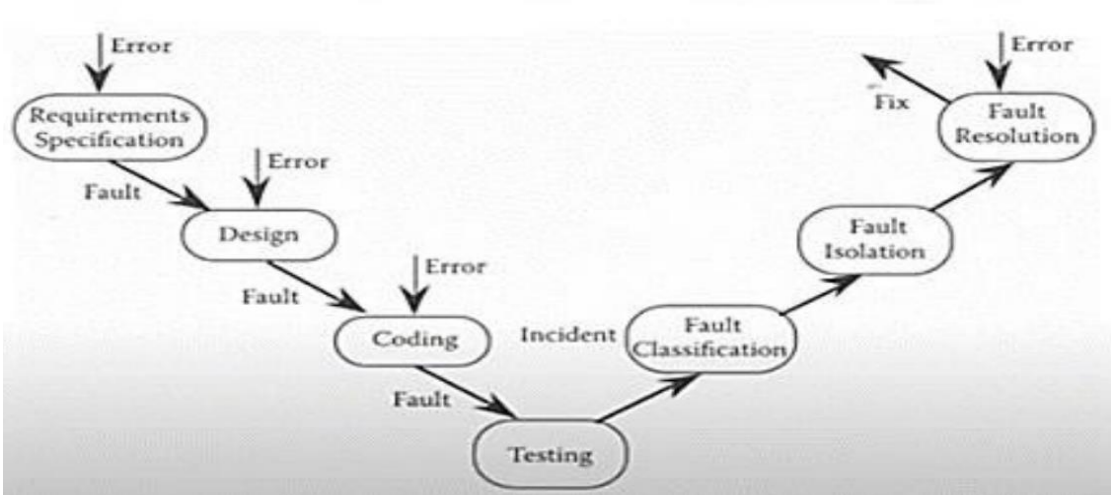
USN										

Internal Assessment Test 1 – JUN 2024

Sub:	**SOFTWARE TESTING**					Sub Code:	**21IS63**	Branch:	**ISE**
Date:	**05/06/2024**	Duration:	**90 min**	Max Marks:	**50**	Sem/Sec:	**VI/ A, B & C**		**OBE**

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1.	a. **Explain the terms: i) Error ii) fault iii) failure iv) incident v) test vi) test case along with testing life cycle.** Scheme: Definition of each term carries 3 marks and life cycle 3 marks. Solution: Error: People make errors. A good synonym for error is ―mistake. When people make mistakes while coding, we call these mistakes ―bugs. Fault: A fault is the result of an error. It is more precise to say that a fault is the representation of an error, where representation is the mode of expression, such as narrative text, dataflow diagrams, hierarchy charts, source code, and so on. Failure: A failure occurs when a faulty code executes. Test: Testing is obviously concerned with errors, faults, failures, and incidents. A test is the act of exercising software with test cases. Test Case: The essence of software testing is to determine a set of test cases for the item to be tested. A test case is (or should be) a recognized work product.  	[6]	1	L2
	b. **Differentiate between black box testing and white box testing.** Scheme: Differentiation of each term carries 2+2 marks. Solution: **Black box testing** Program-a function that maps values from its input domain to values in its output range . Content/implementation is not known. Function is understood completely in terms of its inputs & outputs. For test case identification only specification of the software is used. **White box testing.** Implementation is known and used to identify test cases. Concept of linear graph theory is required to understand. Test coverage metrics –provides way to state the extent to which the software item can be tested.	[4]		

2.	a. **Apply strong and weak-robust techniques of Equivalence class testing in generating test cases for NextDate( ) Function.** Scheme: Problem Statement +Test Case Table of each technique carries 3+3 marks. Solution: **Strong Robust Equivalence Class Test Cases** b. **Generate Decision table for the triangle problem.** Scheme: Problem Statement +Decision Table carries 1+2 marks. Solution:	[7] [3]	2	L3

Case ID	*Month*	*Day*	*Year*	*Expected Output*
WR1	6	15	1912	6/16/1912
WR2	–1	15	1912	Value of month not in the range 1 ... 12
WR3	13	15	1912	Value of month not in the range 1 ... 12
WR4	6	–1	1912	Value of day not in the range 1 ... 31
WR5	6	32	1912	Value of day not in the range 1 ... 31
WR6	6	15	1811	Value of year not in the range 1812 ... 2012
WR7	6	15	2013	Value of year not in the range 1812 ... 2012

