**LAB RECORD**



**BACHELOR OF TECHNOLOGY**

**B.Tech. CS&E Semester (7th - Sem)**

**(Academic Session – 2021 - 2022)**

**Course Code : IT414**

**Course Title : Software Testing & Quality Assurance**

**Enrollment No. : A7605218033**

**Name of Student : Neelesh Singh**

**Date of Submission : 10-11-2021**

**Signature of Student : *Neelesh***

**Grade/Marks Obtained :**

**Faculty Name & Signature: Dr. Puneet Sharma**

**Department of Computer Science & Engineering**

**Amity School of Engineering & Technology**

**Amity University, Lucknow Campus**

****

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Date** | **Name of Experiment** | **Page No.** | **Sign.** |
| 1 | 02-08-2021 | Design test cases using Boundary value analysis by taking quadratic equation problem | 4 |  |
| 2 | 09-08-2021 | Design test cases using Equivalence class partitioning taking triangle problem | 6 |  |
| 3 | 16-08-2021 | Design test cases to validate the Mobile Number. | 8 |  |
| 4 | 08-09-2021 | Design the test cases to validate the PAN Card Number. | 10 |  |
| 5 | 15-09-2021 | Design independent paths by calculating cyclometic complexity using date problem | 12 |  |
| 6 | 22-09-2021 | Design of test cases to determine the previous date | 14 |  |
| 7 | 29-09-2021 | Generate test case for ATM card number | 15 |  |
| 8 | 13-10-2021 | Write a test cases for Aadhar Number Verification | 17 |  |
| 9 | 20-10-2021 | Overview of Testing process using Rational Robot | 19 |  |
| 10 | 27-10-2021 | Write a script to record verification point using Rational Robot (For GUI testing of single click on window OS) | 23 |  |
|  | | | |  |

**TEST CASES:**

“A test case has a component that describes an input, action or event and an expected response, to determine if a feature of an application is working correctly.”

There are levels in which each test case will fall in order to avoid duplication efforts.

**Level 1:** In this level you will write the **basic test cases from the available specification** and user documentation.  
**Level 2:** This is the **practical stage** in which writing test cases depend on actual functional and system flow of the application.  
**Level 3:** This is the stage in which you will group some test cases and **write a test procedure**. Test procedure is nothing but a group of small test cases maximum of 10.  
**Level 4:**  **Automation of the project.** This will minimize human interaction with system and thus QA can focus on current updated functionalities to test rather than remaining busy with regression testing.

**HOW TO WRITE EFFECTIVE TEST CASES:**

Writing **effective test cases** is a skill and that can be achieved by some experience and in-depth study of the application on which test cases are being written.

**Fields in test cases:**

**1) Test case id:   
2) Unit to test:** What to be verified?  
**3) Assumptions:   
4) Test data:** Variables and their values  
**5) Steps to be executed:   
6) Expected result:**

**WHY WE WRITE TEST CASES:**

The basic objective of writing test cases is **to validate the testing coverage of the application.** If you are working in any CMMI company then you will strictly follow test cases standards. So writing test cases brings some sort of standardization and minimizes the ad-hoc approach in testing.

**TEST PLAN**

A **test plan** is a document detailing a systematic approach to testing a system such as a [machine](http://en.wikipedia.org/wiki/Machine) or [software](http://en.wikipedia.org/wiki/Software). The plan typically contains a detailed understanding of what the eventual workflow will be. A test plan documents the strategy that will be used to verify and ensure that a product or system meets its design specifications and other requirements. A test plan is usually prepared by or with significant input from [Test Engineers](http://en.wikipedia.org/wiki/Test_Engineer).

Depending on the product and the responsibility of the organization to which the test plan applies, a test plan may include one or more of the following:

* Design verification or Compliance test - to be performed during the development or approval stages of the product, typically on a small sample of units.
* Manufacturing or Production test - to be performed during preparation or assembly of the product in an ongoing manner for purposes of performance verification and quality control.
* Acceptance or Commissioning test - to be performed at the time of delivery or installation of the product.
* Service and Repair test - to be performed as required over the service life of the product.
* Regression test - to be performed on an existing operational product, to verify that existing functionality didn't get broken when other aspects of the environment are changed (e.g., upgrading the platform on which an existing application runs).

A complex system may have a high level test plan to address the overall requirements and supporting test plans to address the design details of subsystems and components.

Test plan document formats can be as varied as the products and organizations to which they apply. There are three major elements that should be described in the test plan: Test Coverage, Test Methods, and Test Responsibilities. These are also used in a formal [test strategy](http://en.wikipedia.org/wiki/Test_strategy).

