Internal Assesment Test – II, November 2020									
Sub:	Software Testing							Code:	18MCA351
Date:	6-11-2020	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA

	Answer Any 5 **QUESTION**s	Marks	OBE	
			CO	RBT
1	What do you understand by a pseudo code and List same generalized pseudo code and explain with a neat diagram the currency converter and Saturn wind shield wiper controller?	10	CO2	L2
2	Explain triangle problem with conditions all specification and structured implementation and traditional implementation.	10	CO2	L1
3 a	What do you mean by Boundary value Analysis (BVA) and explain the same with diagrams?	06	CO2	L2
3b	Explain equivalence class testing with relevant diagrams.	04	CO2	L2
4a	List the types of Boundary value analysis and Equivalence class test.	04	CO3	L1
4b	Demonstrate SATM (Simple Automated Teller Machine) with diagrams/figures.	06	CO3	L3
5	Give BVA test cases for Next Date problem with conditions. Discusses the same. Introduce Robustness and discuses.	10	CO3	L3
6	Give set of test cases for commission problem based on output boundary value analysis with conditions and discuses.	10	CO3	L2
7	Discuss the usage of decision table method to device test cases with example of commission problem and triangle problem.	10	CO3	L4
8	Justify the usage of boundary value analysis with function of two variables and highlight the limitations of BVA.	10	CO3	L3

Note: If you answer for Q3a and Q3b also should answer and the same for Q4a and Q4b

1. What do you understand by a pseudo code and List same generalized pseudo code and explain with a neat diagram the currency converter and Saturn wind shield wiper controller?

What do you understand by a pseudo code?

- Pseudocode provides a "**language neutral**" way to express logic.
- Program source code Units can be interpreted either as traditional components (**procedures and functions**) or as object-oriented components (**classes and objects**.)
- Terms such as expression, variable list, and field description are used with no formal definition

List same generalized pseudo code

Language Element	*Generalized Pseudocode Construct*
Comment	' <text>
Data structure declaration	Type <type name> <list of field descriptions> End <type name>
Data declaration	Dim <variable> As <type>
Assignment statement	<variable> = <expression>
Input	Input (<variable list>)
Output	Output (<variable list>)
Condition	<expression> <relational operator> <expression>
Compound condition	<Condition> <logical connective> <Condition>
Sequence	statements in sequential order

Simple selection	If <condition> Then
	<then clause>
	EndIf

Selection	If <condition> Then <then clause> Else <else clause> EndIf
Multiple selection	Case <variable> Of Case 1: <predicate> <Case clause> … Case *n*: <predicate> <Case clause> EndCase
Counter-controlled repetition	For <counter> = <start> To <end> <loop body> EndFor
Pretest repetition	While <condition> <loop body> EndWhile

Language Element	*Generalized Pseudocode Construct*
Posttest repetition	Do <loop body> Until <condition>
Procedure definition (similarly for functions and o–o methods)	<procedure name> (Input: <list of variables>; Output: <list of variables>) <body> End <procedure name>
Interunit communication	Call <procedure name> (<list of variables>; <list of variables>)
Class/Object definition	<name> (<attribute list>; <method list>, <body>) End <name>
Interunit communication	msg <destination object name>.<method name> (<list of variables>)
Object creation	Instantiate <class name>.<object name> (list of attribute values)
Object destruction	Delete <class name>.<object name>
Program	Program <program name> <unit list> End<program name>

Currency Convertor

1The purpose of the two programs - This page will focus on the purpose of the content I added onto Python and Small Basic.

2. **The quality of the program** - On this page I will be focusing on the quality of the coding, and the task performance of Python and Small Basic.

3. **Strengths and weaknesses** - This page will focus on the strengths and weaknesses of Python and Small Basic.

4. **Problem definition of the currency converter** - On this page I will state the task that I have been given and what I plan to include in my program.

5. **Design documents for the program** - On this page I will be presenting the stages of how I designed my currency converter, this will include the layout of my program and the coding.

6. **The user interface** - This page will focus on

7. **The code for the program** - This page will inform viewers of how I enabled the features of my program to operate.
8. **The test plan** - This page will include tables that contain information on whether my program operates as I expect it to.
9.1 **Usability tests and feedback** - This page consists of testing conversions on my program and receiving feedback from random users.
9.2 **Refine and improve based on feedback** - This page focuses on how I improved upon the aspects of my program that were criticized.
9.3 **Review of my program** - This page will be an overview on the content and performance of my program.
9.4 **Evaluate the final software** - This will be an evaluation on the final stage of my program, after all improvements that I have made.

