Software Testing and Quality Assurance

ETCS -453

Faculty: **Dr. Ruchi Goel**

Name: **Syeda Reeha Quasar**

Roll No.: **14114802719**

Semester: **7**

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce "Critical thinkers of Innovative Technology"

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur.**

VISION

To nurture young minds in a learning environment of high academic value and im-bibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware - Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carriedout by Hardware equipment as well as the related software enabling deeper un- derstanding of basic concepts and encouraging inquisitive nature.

Life - Long Learning

The Institute strives to match technological advancements and encourage studentsto keep updating their knowledge for enhancing their skills and inculcating theirhabit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above at- tributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all En- gineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to createa platform for multi-disciplinary approach.

EXPERIMENT - 1

Software Testing and Quality Assurance

Abstract

To determine the nature of roots of quadratic equations, its input is triple of +ve integers (say x,y,z) and values may be from interval[1,100] the program output may have one of the following:- [Not a Quadratic equations, Real roots, Imaginary roots, Equal roots]. Perform BVA and Robust Case Testing.

Syeda Reeha Quasar

14114802719

EXPERIMENT – 1

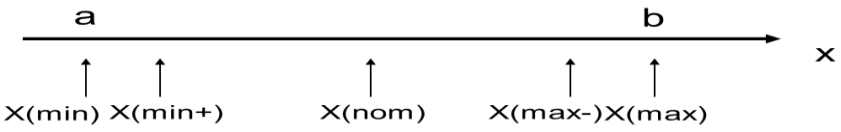
Aim:

Write a program to find whether a number is prime or not. To determine the nature of roots of a quadratic equation, its input is triple of +ve integers (say a,b,c) and values may be from the interval[1,100] The program output may have one of the following:-

- [Not Quadratic equations, Real roots, Imaginary roots, Equal roots]
- Perform BVA (Boundary-value analysis).
- Perform Robust Case Testing

Theory:

BVA(Boundary value analysis)is a 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 center of input The basic boundary value analysis technique can be generalized in two ways: by the number of variables, and by the kinds of ranges. Generalizing the number of variables is easy: if we have a function of n variables, we hold all but one at the nominal values, and let the remaining variable assume the min, min+, nom, max- and max values, and repeat this for each variable. Thus for a function of n variables, boundary value analysis yields 4n + 1 test cases.



In the above program consider the value 1(minimum), 2(just above Minimum), 50 (Nominal), 99(Just below Maximum) and 100(Maximum). Total No. of test cases will be 4*3+1=13

If a, b, and c denote the three integer in quadratic equation $a(x^2)+bx+c=0$ then Calculate discriminant d=(b*b)-4*a*c

- if((a<1)||(b<1)||(c<1)||(a>100)||(b>100)||(c>100)) then "Invalid input"
- if(a==0) then "Not a quadratic equation"
- if (d==0) then "Roots are equal"
- if(d<0) then "Imaginary roots"
- otherwise "Real Roots"

Robust Test Cases –
Here, we go outside the legitimate boundary, it is an extension of boundary value analysis.

Source Code:

```
import cmath

print("Enter the values of variable for the standard Quadratic equation ax^2 + bx
+ c = 0")
a = int(input("Enter Value of a: "))
b = int(input("Enter Value of b: "))
c = int(input("Enter Value of c: "))

sqrt_d = cmath.sqrt((b**2) - (4*a*c))

root1 = (-b - sqrt_d)/(2*a)
root2 = (-b + sqrt_d)/(2*a)

print('The real roots of the equation "{0}x^2 + {1}x + {2} = 0" are {3} and
{4}\n'.format(a, b, c, root1.real, root2.real))

print('The imaginary roots of the equation "{0}x^2 + {1}x + {2} = 0" are {3} and
{4}'.format(a, b, c, root1.imag, root2.imag))
```