**TEST SUITE:**

 A **Test suite**, less commonly known as a *validation suite*, is a collection of [test cases](http://en.wikipedia.org/wiki/Test_case) that are intended to be used to test a software program to show that it has some specified set of behaviors. A test suite often contains detailed instructions or goals for each collection of test cases and information on the system configuration to be used during testing. A group of test cases may also contain prerequisite states or steps, and descriptions of the following tests.

**EXPERIMENT NO. 1**

**Objective:**

**Design test cases using Boundary value analysis by taking quadratic equation problem.**

**Procedure:**

1. ax2+bx+c=0
2. Discriminant D = b2-4ac
3. Roots are real if D > 0
4. Roots are imaginary if D < 0
5. Roots are equal if D = 0
6. Equation is not quadratic if a = 0

**Formula:**

Discriminant

=IF (B2=0, "Not a quadratic equation", (C2\*C2)-(4\*B2\*D2))

Root 1

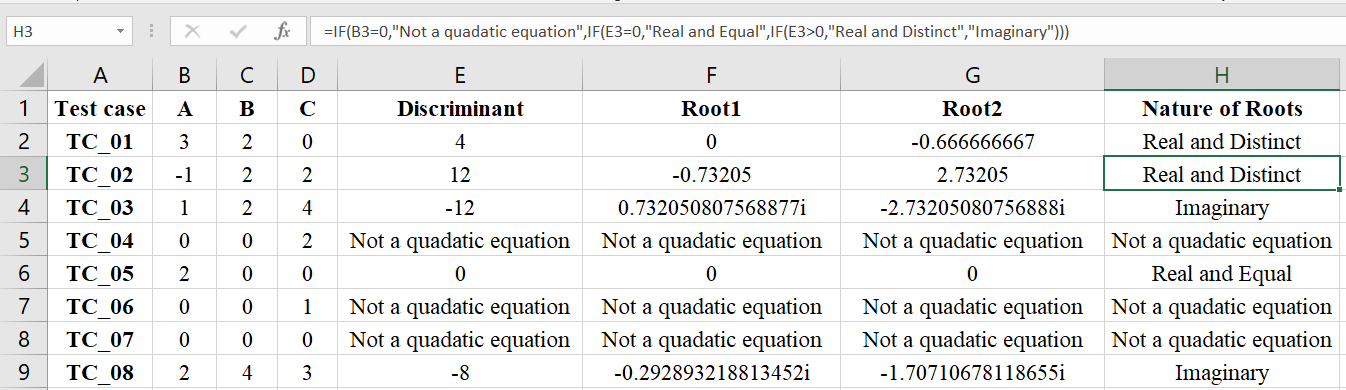
=IF (B2=0, "Not a quadratic equation", IF (E2>=0, (C2+SQRT (E2))/ (2\*B2), COMPLEX (0, (-C2+SQRT (-E2))/ (2\*B2))))

Root 2

=IF (B2=0, "Not a quadratic equation", IF (E2>=0, (-C2- RT (E2))/ (2\*B2), COMPLEX (0, (-C2-SQRT (-E2))/ (2\*B2))))

**Test Cases:**

| **Test case** | **A** | **B** | **C** | **Discriminant** | **Root1** | **Root2** | **Nature of Roots** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC\_01 | 3 | 2 | 0 | 4 | 0 | -0.666666667 | Real and Distinct |
| TC\_02 | -1 | 2 | 2 | 12 | -0.732050808 | 2.732050808 | Real and Distinct |
| TC\_03 | 1 | 2 | 4 | -12 | 0.732050807568877i | -2.73205080756888i | Imaginary |
| TC\_04 | 0 | 0 | 2 | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation |
| TC\_05 | 2 | 0 | 0 | 0 | 0 | 0 | Real and Equal |
| TC\_06 | 0 | 0 | 1 | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation |
| TC\_07 | 0 | 0 | 0 | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation | Not a quadatic equation |
| TC\_08 | 2 | 4 | 3 | -8 | -0.292893218813452i | -1.70710678118655i | Imaginary |

**Output:**

**Result:**

The above experiment is tested and executed successfully.