Windshield wiper

Saturn Windshield Wiper Controller

The windshield wiper on some Saturn automobiles is controlled by a lever with a dial.
The lever has four positions: **OFF, INT** (for intermittent), **LOW,** and **HIGH**; and the dial has three positions, numbered simply 1, 2, and 3.
The dial positions indicate three intermittent speeds, and the dial position is relevant only when the lever is at the INT position. The decision table below shows the windshield wiper speeds (in wipes per minute) for the lever and dial positions.

c1. Lever	OFF	INT	INT	INT	LOW	HIGH
c2. Dial	n/a	1	2	3	n/a	n/a
a1. Wiper	0	4	6	12	30	60

2. Explain triangle problem with conditions all specification and structured implementation and traditional implementation.

Problem Statement

The triangle program accepts three integers, a, b, and c, as input. These are taken to be sides of a triangle. The output of the program is the type of triangle determined by the three sides: *Equilateral, Isosceles, Scalene, or Not a Triangle*.

The triangle program accepts three integers a, b and c must satisfy the following conditions:
c1. $1 \leq a \leq 200$ c4. $a < b + c$
c2. $1 \leq b \leq 200$ c5. $b < a + c$
c3. $1 \leq c \leq 200$ c6. $c < a + b$
The output of the program is the type of triangle determined by the three sides: Equilateral, Isosceles, scalene, or Not A Triangle. If an input value fails any of conditions c1, c2 or c3, then program outputs message as, for example, “Vale of b is not in the range of permitted values”. If values of a, b, and c satisfy conditions c1, c2, and c3, one of four mutually exclusive outputs is given:

1. If all three sides are equal, the program output is Equilateral.
2. If exactly one pair of sides is equal, the program output is Isosceles.
3. If no pair of sides is equal, the program output is Scalene.
4. If any of conditions c4, c5 and c6 is not met, the program output is Not A Triangle.

```
        Dim a, b, c As Integer
        Dim c1, c2, c3, IsATriangle As Boolean
        'Step 1: Get Input
        Do
            Output("Enter 3 integers which are sides of a triangle")
            Input(a, b, c)
            c1 = (1 ≤ a) AND (a ≤ 300)
            c2 = (1 ≤ b) AND (b ≤ 300)
            c3 = (1 ≤ c) AND (c ≤ 300)
            If NOT(c1)
                 Then Output("Value of a is not in the range of permitted values")
            EndIf
            If NOT(c2)
                 Then Output("Value of b is not in the range of permitted values")
            EndIf
            If NOT(c3)
                 ThenOutput("Value of c is not in the range of permitted values")
            EndIf
        Until c1 AND c2 AND c3
        Output("Side A is",a)
        Output("Side B is",b)
        Output("Side C is",c)
        'Step 2: Is A Triangle?
        If (a < b + c) AND (b < a + c) AND (c < a + b)
            Then IsATriangle = True
EndIf
'Step 3: Determine Triangle Type
If IsATriangle
    Then If (a = b) AND (b = c)
            Then Output ("Equilateral")
            Else If (a ≠ b) AND (a ≠ c) AND (b ≠ c)
                    Then Output ("Scalene")
                    Else Output ("Isosceles")
                 EndIf
         EndIf
    Else Output("Not a Triangle")
EndIf
End triangle3
```

3. A. What do you mean by Boundary value Analysis (BVA) and explain the same with diagrams?

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values. So these extreme ends like

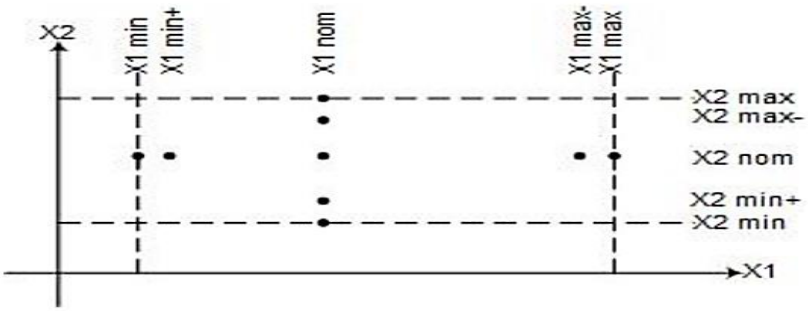
- Start- End,
- Lower- Upper,
- Maximum-Minimum,
- Just Inside-Just Outside values

are called boundary values and the testing is called "boundary testing".