Output:

```
PS E:\sem 6\Software-Testing> python .\quadraticEquation.py
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 1
Enter Value of b: 5
Enter Value of c: 6
The real roots of the equation "1x^2 + 5x + 6 = 0" are -3.0 and -2.0

The imaginary roots of the equation "1x^2 + 5x + 6 = 0" are 0.0 and 0.0
```

```
PS E:\sem 6\Software-Testing> python .\quadraticEquation.py
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 50
Enter Value of b: 50
Enter Value of c: 2
The real roots of the equation "50x^2 + 50x + 2 = 0" are -0.9582575694955839 and -0.041742430504416034

The imaginary roots of the equation "50x^2 + 50x + 2 = 0" are 0.0 and 0.0
PS E:\sem 6\Software-Testing> python .\quadraticEquation.py
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 50
Enter Value of b: 50
Enter Value of c: 50
The real roots of the equation "50x^2 + 50x + 50 = 0" are -0.5 and -0.5

The imaginary roots of the equation "50x^2 + 50x + 50 = 0" are -0.8660254037844386 and 0.8660254037844386
```

```
PS E:\sem 6\Software-Testing> python .\quadraticEquation.py
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 50
Enter Value of b: 50
Enter Value of c: 1
The real roots of the equation "50x^2 + 50x + 1 = 0" are -0.9795831523312719 and -0.020416847668728054

The imaginary roots of the equation "50x^2 + 50x + 1 = 0" are 0.0 and 0.0
```

```
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 5
Enter Value of b: 10
Enter Value of c: 5
The real roots of the equation "5x^2 + 10x + 5 = 0" are -1.0 and -1.0

The imaginary roots of the equation "5x^2 + 10x + 5 = 0" are 0.0 and 0.0
PS E:\sem 6\Software-Testing> python .\quadraticEquation.py
Enter the values of variable for the standard Quadratic equation ax^2 + bx + c = 0
Enter Value of a: 5
Enter Value of b: 1
Enter Value of c: 5
The real roots of the equation "5x^2 + 1x + 5 = 0" are -0.1 and -0.1

The imaginary roots of the equation "5x^2 + 1x + 5 = 0" are -0.99498743710662 and 0.99498743710662
```

Test Cases:

Boundary Value analysis: The basic idea of boundary value analysis is to use input variable at their manimum, just above manimum, normal value, just below maximum and maximum.

In this program, we consider the value as 0 (minimum), 1(just above minimum), 50 (normal), 99 (just below maximum) and 100 (maximum).

Maximum of 4n+1 test cases

Test Case ID	Input Data			Expected Output	Actual Output	Pass/ Fail
	a	b	c			
1	1	50	50	Real Roots	Real Roots	Pass
2	2	50	50	Real Roots	Real Roots	Pass
3	50	50	50	Imaginary Roots	Imaginary Roots	Pass
4	99	50	50	Imaginary Roots	Imaginary Roots	Pass
5	100	50	50	Imaginary Roots	Imaginary Roots	Pass

6	**50**	**1**	**50**	**Imaginary Roots**	**Imaginary Roots**	**Pass**
7	**50**	**2**	**50**	**Imaginary Roots**	**Imaginary Roots**	**Pass**
8	**50**	**99**	**50**	**Imaginary Roots**	**Imaginary Roots**	**Pass**
9	**50**	**100**	**50**	**Real and Equal Roots**	**Real and Equal Roots**	**Pass**
10	**50**	**50**	**1**	**Real Roots**	**Real Roots**	**Pass**
11	**59**	**50**	**2**	**Real Roots**	**Real Roots**	**Pass**
12	**50**	**50**	**99**	**Imaginary Roots**	**Imaginary Roots**	**Pass**
13	**50**	**50**	**100**	**Imaginary Roots**	**Imaginary Roots**	**Pass**

Robust Testing: The term 'robust' is synonymous with strength. So robustness testing is the way to assess the quality of a software product. It the process of verifying whether a software system performs well under stress conditions or not.

The method of carrying out robustness testing follows a set of conventions. A set of invalid inputs or odd/stressful environment is set up. Sometimes it so happens that on providing certain inputs the program may crash . It becomes significant to capture those errors and rectify it in accordance with the requirement specifications.

Hence suitable test cases are developed to perform testing in an appropriate test environment. Total number of test cases generated in robust testing are **6*N +1** due to min-1 , min , min+1 , mid , max -1 , max and max+1.

Robustness testing ensures that a software system qualifies as the end product for which it was meant for, hence serving the right purpose. As we know that a complete software system comprises of various components, such kind of testing ensures reducing cost and time required for efficient delivery of a software system.

So robustness testing is carried out somewhat like this- a combination of valid and invalid inputs is passed to the system and checked for the performance . So a system is tested and validated under different conditions.

Maximum of 6n+1 test cases

Test Case Id	Input Data			Expected Output	Actual Output	Pass/ Fail
	a	b	c			
1	0	50	50	Invalid input	Invalid input	Pass
2	1	50	50	Real Roots	Real Roots	Pass
3	2	50	50	Real Roots	Real Roots	Pass
4	50	50	50	Imaginary Roots	Imaginary Roots	Pass
5	99	50	50	Imaginary Roots	Imaginary Roots	Pass
6	100	50	50	Imaginary Roots	Imaginary Roots	Pass
7	101	50	50	Invalid input	Invalid input	Pass
8	50	0	50	Invalid inputs	Invalid inputs	Pass
9	50	1	50	Imaginary roots	Imaginary roots	Pass
10	50	2	50	Imaginary roots	Imaginary roots	Pass
11	50	99	50	Imaginary roots	Imaginary roots	Pass
12	50	100	50	Equal roots	Equal roots	Pass
13	50	101	50	Invalid inputs	Invalid inputs	Pass
14	50	50	0	Invalid inputs	Invalid inputs	Pass
15	50	50	1	Real Roots	Real Roots	Pass