**EXPERIMENT NO. 2**

**Objective:**

**Design test cases using Equivalence class partitioning taking triangle problem**

**Procedure:**

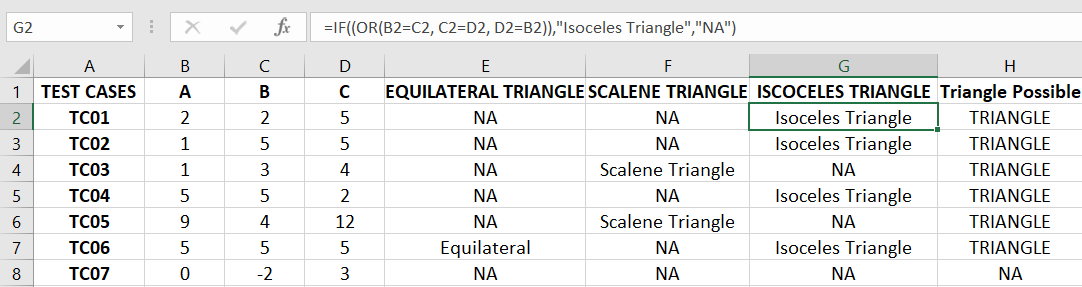
1. The sum of all the angles in triangle is 180°.
2. All the angles are equal, then Equilateral,
3. Only two angles are equal, then Isosceles,
4. All the angle differ, the Scalene,
5. Sum of the entire angle exceed 180°, then Not a Triangle.

**Formula:**

=IF((B2=C2)\*AND(C2=D2)\*AND(B2=D2),"EQUILATERAL",IF((B2<>C2)\*AND(C2<>D2)\*AND(B2<>D2),"SCALENE",IF(B2+C2+D2>180,"NOT A TRIANGLE",IF(B2+C2+D2<0,"NOT A TRAINGLE","ISOSCELES"))))

**Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **A** | **B** | **C** | **Type of Triangle** |
| **TC01** | 1 | 50 | 50 | ISOSCELES |
| **TC02** | 5 | 5 | 5 | EQUILATERAL |
| **TC03** | 99 | 50 | 50 | NOT A TRIANGLE |
| **TC04** | 100 | 50 | 50 | NOT A TRIANGLE |
| **TC05** | 50 | 50 | 50 | EQUILATERAL |
| **TC06** | 50 | 1 | 50 | ISOSCELES |
| **TC07** | 50 | 2 | 50 | ISOSCELES |
| **TC08** | 9 | 5 | 10 | SCALENE |
| **TC09** | 0 | 5 | 5 | ISOSCELES |
| **TC10** | -5 | -5 | 0 | NOT A TRAINGLE |

**Output:**

**Result:**

The above experiment is tested and executed successfully.

**EXPERIMENT NO. 3**

**Objective:** **Design test cases to validate the Mobile Number.**

**Procedure:**

1.Verify the mobile number text field by entering the valid 10-digit mobile number.

•Verify the mobile number text field by entering the 11-digit mobile number. (Condition-first digit should be 0)

•Verify the mobile number text field by entering the valid 12-digit mobile number. (Condition-first two digit should be 91)

2.Verify the mobile number field by entering blank space between the number

3.Verify the mobile number field without entering any value.

4.Verify the mobile number field by entering the special character

5.Verify the mobile number field by entering the 10 "Zero" in text box

6.If 10-digit mobile number, then first two digits should be between 7 to 9

7.If 12-digit mobile number, then third and fourth digits should be between 7 to 9