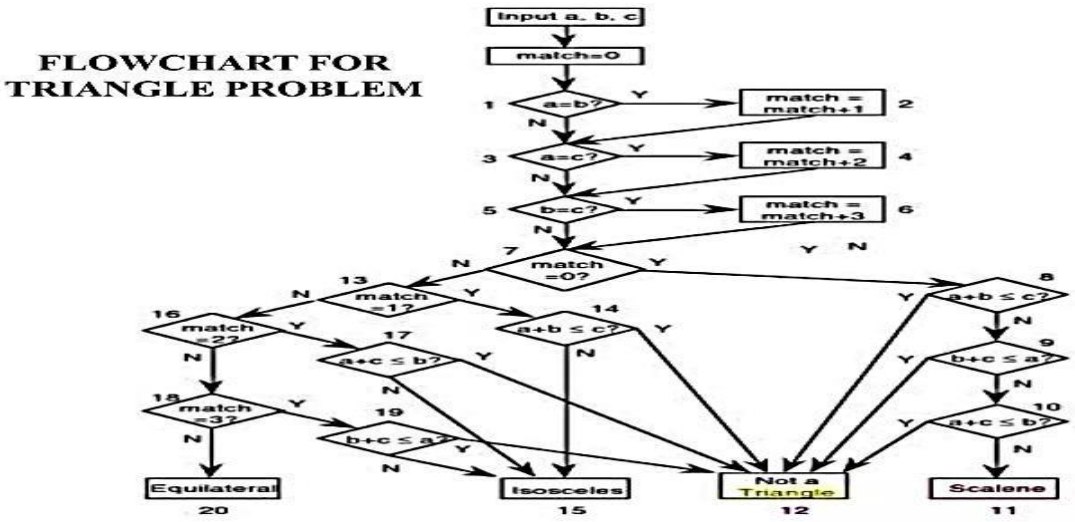
Case ID	*Month*	*Day*	*Year*	*Expected Output*
SR1	–1	15	1912	Value of month not in the range 1 ... 12
SR2	6	–1	1912	Value of day not in the range 1 ... 31
SR3	6	15	1811	Value of year not in the range 1812 ... 2012
SR4	–1	–1	1912	Value of month not in the range 1 ... 12 Value of day not in the range 1 ... 31
SR5	6	–1	1811	Value of day not in the range 1 ... 31 Value of year not in the range 1812 ... 2012
SR6	–1	15	1811	Value of month not in the range 1 ... 12 Value of year not in the range 1812 ... 2012
SR7	–1	–1	1811	Value of month not in the range 1 ... 12 Value of day not in the range 1 ... 31 Value of year not in the range 1812 ... 2012

	1	2	3	4	5	6	7	8	9	10	11
c1: a < b + c?	F	T	T	T	T	T	T	T	T	T	T
c2: b < a + c?	–	F	T	T	T	T	T	T	T	T	T
c3: c < a + b?	–	–	F	T	T	T	T	T	T	T	T
c4: a = b?	–	–	–	T	T	T	T	(F)	F	F	F
c5: a = c?	–	–	–	T	T	(F)	F	T	T	F	F
c6: b = c?	–	–	–	T	(F)	T	F	T	F	T	F
a1: Not a triangle	X	X	X								
a2: Scalene											X
a3: Isosceles							X		X	X	
a4: Equilateral				X							
a5: Impossible					(X)	(X)		(X)			

3.	**Explain the Traditional and Structural Implementation of triangle problem statement using Fortran Style along with its flowchart.** Scheme: Implementation of Traditional & Structural approach, Flowchart carries 4+3+4 marks. Solution:	[10]	1	L2

```
Dim a, b, c, match As INTEGER
Output("Enter 3 integers which are sides of a triangle")
Input(a,b,c)
Output("Side A is ",a)
Output("Side B is ",b)
Output("Side C is ",c)
match = 0
If a = b                                                              '(1)
    Then  match = match + 1                                           '(2)
EndIf
If a = c                                                              '(3)
    Then  match = match + 2                                           '(4)
EndIf
If b = c                                                              '(5)
    Then  match = match + 3                                           '(6)
EndIf
If match = 0                                                          '(7)
    Then    If (a+b)<=c                                               '(8)
                Then    Output("NotATriangle")                        '(12.1)
                Else    If (b+c)<=a                                   '(9)
                            Then    Output("NotATriangle")            '(12.2)
                            Else    If (a+c)<=b                       '(10)
                                        Then    Output("NotATriangle")  '(12.3)
                                        Else    Output ("Scalene")      '(11)
                                    EndIf
                        EndIf
            EndIf
    Else    If match=1                                                '(13)
                Then    If (a+c)<=b                                   '(14)
                            Then    Output("NotATriangle")            '(12.4)
                            Else    Output ("Isosceles")              '(15.1)
                        EndIf
                Else    If match=2                                    '(16)
                            Then    If (a+c)<=b
                                        Then    Output("NotATriangle")  '(12.5)
                                        Else    Output ("Isosceles")    '(15.2)
                                    EndIf

                Else    If match=3                                    '(18)
                            Then    If (b+c)<=a                       '(19)
                                        Then    Output("NotATriangle")  '(12.6)
                                        Else    Output ("Isosceles")    '(15.3)
                                    EndIf
                            Else    Output ("Equilateral")            '(20)
                        EndIf
            EndIf
EndIf
```




```
Dim a,b,c As Integer
Dim IsATriangle As Boolean
'
'Step 1 : Get Input
Output("Enter 3 integers which are sides of a triangle")
Input(a,b,c)
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
    Else IsATriangle = False
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 triangle2
```