The basic idea in boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum

The values used to test the extremities are	
Min	Minimal
Min+	Just above Minimal
Nom	Average/Nominal/normal
Max-	Just below Maximum
Max	Maximum



Keep X_1 at normal value and vary X_2	Keep X_2 at normal value and vary X_1
$<X_{1nom},X_{2min}>,<X_{1nom},X_{2min+}>,$ $<X_{1nom},X_{2nom}>,<X_{1nom},X_{2max-}>,<X_{1nom},X_{2max}>$	$<X_{1min},X_{2nom}>,<X_{1min+},X_{2nom}>,<X_{1max-},$ $X_{2nom}>,<X_{1max},X_{2nom}>$

3.B. Explain equivalence class testing with relevant diagrams.

- The key point in equivalence class testing is to choose the equivalence relation that determines the classes.
- We need the function that we have used in boundary value testing for the sake of comprehensible drawings, which relates to a function F of two variables **x_1 and x_2.**
- When F is implemented as a program, the input variables **x_1 and x_2** will have the following boundaries, and intervals within the boundaries:

  $a \leq x_1 \leq d$, with intervals **[a, b) , [b, c) , [c, d]**

  $e \leq x_2 \leq g$, with intervals **[e, f), [f, g]**
- Where square brackets and parentheses denote, respectively, closed and open interval endpoints. The intervals presumably correspond to some distinction in the program being tested.

Equivalence Partitions analysis

Example: Requirement for 'Password' field from "Add users modal window" of Admins actions menu: password field can not be shorter than 4 and longer than 28 (including) characters (numeric and alphabetic)

Equivalance classes

2	15	35
< 4	between 4 and 28	>28
invalid	valid	invalid

Define and execute the test cases:

1. Password field contain 2 characters – Fail;
2. Password field contain 15 characters – Pass;
3. Password field contain 35 characters – Fail.

4. A. List the types of Boundary value analysis and Equivalence class test.

There are four variations of boundary value testing:

- Normal boundary value testing
- Robust boundary value testing
- Worst-case boundary value testing
- Robust worst-case boundary value testing

Equivalence Partitioning also called as equivalence class partitioning. It is abbreviated as ECP. It is a software testing technique that divides the input test data of the application under test into each partition at least once of equivalent data from which test cases can be derived. An advantage of this approach is it reduces the time required for performing testing of a software due to a smaller number of test cases.

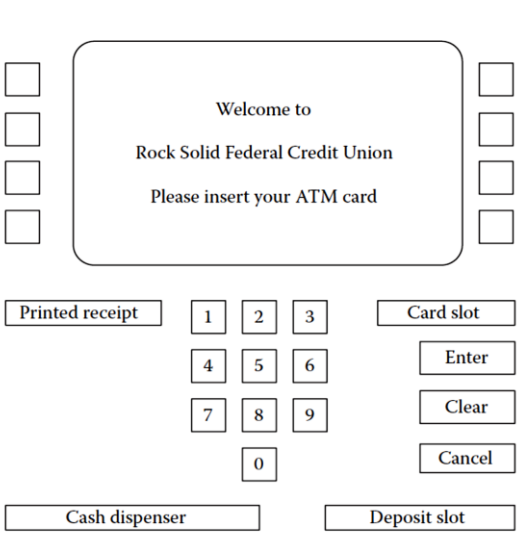
The equivalence class testing can be categorized into four different types, which are integral part of testing and cater to different data set.

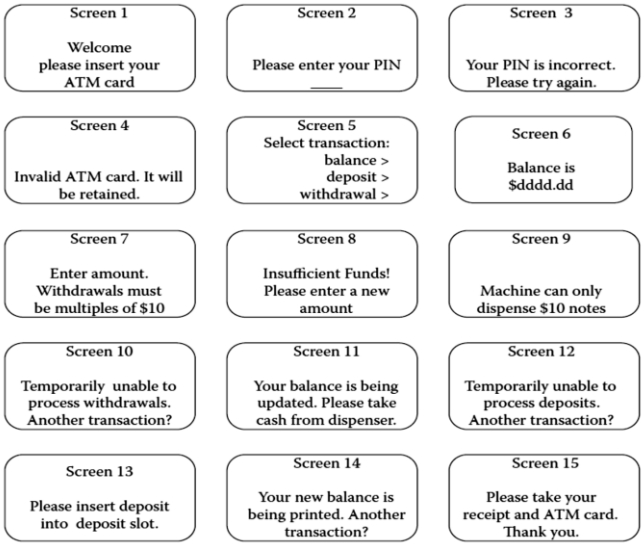
- Weak Normal Equivalence Class Testing
- Strong Normal Equivalence Class Testing
- Weak Robust Equivalence Class Testing
- Strong Robust Equivalence Class Testing

4.B. Demonstrate SATM (Simple Automated Teller Machine) with diagrams/figures.