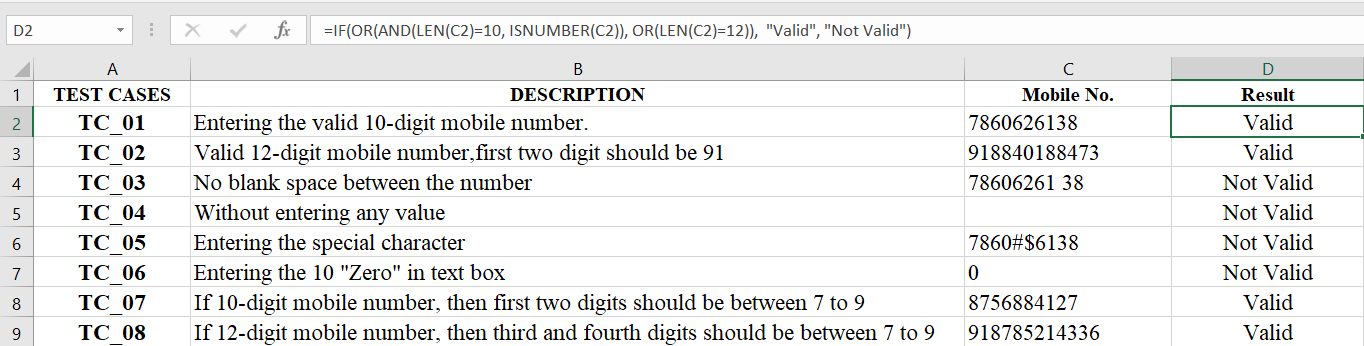
**Formula:**

=IF(AND(MID(C2,1,1)<>"",MID(C2,2,1)<>"",MID(C2,3,1)<>"",MID(C2,4,1)<>"",MID(C2,5,1)<>"",MID(C2,6,1)<>"",MID(C2,7,1)<>"",MID(C2,8,1)<>"",MID(C2,9,1)<>"",MID(C2,10,1)<>""),AND(COUNT(FIND(MID(C2,1,1),"1234567890"))=1,COUNT(FIND(MID(C2,2,1),"1234567890"))=1,COUNT(FIND(MID(C2,3,1),"1234567890"))=1,COUNT(FIND(MID(C2,4,1),"1234567890"))=1,COUNT(FIND(MID(C2,5,1),"1234567890"))=1,COUNT(FIND(MID(C2,6,1),"1234567890"))=1,COUNT(FIND(MID(C2,7,1),"1234567890"))=1,COUNT(FIND(MID(C2,8,1),"1234567890"))=1,COUNT(FIND(MID(C2,9,1),"1234567890"))=1,COUNT(FIND(MID(C2,10,1),"1234567890"))=1))

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** |  | **Input** | **Result** |
| **TC01** | Total 10 digits should be present | 7007889369 | TRUE |
| **TC04** | No the blank space between the number | 7007889 69 | FALSE |
| **TC05** | Enter 10 "Zero" in text box | "0000000000" | FALSE |
| **TC06** | Enter only 10 blank space. |  | FALSE |
| **TC07** | Enter without giving any value |  | FALSE |
| **TC08** | No special character in between | 70078893@9 | FALSE |

**Output:**

Graphical user interface, text, application, email

Description automatically generated

**Result:**

The above experiment is tested and executed successfully.

**EXPERIMENT NO. 4**

**Objective:**

**Design the test cases to validate the PAN Card Number.**

**Procedure:**

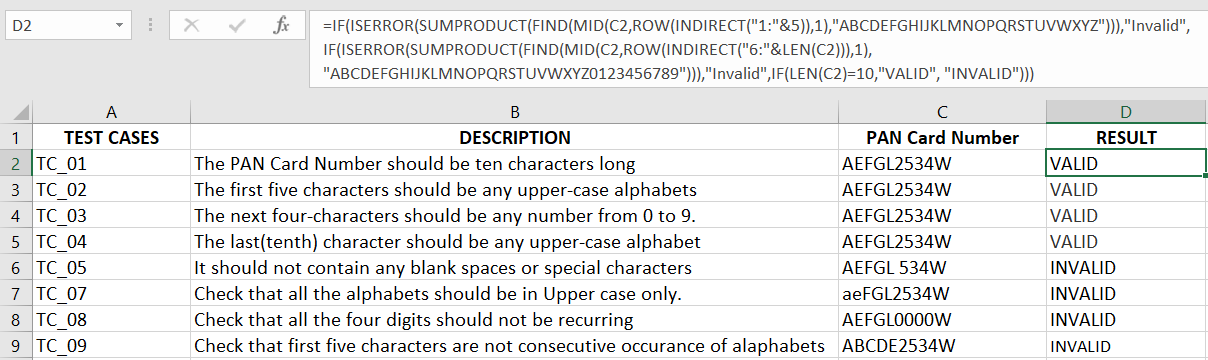
1. The PAN Card Number should be ten **characters** long.
2. The first five characters should be any upper-case alphabets.
3. The next four-characters should be any number from 0 to 9.
4. The last(tenth) character should be any upper-case
5. It should not contain any blank spaces or special characters.
6. Check that all the alphabets should be in Upper case only.
7. There should not be Consecutive Occurrence of Alphabets

**Formula:**

=IF(ISERROR(SUMPRODUCT(FIND(MID(C2,ROW(INDIRECT("1:"&5)),1),"ABCDEFGHIJKLMNOPQRSTUVWXYZ"))),"Invalid",IF(ISERROR(SUMPRODUCT(FIND(MID(C2,ROW(INDIRECT("6:"&LEN(C2))),1),"ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"))),"Invalid",IF(LEN(C2)=10,"Valid", "Invalid")))

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| TC01 | It should be ten characters long | AEFGL2534W | Valid |
| TC02 | It should be ten characters long | AEFGL2534W | Invalid |
| TC03 | The first five characters should be any upper-case alphabets | AEFGL2534W | Valid |
| TC04 | The next four-characters should be any number from 0 to 9. | AEFGL2534W | Valid |
| TC05 | The last(tenth) character should be any upper-case alphabet | AEFGL2534W | Valid |
| TC06 | It should not contain any blank spaces. | AEFGL 2534W | Invalid |
| TC07 | Check that all the alphabets should be in Upper case only | AEFGL2534W | Valid |
| TC08 | Check that all the alphabets should be in Upper case only | aeFGL2534W | Invalid |

**Output:**

**Result:**

The above experiment is tested and executed successfully.