16	50	50	2	Real Roots	Real Roots	Pass
17	50	50	99	Imaginary Roots	Imaginary Roots	Pass
18	50	50	100	Imaginary Roots	Imaginary Roots	Pass
19	50	50	101	Invalid Inputs	Invalid inputs	Pass

Viva Questions

Q1. What is the difference BVA and Robustness Testing?

In BVA, we remain within the legitimate boundary of our range i.e. for testing we consider values like (min, min+, nom, max-, max) whereas in Robustness testing, we try to cross these legitimate boundaries as well.

Q2. What is Robustness Testing? Explain

Robustness testing is any quality assurance methodology focused on testing the robustness of software. That is the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

Q3. What kind of input do we need from the end user to begin proper testing?

The product has to be used by the user. He is the most important person as he has more interest than anyone else in the project. It includes Acceptance Plan, Requirement Documents, Risk Analysis, Live Data and Test Suite.

Q4. How many numbers of test cases are there in Robustness Testing?

Total Number of Test Cases = **6*N + 1** where N is the number of input variables.

Q5: How Many types of Black box Testing are there?

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

1 **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.

2 **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.

3 **Regression testing** - Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

EXPERIMENT - 2

Software Testing and Quality Assurance

Abstract

To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,lOO].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle]. Perform BVA, Equivalence Class Testing (Using Input Domain and Output domain). Which Technique is best for the given problem statement. Give reasons.

Syeda Reeha Quasar

14114802719

EXPERIMENT – 2

Aim:

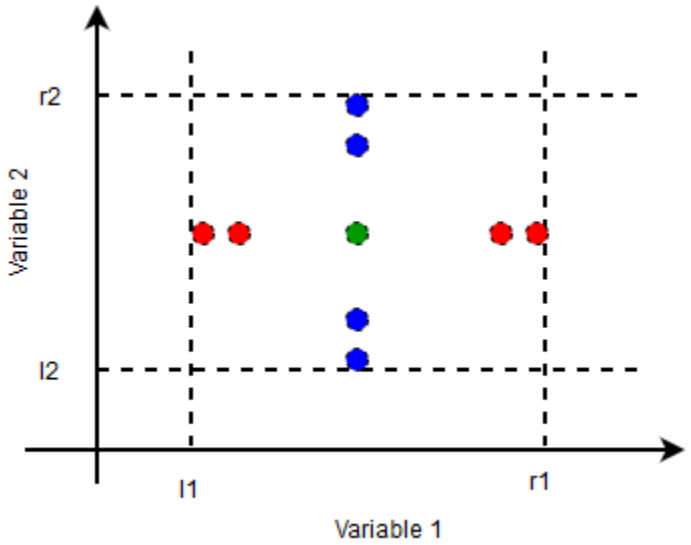
To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,lOO].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle]. Perform BVA, Equivalence Class Testing (Using Input Domain and Output domain). Which Technique is best for the given problem statement. Give reasons.

Theory:

Boundary Value Analysis (BVA) is a black box software testing technique where test cases are designed using boundary values. BVA is based on the *single fault assumption*, also known as critical fault assumption which states that failures are rarely the product of two or more simultaneous faults. Hence while designing the test cases for BVA we keep all but one variable to the nominal value and allowing the remaining variable to take the extreme value.

Test Case Design for BVA:

While designing the test cases for BVA first we determine the number of input variables in the problem. For each input variable, we then determine the range of values it can take. Then we determine the extreme values and nominal value for each input variable.



If any of these conditions is violated output is Not a Triangle.

- For Equilateral Triangle all the sides are equal.

- For Isosceles Triangle exactly one pair of sides is equal.
- For Scalene Triangle all the sides are different.

The table shows the Test Cases Design for the Triangle Problem.The range value [l, r] is taken as [1, 100] and nominal value is taken as 50.

