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:

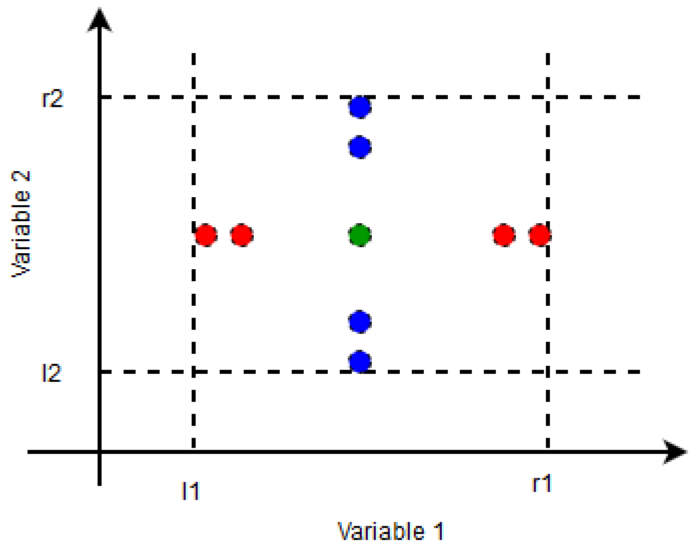
To determine the type of triangle. Its input is triple of +ve integers (say x,y,z) and the values may be from interval[1,IOO].The program output may be one of the following [Scalene, Isosceles, Equilateral, Not a Triangle]. Perform BVA, Equivalence Class Testing (Using Input Domain and Output domain).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

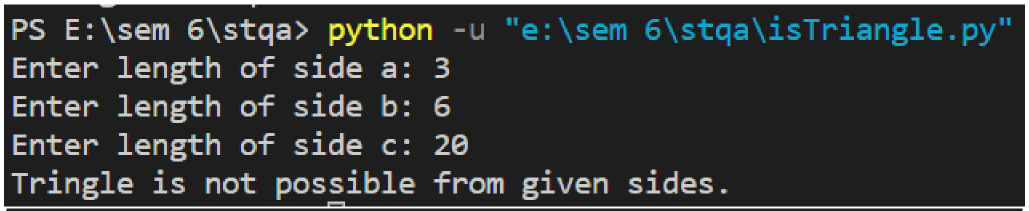
The table shows the Test Cases Design for the Triangle Problem.The range value [l, r] is taken as [1, 100] and nominal value is taken as 50.

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

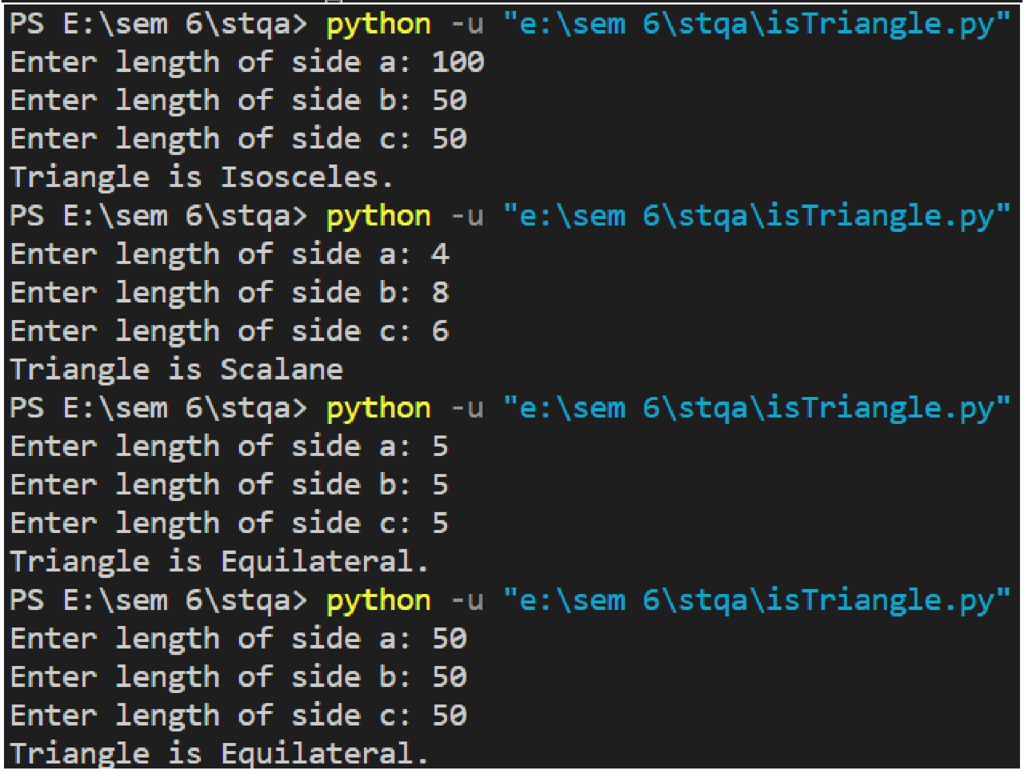
Source Code:

```
# 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.