Problem Statement

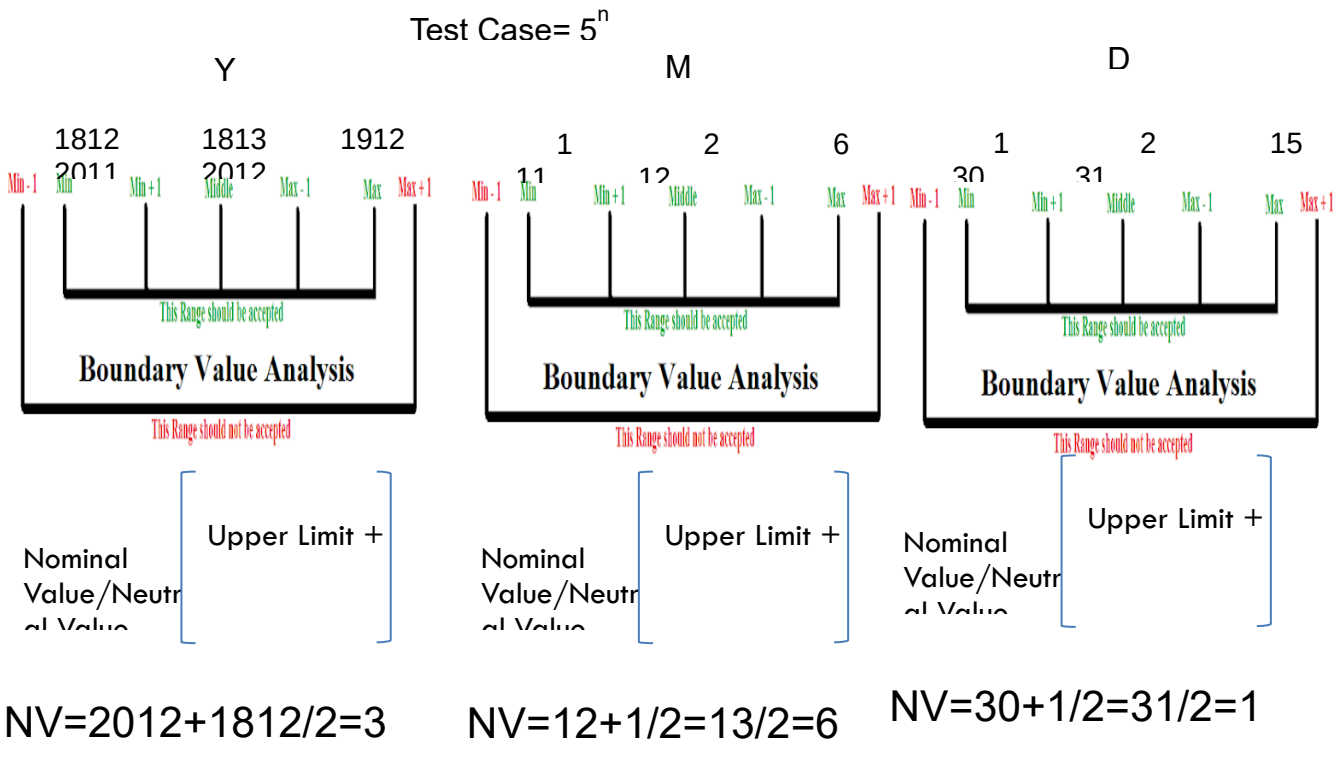
The SATM system communicates with bank customers via the 15 screens shown in Figure Using a terminal with features as shown in Figure SATM customers can select any of three transaction types: deposits, withdrawals, and balance inquiries. For simplicity, these transactions can only be done on a checking account.