**EXPERIMENT NO-5**

**Objective:**

**Design independent paths by calculating cyclometric complexity using date problem Date.**

**Procedure:**

1. Date should be between 1 to 31 if month is 1,3,5,7,8,10,12
2. Date should be between 1 to 30 if month is 4,6,9,11
3. Date should be between 1 to 29 if month is Feb and its leap year else Date should be between 1 to 28.
4. Month should be between 1 to 12
5. Year should be greater than 1900.

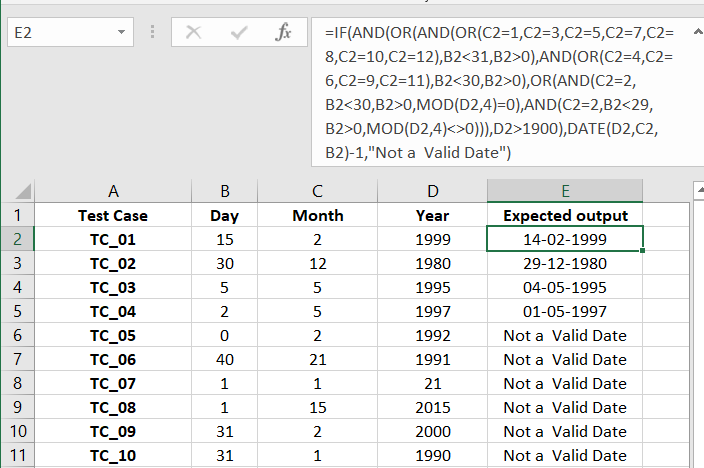
**Formula:**

=IF (AND ( OR ( AND ( OR ( Month=1 , Month =3 , Month =5 , Month =7 , Month =8 , Month =10 , Month =12 ) , Day < 31 , Day >0 ) ,   
AND ( OR (Month =4 , Month =6 , Month =9 , Month =11) ,Day <30, Day>0) ,OR ( AND ( Month =2 , Day <30 , Day >0 , MOD(Year,4)=0 ) ,   
AND ( Month =2 , Day <29, Day >0, MOD(Year,4)<>0 ) ) ) , Year >1900 ) ,   
 DATE( Year , Month , Day ) - 1, " Not a Valid Date " )

**Test Cases:**

| **Test Case** | **Day** | **Month** | **Year** | **Expected output** |
| --- | --- | --- | --- | --- |
| **TC\_01** | 15 | 2 | 1999 | 14-02-1999 |
| **TC\_02** | 30 | 12 | 1980 | 29-12-1980 |
| **TC\_03** | 5 | 5 | 1995 | 04-05-1995 |
| **TC\_04** | 2 | 5 | 1997 | 01-05-1997 |
| **TC\_05** | 0 | 2 | 1992 | Not a Valid Date |
| **TC\_06** | 40 | 21 | 1991 | Not a Valid Date |
| **TC\_07** | 1 | 1 | 21 | Not a Valid Date |
| **TC\_08** | 1 | 15 | 2015 | Not a Valid Date |
| **TC\_09** | 31 | 2 | 2000 | Not a Valid Date |
| **TC\_10** | 31 | 1 | 1990 | Not a Valid Date |