The total test cases is, **4n+1 = 4*3+1 = 13**

Source Code:

```
# Function definition to check validity
def is_valid_triangle(a,b,c):
    if a+b>=c and b+c>=a and c+a>=b:
        return True
    else:
        return False

# Function definition for type
def type_of_triangle(a,b,c):
    if a==b and b==c:
        print('Triangle is Equilateral.')
    elif a==b or b==c or a==c:
        print('Triangle is Isosceles.')
    else:
        print('Triangle is Scalane')

# Reading Three Sides
side_a = float(input('Enter length of side a: '))
side_b = float(input('Enter length of side b: '))
side_c = float(input('Enter length of side c: '))

# Function call & making decision
if is_valid_triangle(side_a, side_b, side_c):
    type_of_triangle(side_a, side_b, side_c)
else:
    print('Tringle is not possible from given sides.')
```

Output:

```
PS E:\sem 6\stqa> python -u "e:\sem 6\stqa\isTriangle.py"
Enter length of side a: 3
Enter length of side b: 6
Enter length of side c: 20
Tringle is not possible from given sides.
PS E:\sem 6\stqa> python -u "e:\sem 6\stqa\isTriangle.py"
Enter length of side a: 100
Enter length of side b: 50
Enter length of side c: 50
Triangle is Isosceles.
PS E:\sem 6\stqa> python -u "e:\sem 6\stqa\isTriangle.py"
Enter length of side a: 4
Enter length of side b: 8
Enter length of side c: 6
Triangle is Scalane
PS E:\sem 6\stqa> python -u "e:\sem 6\stqa\isTriangle.py"
Enter length of side a: 5
Enter length of side b: 5
Enter length of side c: 5
Triangle is Equilateral.
PS E:\sem 6\stqa> python -u "e:\sem 6\stqa\isTriangle.py"
Enter length of side a: 50
Enter length of side b: 50
Enter length of side c: 50
Triangle is Equilateral.
```

Test Cases:

Boundary Value analysis: The basic idea of boundary value analysis is to use input variable at their manimum, just above manimum, normal value, just below maximum and maximum.

In this program, we consider the value as 0 (minimum), 1(just above minimum), 50 (normal), 99 (just below maximum) and 100 (maximum).

Maximum of 4n+1 test cases

Test Case ID	Input Data			Expected Output	Actual Output	Pass/ Fail
	a	b	c			
1	1	5	5	Isosceles	Isosceles	Pass

2	2	5	5	Isosceles	Isosceles	Pass
3	9	5	5	Isosceles	Isosceles	Pass
4	10	5	5	Not a Triangle	Not a Triangle	Pass
5	5	1	5	Isosceles	Isosceles	Pass
6	5	2	5	Isosceles	Isosceles	Pass
7	5	9	5	Isosceles	Isosceles	Pass
8	4	8	6	Scalene	Scalene	Pass
9	50	50	50	Equilateral	Equilateral	Pass
10	5	10	5	Not a Triangle	Not a Triangle	Pass
11	100	100	100	Equilateral	Equilateral	Pass
12	5	5	10	Not a Triangle	Not a Triangle	Pass
13	5	5	5	Equilateral	Equilateral	Pass

Robust Testing: The term 'robust' is synonymous with strength. So robustness testing is the way to assess the quality of a software product. It the process of verifying whether a software system performs well under stress conditions or not.

Total test cases, = 6*n+1 = 6*3+1 = 19

The method of carrying out robustness testing follows a set of conventions. A set of invalid inputs or odd/stressful environment is set up. Sometimes it so happens that on providing certain inputs the program may crash . It becomes significant to capture those errors and rectify it in accordance with the requirement specifications.

Hence suitable test cases are developed to perform testing in an appropriate test environment. Total number of test cases generated in robust testing are **6*N +1** due to min-1 , min , min+1 , mid , max -1 , max and max+1.

So robustness testing is carried out somewhat like this- a combination of valid and invalid inputs is passed to the system and checked for the performance . So a system is tested and validated under different conditions.

Maximum of 6n+1 test cases

Test Case Id	Input Data			Expected Output	Actual Output	Pass/ Fail
	a	b	c			
1	0	50	50	Invalid input	Invalid input	Pass
2	50	50	50	Equilateral	Equilateral	Pass
3	5	10	5	Not a Triangle	Not a Triangle	Pass
4	100	100	100	Equilateral	Equilateral	Pass
5	5	5	10	Not a Triangle	Not a Triangle	Pass
6	5	5	5	Equilateral	Equilateral	Pass
7	101	50	50	Invalid input	Invalid input	Pass
8	50	0	50	Invalid inputs	Invalid inputs	Pass
9	5	1	5	Isosceles	Isosceles	Pass
10	5	2	5	Isosceles	Isosceles	Pass
11	5	9	5	Isosceles	Isosceles	Pass
12	4	8	6	Scalene	Scalene	Pass

13	50	101	50	Invalid inputs	Invalid inputs	Pass
14	50	50	0	Invalid inputs	Invalid inputs	Pass
15	1	5	5	Isosceles	Isosceles	Pass
16	2	5	5	Isosceles	Isosceles	Pass
17	9	5	5	Isosceles	Isosceles	Pass
18	10	5	5	Not a Triangle	Not a Triangle	Pass
19	50	50	101	Invalid Inputs	Invalid inputs	Pass

Result:
BVA and Robust testing was performed for given conditions and verified.