5. Give BVA test cases for Next Date problem with conditions. Discusses the same. Introduce Robustness and discuses.

   All 125 worst-case test cases for NextDate are listed in Table. Take some time to examine it for gaps of untested functionality and for redundant testing. For example, would anyone actually want to test January 1 in five different years? Is the end of February tested sufficiently?



Case	Month	Day	Year	Expected Output
1	1	1	1812	1, 2, 1812
2	1	1	1813	1, 2, 1813
3	1	1	1912	1, 2, 1912
4	1	1	2011	1, 2, 2011
5	1	1	2012	1, 2, 2012
6	1	2	1812	1, 3, 1812
7	1	2	1813	1, 3, 1813
8	1	2	1912	1, 3, 1912
9	1	2	2011	1, 3, 2011
10	1	2	2012	1, 3, 2012
11	1	15	1812	1, 16, 1812
12	1	15	1813	1, 16, 1813
13	1	15	1912	1, 16, 1912
14	1	15	2011	1, 16, 2011
15	1	15	2012	1, 16, 2012
16	1	30	1812	1, 31, 1812
17	1	30	1813	1, 31, 1813
18	1	30	1912	1, 31, 1912
19	1	30	2011	1, 31, 2011
20	1	30	2012	1, 31, 2012
21	1	31	1812	2, 1, 1812
22	1	31	1813	2, 1, 1813
23	1	31	1912	2, 1, 1912
24	1	31	2011	2, 1, 2011
25	1	31	2012	2, 1, 2012
26	2	1	1812	2, 2, 1812
27	2	1	1813	2, 2, 1813
28	2	1	1912	2, 2, 1912

6. Give set of test cases for commission problem based on output boundary value analysis with conditions and discuses.

A rifle salesperson in the former Arizona Territory sold

- **Items:** ***rifle locks***, ***stocks***, *and* ***barrels*** made by a gunsmith in Missouri.
- **Price:** Locks cost $45, stocks cost $30, and barrels cost $25.
- **Sales:** The salesperson had to sell at least one lock, one stock, and one barrel (but not necessarily one complete rifle) per month, and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 stocks, and 90 barrels.
- **Commission:** The gunsmith then knew the sales for the month were complete and computed the salesperson's commission as follows: 10% on sales up to (and including) $1000, 15% on the next $800, and 20% on any sales in excess of $1800.

This problem separates into three distinct pieces:
the input data portion, in which we could deal with input data validation the
sales calculation, and
the commission calculation portion.

Items	Price per item in $	Total Item Sold	Total Amount in $	Rubrics for Commission 10% on sales up to $1000, 15% on the next $800, and 20% on any sales in excess of $1800
rifle locks	$45	70	3150	100+120+270 = 490
stocks	$30	80	2400	100+120+120 = 340
barrels	$25	90	2250	100+120+090 = 310

```
Program Commission (INPUT,OUTPUT)
'Dim locks, stocks, barrels As Integer
Dim lockPrice, stockPrice, barrelPrice As Real
Dim totalLocks,totalStocks,totalBarrels As Integer
Dim lockSales, stockSales, barrelSales As Real
Dim sales,commission : REAL
'lockPrice = 45.0stock
Price = 30.0
barrelPrice = 25.0
totalLocks = 0
totalStocks = 0
totalBarrels = 0
Input(locks)
While NOT(locks = -1) 'Input device uses -1 to
indicate end of data
Input(stocks, barrels)
totalLocks = totalLocks + locks
totalStocks = totalStocks + stocks
totalBarrels = totalBarrels + barrels
Input(locks)
EndWhile
Output("Locks sold:", totalLocks)
Output("Stocks sold:", totalStocks)
Output("Barrels sold:", totalBarrels)
'
lockSales = lockPrice * totalLocks
stockSales = stockPrice * totalStocks
barrelSales = barrelPrice * totalBarrels
sales = lockSales + stockSales + barrelSales
Output("Total sales:", sales)
'
If (sales > 1800.0)
Then
commission = 0.10 * 1000.0
commission = commission + 0.15 * 800.0
commission = commission + 0.20 * (sales-1800.0)
Else If (sales > 1000.0)
Then
commission = 0.10 * 1000.0
commission = commission + 0.15*(sales-1000.0)
Else commission = 0.10 * sales
EndIf
EndIf
Output("Commission is $",commission)
End Commission
```

7. Discuss the usage of decision table method to device test cases with example of commission problem and triangle problem.