**Sample Output:**



**Result:** The above experiment is tested and executed successfully.

**EXPERIMENT NO. 6**

**Objective:** Design of test cases to determine the previous date

**Procedure:**

1<=Date<=31  
1<=Month<=12  
Previous date, (Day, Month, Year)-1

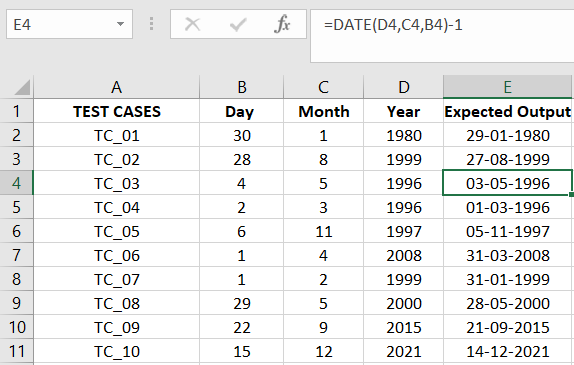
**Formula:**

=DATE (Year, Month, Day) - 1

**Test Cases:**

| **TEST CASES** | **Day** | **Month** | **Year** | **Expected Output** |
| --- | --- | --- | --- | --- |
| TC\_01 | 30 | 1 | 1980 | 29-01-1980 |
| TC\_02 | 28 | 8 | 1999 | 27-08-1999 |
| TC\_03 | 4 | 5 | 1996 | 03-05-1996 |
| TC\_04 | 2 | 3 | 1996 | 01-03-1996 |
| TC\_05 | 6 | 11 | 1997 | 05-11-1997 |
| TC\_06 | 1 | 4 | 2008 | 31-03-2008 |
| TC\_07 | 1 | 2 | 1999 | 31-01-1999 |
| TC\_08 | 29 | 5 | 2000 | 28-05-2000 |
| TC\_09 | 22 | 9 | 2015 | 21-09-2015 |
| TC\_10 | 15 | 12 | 2021 | 14-12-2021 |

**Sample Output:**



**Result:** The above experiment is tested and executed successfull

**EXPERIMENT NO. 7**

**Objective:**

**Write a Test case for verifying type of ATM card number.**

**Procedure:**

**Check for the type of ATM Card**

* **Visa**
* **MasterCard**
* **Rupay**
* **American Express**

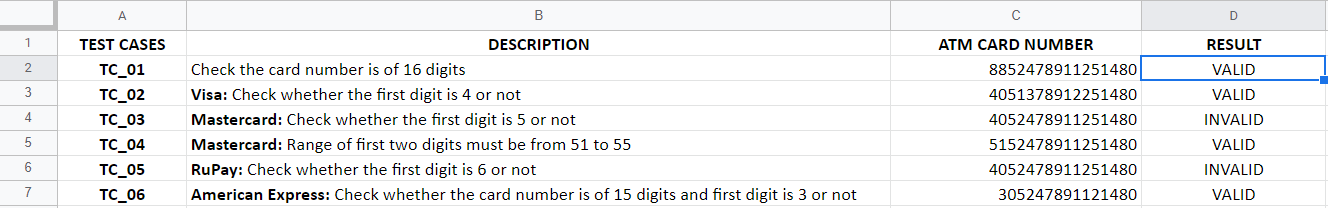
1. Check the card number is of 16 digits
2. Mastercard: Check whether the first digit is 5 or not, Range will be 51-55
3. Maestrocard: Range from 50-55.
4. Visa: Check whether the first digit is 4 or not.
5. RuPay: Check whether the first digit is 6 or not

**Formula:**

=IF ( AND ( Card Type="Maestro" , IsNumber( PIN ) , LEN( ATM\_Number ) = 16 , LEN ( PIN ) = 4 , LEN ( CVV ) = 0 ) , " Valid Mastero " ,   
IF ( AND ( Card Type <> " Maestro " , IsNumber ( PIN ) , IsNumber( CVV ) , LEN ( ATM\_Number ) = 16 , LEN ( PIN ) = 4 , LEN ( CVV ) = 3 ) , " Valid Visa , Rupay or Master Card " , " Invalid " ) )

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Description** | **Input** | **Result** |
| **TC\_01** | Check the card number is of 16 digits | 8852478911251480 | TRUE |
| **TC\_02** | **Visa:** Check whether the first digit is 4 or not | 4051378912251480 | TRUE |
| **TC\_03** | **Mastercard:** Check whether the first digit is 5 or not | 4052478911251480 | FALSE |
| **TC\_04** | **Mastercard:** Range of first two digits must be from 51 to 55 | 5152478911251480 | TRUE |
| **TC\_06** | **RuPay:** Check whether the first digit is 6 or not | 4052478911251480 | TRUE |
| **TC07** | **American Express:** Check whether the card number is of 15 digits and first digit is 3 or not | 305247891121480 | TRUE |

**Output:**

**Result:**

The above experiment is tested and executed successfully.