Viva Questions

Q1. What is equivalence partitioning explain with example.
In BVA, we remain within the legitimate boundary of our range i.e. for testing we consider values like (min, min+, nom, max-, max) whereas in Robustness testing, we try to cross these legitimate boundaries as well.

Q2. What is Test bed and Test data?
A testbed (also spelled test bed) is **a platform for conducting rigorous, transparent, and replicable testing of scientific theories, computational tools, and new technologies**. The term is used across many disciplines to

describe experimental research and new product development platforms and environments.

Test data is **data which has been specifically identified for use in tests, typically of a computer program**. Some data may be used in a confirmatory way, typically to verify that a given set of input to a given function produces some expected result.

Q3. Why does software have bugs?

Bugs in software can arise from mistakes and errors made in interpreting and extracting users' requirements, planning a program's design, writing its source code, and from interaction with humans, hardware and programs, such as operating systems or libraries.

All software has bug**s** because we are human beings and sometimes we do it wrong when it comes to programming. No matter how much care is taken and no matter how many nets are put in place to prevent it from happening, there are so many variables that not all of them can be predicted.

Q4. How do you decide when you have 'tested enough'?

There is no written rule. According to BCS/ISTQB Software Testing Foundation, you cannot physically test for every scenario. When deciding how much testing you should carry out, you may want to consider the level of risk involved, including technical and business risk and even budget or time constraints

1. 100% requirements coverage is achieved.
2. More than 95% of test coverage and 100% functional coverage is achieved.

3. When we achieved the target time.

4. All showstopper and Major defect are identified, verified and closed.

5. Less than 5%Minor defect are open, and if open work around is available.

6. All defects are retested and closed.

7. All corresponding regression scenario of retested and closed defect are also tested.

8. All critical testcases are passed

9. All test document and deliverables are prepared, reviewed and published across.

10. Sign off is given

Q5: Describe the difference between validation and verification

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.

The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

EXPERIMENT - 3

Software Testing and Quality Assurance

Abstract

To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,lOO].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle]. Perform BVA, Equivalence Class Testing (Using Input Domain and Output domain). Which Technique is best for the given problem statement. Give reasons.

Syeda Reeha Quasar

14114802719

EXPERIMENT – 3

Aim:

Equivalence Class Partitioning In the lecture, we used the example of an app that classified Risk Exposure (RE) as High, Moderate, or Low on the basis of Risk Probability (RP) and Risk Impact (RI).

Consider the following specification for such an app:

1) the app accepts two integers, RP and RI, as input,

2) both RP and RI must be in the range [1, 5],

3) if either input is not an integer, it is invalid and the app outputs "Invalid,

4) if either input is an integer outside the range specified in (2), it is invalid and the app outputs "Out of Range",

5) given valid inputs, the app calculates RE as the product of RP and RI, and outputs:

   a. "High", if RE is greater than 9

   b. "Low" if RE is less than or equal to 2

   c. "Moderate" if neither (a) nor (b) are satisfied

i) Partition the domain of each parameter into equivalence classes, label the classes, and list them.

ii) Develop a set of test cases for the app to satisfy Each Choice Coverage of the equivalence classes. Indicate the equivalence classes covered by each test case and, as always, include the expected result. Notice the actual classification process is not adequately tested by your set of test cases.

iii) To better test the classification performed by the app, partition the output domain and develop additional test cases to cover any class not covered by your test cases in (ii).

Theory:

Risk is the probability of a negative or undesirable outcome or event. Risk is any problem that might occur that would decrease customer, user, stakeholder perception of quality and/or project success.

Types of risks:

There are 2 types of risks- Product Risk and Quality Risk.

1. Product risk-
   When the Primary effect of a potential problem is on product quality, then the potential problem is called Product risk. It can also be called a quality risk. Example- A defect that can cause a system crash or monetary loss.

2. Project Risk-
   When the Primary effect of a potential problem is on the overall success of the project, those potential problems are called Project risk. They can also be called Planning risks. Example- Staffing issues that can delay project completion.

Not all risks are equally important. There is a number of ways we can classify the level of risk. The easiest way is to look at two factors-

1. Likelihood of occurrence of the problem. Likelihood arises from technical considerations.
2. Impact of the problem, if it occurs. Impact arises from business considerations.

What Is Risk-Based Testing?