- A decision table consists of a number of columns (rules) that comprise all test situations
- Example: the triangle problem
  - C1: a, b,c form a triangle
  - C2: a=b
  - C3: a= c
  - C4: b= c
  - A1: Not a triangle
  - A2:scalene
  - A3: Isosceles
  - A4:equilateral
  - A5: Right angled

Sample Decision table

	r1	r2	...				rn
C1	0	1					0
c2	-	1					0
C3	-	1					1
C4	-	1					0
a1	1	0					0
a2	0	0					1
a3	0	0					0
a4	0	1					0
a5	0	0					1

More Complete Decision Table for the Triangle Problem

Conditions										
C1: a < b+c?	F	T	T	T	T	T	T	T	T	T
C2: b < a+c?	-	F	T	T	T	T	T	T	T	T
C3: c < a+b?	-	-	F	T	T	T	T	T	T	T
C4: a=b?	-	-	-	T	T	T	T	F	F	F
C5: a=c?	-	-	-	T	T	F	F	T	T	F
C6: b=c?	-	-	-	T	F	T	F	T	F	T
Actions										
A1: Not a Triangle	X	X	X							
A2: Scalene										X
A3: Isosceles							X		X	X
A4: Equilateral				X						
A5: Impossible					X	X		X		

Test Cases for the Triangle Problem

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

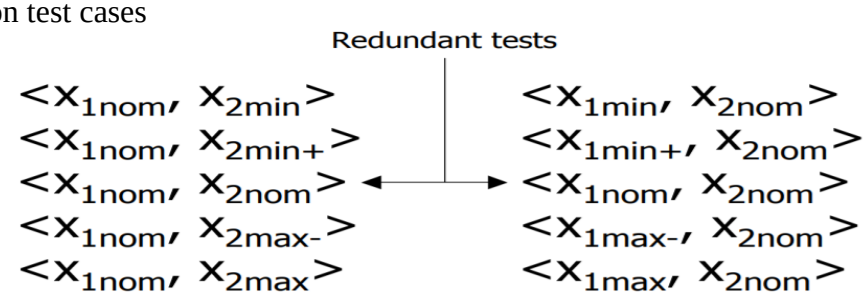
Case Id	Description	Input Data			Expected Output	Actual Output	Status
		a	b	c			
1	Enter the value of a, b and c such that a is less than sum of two sides	4	1	2	Message should be displayed can't form a triangle		
2	Enter the value of a, b and c such that b is less than sum of two sides and a is less than sum of other two sides	1	4	2	Message should be displayed can't form a triangle		
3	Enter the value of a, b and c such that c is less than sum of two sides and a and b is less than sum of other two sides	1	2	4	Message should be displayed can't form a triangle		
4	Enter the value a, b and c satisfying precondition and a=b, b=c and c=a	5	5	5	Should display the message Equilateral triangle		
5	Enter the value a ,b and c satisfying precondition and a=b and b ≠ c	2	2	3	Should display the message Isosceles triangle		
6	Enter the value a ,b and c satisfying precondition and b=c and c ≠ a	3	2	2	Should display the message Isosceles triangle		
7	Enter the value a ,b and c satisfying precondition and c=a and a ≠ b	2	3	2	Should display the message Isosceles triangle		
8	Enter the value a, b and c satisfying precondition and a ≠b , b ≠ c and c ≠ a	3	4	5	Should display the message Scalene triangle		

Test Cases for the Commission Problem

The commission problem is not well served by a decision table analysis. This is not surprising because very little decisional logic is used in the problem. Because the variables in the equivalence classes are truly independent, no impossible rules will occur in a decision table in which conditions correspond to the equivalence classes. Thus, we will have the same test cases as we did for equivalence class testing.

8. Justify the usage of boundary value analysis with function of two variables and highlight the limitations of BVA.

Two-variable function test cases



Apply BVA to the Triangle problem

$1 \leq a \leq 200$
$1 \leq b \leq 200$
$1 \leq c \leq 200$

Limitations of BVA

Boundary Value Analysis works well when the Program Under Test (PUT) is a "function of several independent variables that represent bounded physical quantities"

When these conditions are met BVA works well but when they are not we can find deficiencies in the results. For example, the NextDate problem, where Boundary Value Analysis would place an even testing regime equally over the range, tester's intuition and common sense shows that we require more emphasis towards the end of February or on leap years.

The reason for this poor performance is that BVA cannot compensate or take into consideration the nature of a function or the dependencies between its variables. This lack of intuition or understanding for the variable nature means that BVA can be seen as quite rudimentary.