**EXPERIMENT NO. 8**

**Objective:**

**Design of test cases for Aadhar number verification**

**Procedure:**

1. Aadhar number should be of 12 digit.
2. Number of digit should not be less that 12 digit
3. Should not contain alphabets
4. Should not have any whitespaces
5. Should not have special Characters
6. Should not start with 0 or 1
7. Check whether Mobile number is registered or not
8. first two digits should not be zero

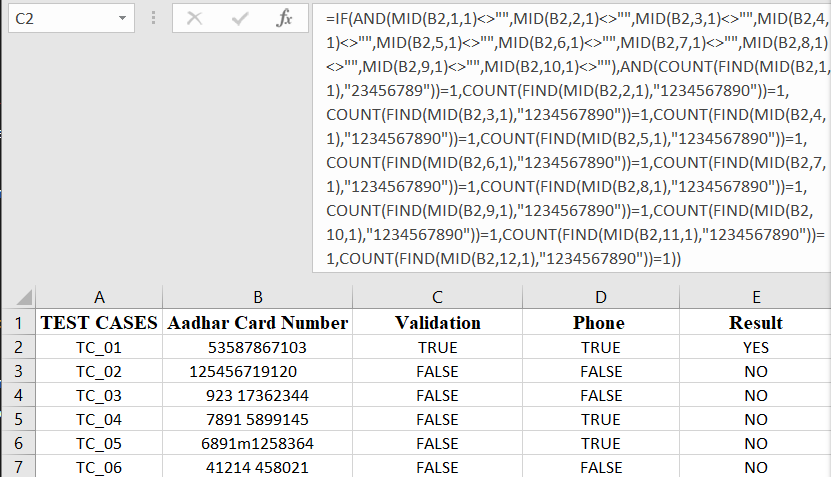
**Formula:**

* =IF(AND(MID(B2,1,1)<>"",MID(B2,2,1)<>"",MID(B2,3,1)<>"",MID(B2,4,1)<>"",MID(B2,5,1)<>"",MID(B2,6,1)<>"",MID(B2,7,1)<>"",MID(B2,8,1)<>"",MID(B2,9,1)<>"",MID(B2,10,1)<>""),AND(COUNT(FIND(MID(B2,1,1),"23456789"))=1,COUNT(FIND(MID(B2,2,1),"1234567890"))=1,COUNT(FIND(MID(B2,3,1),"1234567890"))=1,COUNT(FIND(MID(B2,4,1),"1234567890"))=1,COUNT(FIND(MID(B2,5,1),"1234567890"))=1,COUNT(FIND(MID(B2,6,1),"1234567890"))=1,COUNT(FIND(MID(B2,7,1),"1234567890"))=1,COUNT(FIND(MID(B2,8,1),"1234567890"))=1,COUNT(FIND(MID(B2,9,1),"1234567890"))=1,COUNT(FIND(MID(B2,10,1),"1234567890"))=1,COUNT(FIND(MID(B2,11,1),"1234567890"))=1,COUNT(FIND(MID(B2,12,1),"1234567890"))=1))
* =IF(AND(C2=TRUE,D2=TRUE),"YES","NO")

**Test Cases:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TEST CASES** | **Aadhar Card Number** | **Validation** | **Phone** | **Result** |
| TC\_01 | 53587867103 | TRUE | TRUE | YES |
| TC\_02 | 125456719120 | FALSE | FALSE | NO |
| TC\_03 | 923 17362344 | FALSE | FALSE | NO |
| TC\_04 | 7891 5899145 | FALSE | TRUE | NO |
| TC\_05 | 6891m1258364 | FALSE | TRUE | NO |
| TC\_06 | 41214 458021 | FALSE | FALSE | NO |

**Output:**



**Result:**

The above experiment is tested and executed successfully.

**Experiment No. 9**

**Objective:**

**Overview of Testing and basics using Rational Robot.**

**Theory:**

1. **Features of Rational Robot**

Rational Robot is an automated functional, regression testing tool for automating Windows, Java, IE and ERP applications under windows platform.  Rational Robot provides test cases for common objects such as menus, lists, bitmaps and specialized test cases for objects specific to the development environment. It integrates with tools like Rational Test Manager, Rational Clear quest and Requisite Pro in the Rational Unified Processor for Defect Tracking, Change Management and Requirement Traceability. It also supports UI technologies like Java, the Web, all VS.NET controls, Oracle Forms, Borland Delphi and Sybase Power Builder applications.

1. **Rational Administrator**

It is a tool for managing associations between rational artifacts such as Test Data stores, Requisite Pro projects and Rose models.

* Rational Projects are created using Rational Administrator
* Users and Groups can be maintained
* Project assets can be upgraded

1. **Recording Options**Using Object oriented technology, Robot identifies an object by its name property not by its location coordinates. There are two different options

* GUI  –   Functional Testing
* VU   –   Performance Testing

1. **SQABasic language**

SQABasic is similar to Microsoft Visual Basic. All the scripts will be in scriptname.rec format.  When you playback the script, Robot automatically compiles and runs the script, which repeats your actions and executes the verification points.