In risk-based testing, we use the risk items identified during risk analysis, along with the level of risk associated with each risk item to guide the testing. It is a type of software testing technique that is primarily based on the probability of the risk. Risk-based testing involves the following steps:

1. Accessing the risk based on software quality.
2. Frequency of use.
3. Criticality of Business.
4. Possible areas with defects, etc.

Characteristics Of Risk-Based Testing (RBT):

Below are some characteristics of Risk-based testing (RBT)-

1. RBT strategy matches the level of test effort to the level of risk. The higher the risk, the more is the test effort. This applies to test execution as well as other test activities like test designing and implementation.
2. It matches the order of testing with the level of risk. Higher risk tests tend to find more defects or are related to more important areas in the application or maybe both. So higher the risk, we plan the tests earlier in the cycle- both in design and execution. This helps in building the confidence of business stakeholders as well.
3. By effort allocation and maintaining the order of testing, the quality risk gets systematically and predictably reduced. By maintaining a traceability matrix of tests vs risks and defects identified to risks, reporting the risk as residual risks make sense. This allows the concerned stakeholders to decide when to stop testing i.e whenever the risk of continuing testing exceeds the risk of testing completion.

4. If schedule reduction requires scope reduction, it is easier to decide what needs to go out. It will always be acceptable and explainable to business stakeholders as risk levels are agreed upon.
5. To identify risks, we need to take inputs from various sources like – Requirements specifications, design specifications, existing application data, previous incident records, etc. However, if this information is not readily available, we can still use this approach by getting inputs from stakeholders for the risk identification and assessment process.

When To Implement Risk-Based Testing?

Risk-based testing approach is implemented in scenarios where-

1. There is time/resource or budget constraints. For example- A hotfix to be deployed in production.
2. When a proof of concept is being implemented.
3. When the team is new to the technology platform or to the application under test.
4. When testing in Incremental models or Iterative models.
5. Security testing is done in Cloud computing environments.

How Risk-Based Testing Is Implemented?

Risk can guide testing in multiple ways but below are the major ones –

1. The effort is allocated by test managers proportional to the risk associated with the test item.
2. Test techniques are selected in a way that matches the rigor and extensiveness required based on the level of risk of the test item.
3. Test activities should be carried out in reverse order of risk i.e The Highest risk item should be tested first.
4. Prioritization and resolution of defects should be done as appropriate with the level of risk associated.
5. During test planning and control, test managers should carry out *risk control* for all significant risk items. The higher the level of risk associated, the more thoroughly it should be controlled.
6. Reporting should be done in terms of residual risks. Example- Which tests have not run yet? Which defects have not fixed yet?

Benefits of Risk-Based Testing (RBT):
By identifying and analyzing the risks related to the system, it is possible to make testing efficient and effective-

1. Efficient-
   RBT is efficient because you test the most critical areas of the system early in the cycle (the earlier the defect is detected the lower is the cost of solving those issues)
2. Effective-
   RBT is effective because your time is spent according to the risk rating and mitigation plans. You do not spend time on items and activities which might not be very important in the overall scheme of things.
3. Reduced Test cases-
   Test case count gets reduced as test cases become more focused.
4. Cost reduction-
   A reduction in cost per quality as critical issues get fixed early in the cycle and hence lowering the cost of change.
5. Faster time to market-
   Faster time to market is more easily achievable with RBT approach as the most important features are available in a shippable position early in the cycle.

Solution:

i)

RP	RI	Equivalence Class
RP < 1	1 <= RI <=5	Invalid
RP > 5	1 <= RI <=5	Invalid
1 <= RP <=5	RI < 1	Invalid
1 <= RP <=5	RI > 5	Invalid
Non Integer	1 <= RI <=5	Invalid
Non Integer	RI < 1	Invalid
Non Integer	RI > 5	Invalid

RP < 1	Non Integer	Invalid
RP > 5	Non Integer	Invalid
1 <= RP <=5	Non Integer	Invalid
1 <= RP <=5	1 <= RI <=5	Valid

ii)

RP	RI	Equivalence Class	Test Case RP	Test Case RI	Expected Output
RP < 1	1 <= RI <=5	Invalid	0	3	Out of Range
RP > 5	1 <= RI <=5	Invalid	7	2	Out of Range
1 <= RP <=5	RI < 1	Invalid	3	-1	Out of Range
1 <= RP <=5	RI > 5	Invalid	4	23	Out of Range
Non Integer	1 <= RI <=5	Invalid	23.4	3	Invalid
Non Integer	RI < 1	Invalid	a	0	Invalid
Non Integer	RI > 5	Invalid	Hello	17	Invalid
RP < 1	Non Integer	Invalid	-45	3.2	Invalid
RP > 5	Non Integer	Invalid	60	xyz	Invalid
1 <= RP <=5	Non Integer	Invalid	2	*	Invalid
1 <= RP <=5	1 <= RI <=5	Valid	2	3	Moderate

iii)

Test Case RP	Test Case RI	Expected Output
4	3	High
1	1	Low
3	2	Moderate

Result:
Problem based on the given conditions was solved and verified.