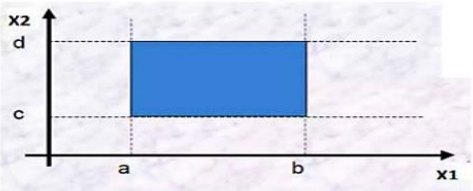
4.	**Explain Boundary Value Analysis (BVA) and generate commission problem test cases for the same.** Scheme: Explanation of BVA + Test cases carries 5+5 marks. Solution:	[10]	2	L2

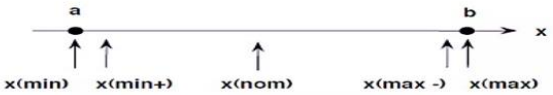
- **Boundary Value Analysis** is a black box test design technique where test case are designed by using boundary values

- Boundary value analysis (BVA) is based on testing at the boundaries between partitions

- Basic idea of BVA is to use input variable values at their: **blue shaded region – input domain space**
  - Minimum (min)
  - Above Minimum (min+)
  - Nominal Value (nom) (Average Value)
  - Below Maximum (max-)
  - Maximum (max)

$a <= x_1 <= b$
$c <= x_2 <= d$

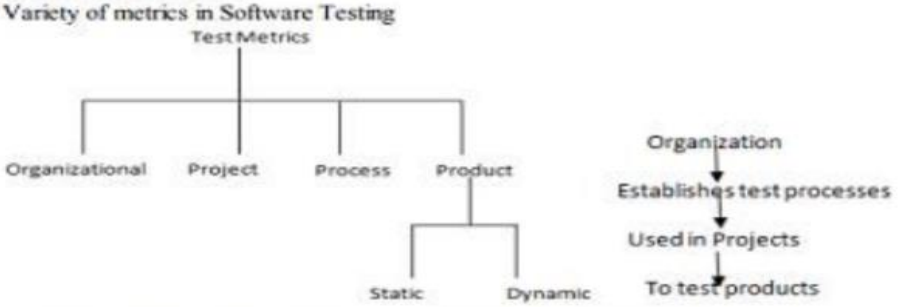
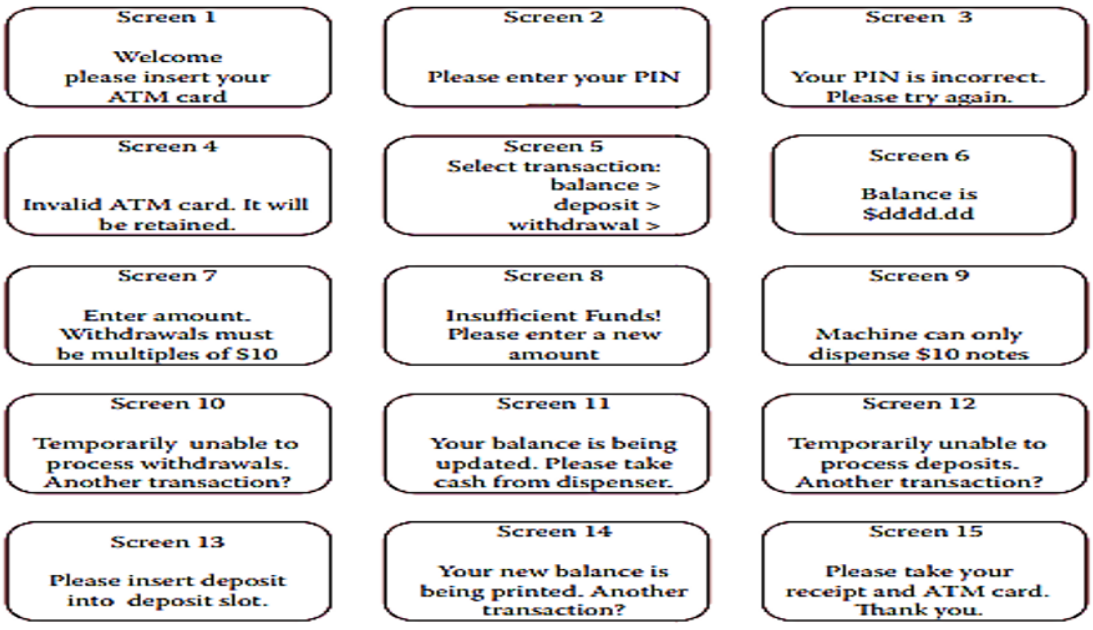