1. **Shell Scripts**

It is a master script that calls other automated scripts and plays them back in sequence. “Call script test1” is a command to call script named test1. Combined into a single shell script, scripts can run in unattended mode and perform comprehensive test coverage.  It centralizes test results into one test log.

1. **Low level Recording**

Turn “Low Level Recording On” in Robot during recording, mouse and keyboard actions are automatically stored in an external file.

1. **Verification Points**

Verification points verify that a certain action has taken place, or verify the state of an object. There are 11 Verification points in Robot

* **Alpha-numeric:** Verifies alphanumeric data.  Used for edit boxes, pushbuttons,    labels, text fields, etc.,
* **Object Properties:** Tests object attributes such as color, font and position.
* **Menu:** Verifies the menu values and optionally their state (enabled or disabled) of a window
* **Clip Board:** Verifies the contents of the windows clipboard
* **Window Existence:**  Tests to see if a particular window does or does not exist on the screen.
* **Region Image:** Graphically compares an area of the screen you specify
* **Window Image:** Graphically compares an entire window such as a window box.
* **Object Data:** Test data contents of objects (eg. Dropdown)
* **File Comparison:** Compares the contents of the two files (size and the contents)
* **File Existence:** Checks for the existence of a specified file
* **Module Existence**: Used to verify whether a specified module is loaded into a specified context, or loaded anywhere in memory.

When you are creating verification points, there will be two options – Wait State and

**Expected Results-** Wait states are useful when AUT requires an unknown amount of time to complete a task. Using a wait state keeps the verification point form failing if the task is not completed immediately or if the data is not accessible immediately.  
  
**Expected Results** – Click Pass or Fail in the Verification Point Name dialog box.

1. **Variable Window**

During debugging, if you want to examine variable and constant values, you can variables window. View->Variables.

1. **Object Mapping**

If AUT contains a custom object or any object that Robot does not recognize, you can create a custom object mapping before start recording.  By adding the object’s class to the list of classes that Robot recognizes, and then associating the class to a standard object type. Robot saves this custom class/object type mapping in the project and uses it to identify the custom object during playback.

1. **Debug Tools**

**Animate (F11)** – Animation mode allows you to see each line of script as it executes.  
**Step Over (F10)** – Use to execute a single command line within a script  
**Step into (F8)** – Use to being single step execution  
**Step out (F7)** – Use to step out of the called script and return to the calling script.  
**Go Until Cursor (F6)** – Use to play back the active GUI script, stopping at the text cursor location.

1. **Library Files and Header Files**

Header files have .sbh extensions and contain the procedure declarations and global variables referred to in your script files.  There are two types of library files. Those with .sbl extensions can’t have verification points. Those with .rec extensions are stored in the project and can have verification points. Both Header and library are in \SQABAS32 in the project directory.

1. **Image Masks used for dynamic objects**  
   Image masks are used to hide an area of the screen. When you play back a script that contains an Image VP and a mask, Robot ignores the masked area when comparing actual results to the recorded baseline.
2. **Data Pool**   
   A Data pool is a test dataset that supplies data variables in a test script during playback.  Using data pools allows you to run multiple iterations of a script using different data each time.  It can be created and managed using Test Manager for data driven tests.

**Experiment No. 10**

**Objective:**

**Write a test case to test login window using manual testing.**

**Theory:**

Login window is partly a user process , it starts before anyone logs into the computer and is responsible for displaying the login screen ( or not, if auto login is set), validating login attempts, and setting up the user environment ( launching the Finder, Dock, any login apps, etc) at login. It also acts as a process monitor for user processes, restarts the Finder or Dock if they crash, and implements the Force Quit Application window. Finally, it handles the logout, restart, and shutdown procedures.

**Requirements for login screen:**

1. Login window should contain two fields i.e. username, password and OK button.
2. After successful login, should show confirmation message “Login Screen”.
3. If we enter invalid data, should show invalid message “Login data incorrect”.
4. Username and password both are case sensitive.

**Test cases:**

1. Verify whether “Login Screen” message is displayed when user enters valid data.
2. Verify whether “Login data incorrect” message is displayed without entering any username and password and select OK button.
3. Verify whether “incorrect login”, message is displayed when user enter wrong username and password and select submit.
4. Verify whether “Login data incorrect” message is displayed when user enters only username and selects submit.
5. Verify whether “Login data incorrect” displayed when user enter only password and selects submit.
6. Verify whether “Login data incorrect” displayed when user enter valid user name but wrong password and selects OK.
7. Verify whether “Login data incorrect” message is displayed when user enters invalid username and valid password.
8. Verify whether “Login data incorrect” message is displayed when user enter different space for username and / Or password.
9. Try copy/paste in the password textbox.