Viva Questions

Q1. What is risk-based testing?.
Risk-based Testing is the term used for an approach to creating a Test Strategy that is based on prioritizing tests by risk. The basis of the approach is a detailed risk analysis and prioritizing of risks by risk level. Tests to address each risk are then specified, starting with the highest risk first.

Q2. How to perform risk based testing?

1. Make a prioritized list of risks.
2. Perform testing that explores each risk.
3. As risks evaporate and new ones emerge, adjust your test effort to stay focused on the current crop.

Q3. How to improve skills designing test cases and make sure high coverage rate?

Test designing is successful when the requirements are analyzed and understood completely. To ensure 100% test coverage is achieved, you should

not miss out on creating test cases for any requirements and from time to time we can check ourselves with the help of a traceability matrix.

Q4. What's the difference between System testing and Acceptance testing?

Acceptance testing checks the system against the "Requirements." It is similar to System testing in that the whole system is checked but the important difference is the change in focus:

System testing checks that the system that was specified has been delivered.

Acceptance testing checks that the system will deliver what was requested. The customer should always do Acceptance testing and not the developer.

The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgement. This testing is more about ensuring that the software is delivered as defined by the customer. It's like getting a green light from the customer that the software meets expectations and is ready to be used.

Q5: How do you define a testing policy?

The first step any organization needs to do is define one unique definition for testing within the organization so that everyone is of the same mindset.

EXPERIMENT - 4

Software Testing and Quality Assurance

Abstract

Develop a complete limited entry decision table for the following decision situation: An airline offers only flights in Germany and Europe. Under special conditions a discount is offered — a discount with respect to the normal airfare.

Syeda Reeha Quasar

14114802719

EXPERIMENT – 4

Aim:

Develop a complete limited-entry decision table for the following decision situation:

An airline offers only flights in Germany and Europe. Under special conditions a discount is offered — a discount with respect to the normal airfare.

Rules:

- Passengers older than 18 with destinations in Germany are offered a discount at 20%, if the departure is not on a Monday or Friday- If the passengers stay at least 6 days at the destination, an additional discount of 10% is offered.
- For destinations outside Of Germany passengers are offered a discount Of 25%, if the departure is not on a Monday or Friday.
- Passengers older than 2 but younger than 18 years are offered a discount of 40% for all destinations.
- Children under 2 travel for free.

For each rule, design the test case

Theory:

A decision table is a brief visual representation for specifying which actions to perform depending on given conditions. The information represented in decision tables can also be represented as decision trees or in a programming language using if-then-else and switch-case statements.

A decision table is a good way to settle with different combination inputs with their corresponding outputs and is also called a cause-effect table. The reason to call cause-effect table is a related logical diagramming technique called cause-effect graphing that is basically used to obtain the decision table.

Importance of Decision Table:

- Decision tables are very much helpful in test design techniques.
- It helps testers to search the effects of combinations of different inputs and other software states that must correctly implement business rules.
- It provides a regular way of starting complex business rules, that is helpful for developers as well as for testers.
- It assists in the development process with the developer to do a better job. Testing with all combinations might be impractical.

- A decision table is basically an outstanding technique used in both testing and requirements management.
- It is a structured exercise to prepare requirements when dealing with complex business rules.
- It is also used in model complicated logic.

Advantages of Decision Table:

- Any complex business flow can be easily converted into test scenarios & test cases using this technique.
- Decision tables work iteratively which means the table created at the first iteration is used as input tables for the next tables. The iteration is done only if the initial table is not satisfactory.
- Simple to understand and everyone can use this method to design the test scenarios & test cases.
- It provides complete coverage of test cases which helps to reduce the rework on writing test scenarios & test cases.
- These tables guarantee that we consider every possible combination of condition values. This is known as its completeness property.

Solution:

Airline Passenger Discount Policy

An airline offers only flights to India and Asia. Under special conditions, a discount is offered on the normal airfare:

- Passengers older than 18 with destinations in India are offered a discount of 20%, as long as the departure is not on a Monday or Friday.
- For destinations outside of India, passengers are offered a discount of 25%, if the departure is not on a Monday or Friday.
- Passengers who stay at least 6 days at their destination receive an additional discount of 10%.
- Passengers older than 2 but younger than 18 years are offered a discount of 40% for all destinations.
- Children 2 and under travel for free.