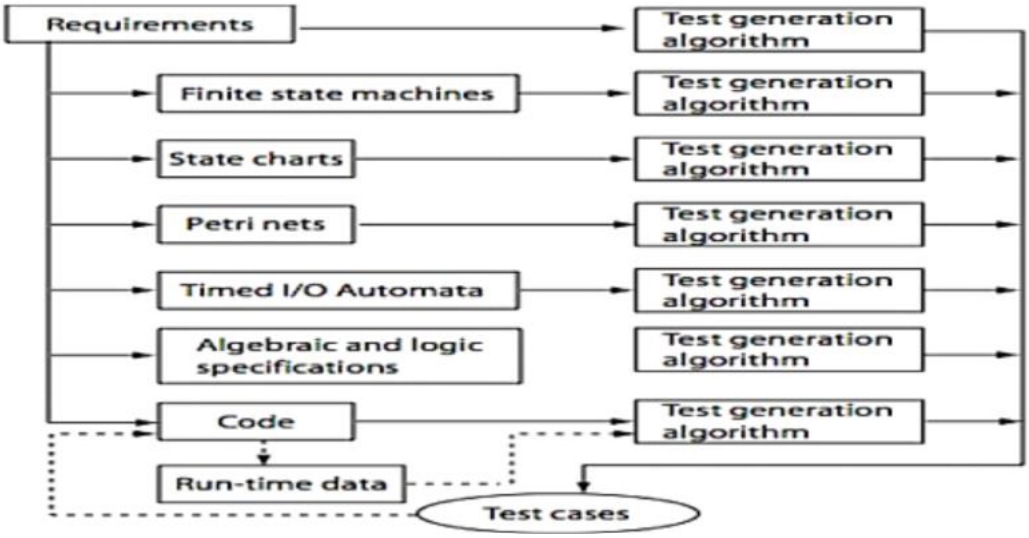


Test case for a variable x, where $a <= x_1 <= b$

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

Case	Locks	Stocks	Barrels	Sales	Comm	Comment
1	1	1	1	100	10	Output minimum
2	1	1	2	125	12.5	Output minimum +
3	1	2	1	130	13	Output minimum +
4	2	1	1	145	14.5	Output minimum +
5	5	5	5	500	50	Midpoint
6	10	10	9	975	97.5	Border point –
7	10	9	10	970	97	Border point –
8	9	10	10	955	95.5	Border point –
9	10	10	10	1000	100	Border point
10	10	10	11	1025	103.75	Border point +
11	10	11	10	1030	104.5	Border point +
12	11	10	10	1045	106.75	Border point +
13	14	14	14	1400	160	Midpoint
14	18	18	17	1775	216.25	Border point –
15	18	17	18	1770	215.5	Border point –

5.	**Elaborate different test metrics in testing the software.** Scheme and Solution: Explanation of each metric carries 2+2+2+2+2 = 10 Marks **Test Metrics:** • Quantitative measurement determining the extent to which a software process, product or project possesses a certain attribute (used for tracking purposes) • Goal for the metric is to quantify the progress of the product toward a specified quality objective • standard measurement. • Variety of metrics in Software Testing  • Each set of metrics has its value in Monitoring, planning and control • There are 4 metrics which are core areas • Schedule -measures actual completion times of various activities and compare these with estimated time to completion	[10]	1	L2
6.	**Describe the following:** **a. SATM system screens b. Test generation Strategies** Scheme: Explanation+Diagram of SATM Screens 2+3 marks and Explanation+ Diagram of strategies 2+3 marks Solution:  Model based: require that a subset of the requirements be modeled using a formal notation (usually graphical). Models: Finite State Machines, Timed automata, Petri net, etc. Specification based: require that a subset of the requirements be modeled using a formal mathematical notation. Examples: B, Z, and Larch. Code based: generate tests directly from the code.	[10]	1	L2



Faculty Signature CCI Signature HOD Signature