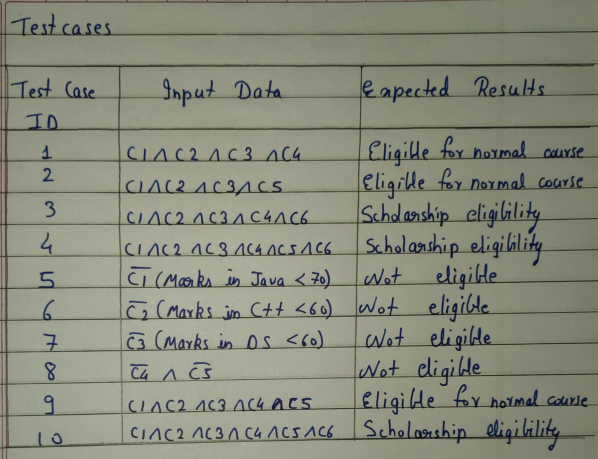
(ii) Eligible for scholarship course

(iii) Eligible for Normal course

Draw Cause Effect graph and Design test cases for this program using decision table testing.



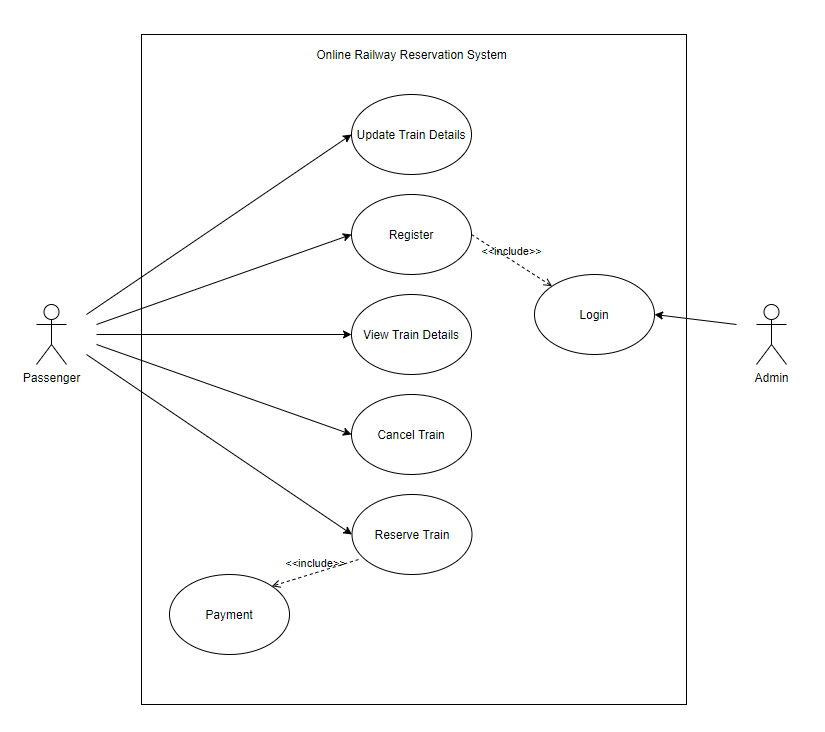




**D) Use case testing**

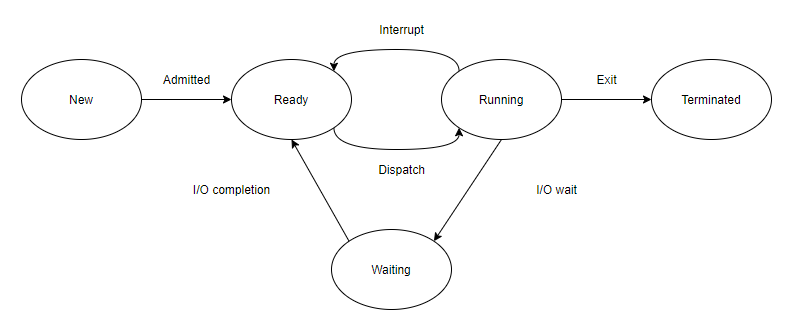
Draw a use case diagram for online railway reservation system. Derive

use case testcases.



**E) State Transition Testing**

Draw state diagram for 5 state process model of operating System. Derive state transition test cases for the same.



**CONCLUSION:** We have studied the Black Box Testing in software testing.