Extracting Rules

Conditions:

- Destination (India, Asia)
- Passenger Age (<= 2, > 2 && < 18, > 18
- Depart on Monday or Friday (Yes, No)
- Stay 6 days or more (Yes, No)

Actions:

- Travel Free
- 0% discount
- 10% discount
- 20% discount
- 40% discount

Number of rules: 2 values * 3 values * 2 values * 2 values = 24 rules

Scenarios

Airline Discount Policy	Rules																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Destination:	I	I	I	I	I	I	I	I	I	I	I	I	A	A	A	A	A	A	A	A	A	A	A	A
Passenger Age:	<=2	<=2	<=2	<=2	3-18	3-18	3-18	3-18	>18	>18	>18	>18	<=2	<=2	<=2	<=2	3-18	3-18	3-18	3-18	>18	>18	>18	>18
Depart Mon/Fri?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Stay >= 6 Days?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
No Discount										X												X		
10% Discount	X		X		X		X		X		X		X		X		X		X		X		X	
20% Discount											X	X												
25% Discount															X	X			X	X			X	X
40% Discount					X	X	X	X									X	X	X	X				
Travel Free	X	X	X	X									X	X	X	X								

Reduced Table

Airline Discount Policy	Rules												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Destination:	--	I	I	--	--	I	I	A	A	A	A	A	A
Passenger Age:	<=2	3-18	3-18	>18	>18	>18	>18	3-18	3-18	3-18	3-18	>18	>18
Depart Mon/Fri?	--	--	--	Y	Y	N	N	Y	Y	N	N	N	N
Stay >= 6 Days?	--	Y	N	N	Y	Y	N	Y	N	Y	N	Y	N
No Discount				X									
10% Discount			X		X	X		X		X		X	
20% Discount						X	X						
25% Discount										X	X	X	X
40% Discount		X	X					X	X	X	X		
Travel Free	X												

Decision Table:

	Destination	Age	IsWeekDay	MoreThan6	Discount
Rule 1		Group1			100
Rule 2	India	Group2		Y	40
Rule 3	India	Group2		N	50
Rule 4		Group3	Y	N	0
Rule 5		Group3	Y	Y	10
Rule 6	India	Group3	N	Y	30
Rule 7	India	Group3	N	N	20
Rule 8	Asia	Group2	Y	Y	50
Rule 9	Asia	Group2	Y	N	40
Rule 10	Asia	Group2	N	Y	75
Rule 11	Asia	Group2	N	N	65
Rule 12	Asia	Group3	N	Y	35
Rule 13	Asia	Group3	N	N	25

Result:

For each rule test-case was designed and verified.

Viva Questions

Q1. Why we use decision tables?

The techniques of equivalence partitioning and boundary value analysis are often applied to specific situations or inputs. However, if different combinations of inputs result in different actions being taken, this can be more difficult to

show using equivalence partitioning and boundary value analysis, which tend to be more focused on the user interface. The other two specification-based techniques, decision tables and state transition testing are more focused on business logic or business rules. A decision table is a good way to deal with combinations of things (e.g. inputs). This technique is sometimes also referred to as a 'cause-effect' table. The reason for this is that there is an associated logic diagramming technique called 'cause-effect graphing' which was sometimes used to help derive the decision table.

Q2 What are the disadvantages of Decision Table testing?

The main disadvantage is that when the number of inputs increases the table will become more complex.

Q3. Why Decision Table Testing is Important?

Decision Table Testing is Important because it helps to test different combinations of conditions and provides better test coverage for complex business logic. When testing the behavior of a large set of inputs where system behavior differs with each set of inputs, decision table testing provides good coverage and the representation is simple so it is easy to interpret and use.

In Software Engineering, boundary value and equivalent partition are other similar techniques used to ensure better coverage. They are used if the system shows the same behavior for a large set of inputs. However, in a system where for each set of input values the system behavior is different, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage.In this case, decision table testing is a good option.

This technique can make sure of good coverage, and the representation is simple so that it is easy to interpret and use.

This table can be used as the reference for the requirement and for functionality development since it is easy to understand and cover all the combinations.

Q4. What are the advantages of Decision Table Testing?

1. When the system behavior is different for different inputs and not the same for a range of inputs, both equivalent partitioning, and boundary value analysis won't help, but a decision table can be used.
2. The representation is simple so that it can be easily interpreted and is used for development and business as well.
3. This table will help to make effective combinations and can ensure better coverage for testing
4. Any complex business conditions can be easily turned into decision tables
5. In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

Q5: Why is Decision Table also called a Cause-Effect table?

Decision table testing is a software testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage.