USN										

Internal Assessment Test 2 – JUL 2024

Scheme of Evaluation

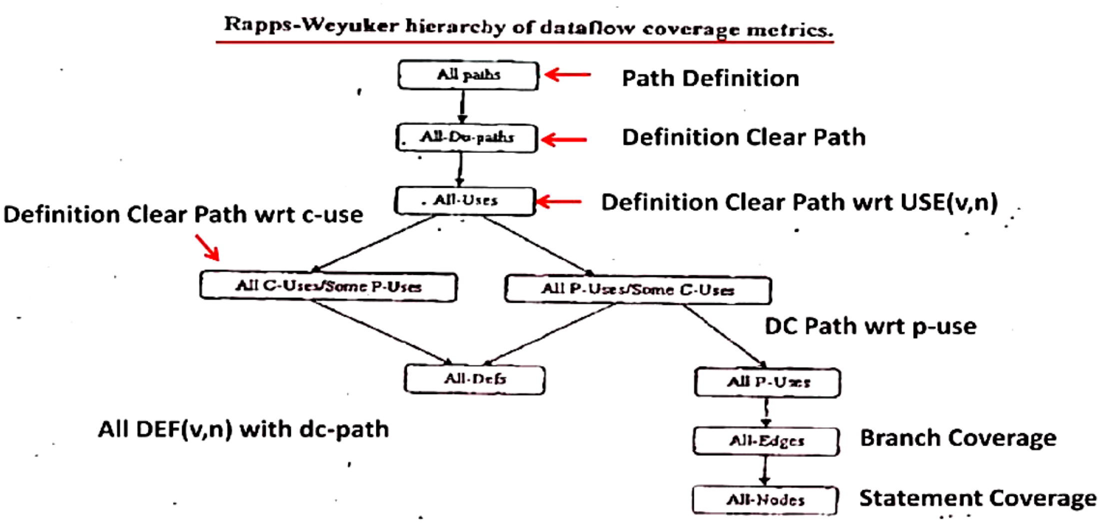
Sub:	**SOFTWARE TESTING**					Sub Code:	**21IS63**	Branch:	**ISE**
Date:	**10/07/2024**	Duration:	**90 min**	Max Marks:	**50**	Sem/Sec:	**VI/ A, B & C**		**OBE**

Answer any FIVE FULL Questions — MARKS | CO | RBT

1\. **a.Describe Rapps-Weyuker hierarchy of dataflow coverage metrics.** [4] CO: 3 RBT: L2

Scheme: Diagram+Explanation-2+2 marks

Solution:



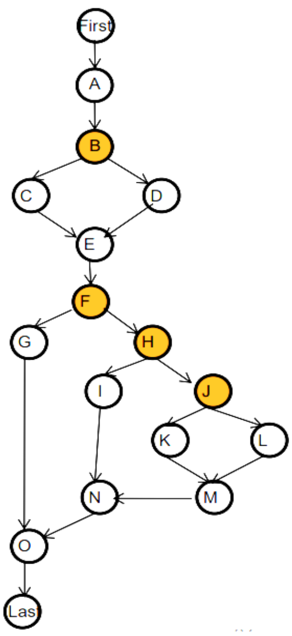
b.Explain McCabe's basis path testing for Triangle problem. [6]

Scheme: Definition+McCabes path Table with flow graph+Test case – 1+3+2 marks

Solution:

McCabe's Path of Triangle Program

McCabe	Paths	Expected Results
Original	P1: First-A-B-C-E-F-H-J-K-M-N-O-Last	Scalene
Flip P1 at B	P2: First-A-B-D-E-F-H-J-K-M-N-O-Last	Infeasible path
Flip P1 at F	P3: First-A-B-C-E-F-G-O-Last	Infeasible path
Flip P1 at H	P4: First-A-B-C-E-F-H-I-N-O-Last	Equilateral
Flip P1 at J	P5: First-A-B-C-E-F-H-J-L-M-N-O-Last	Isosceles



2.	a.Generate Decision Table and Test cases for the NextDate Function. Scheme: Decision Table +Test Cases – 3 +3 marks Solution:	[6]	2	L2

	1	2	3	4	5	6	7	8	9	10		
c1: Month in	M1	M1	M1	M1	M1	M2	M2	M2	M2	M2		
c2: Day in	D1	D2	D3	D4	D5	D1	D2	D3	D4	D5		
c3: Year in	—	—	—	—	—	—	—	—	—	—		
Actions												
a1: Impossible					X							
a2: Increment day	X	X	X			X	X	X	X			
a3: Reset day				X						X		
a4: Increment month				X						X		
a5: Reset month												
a6: Increment year												
	11	12	13	14	15	16	17	18	19	20	21	22
c1: Month in	M3	M3	M3	M3	M3	M4	M4	M4	M4	M4	M4	M4
c2: Day in	D1	D2	D3	D4	D5	D1	D2	D2	D3	D3	D4	D5
c3: Year in	—	—	—	—	—	—	Y1	Y2	Y1	Y2	—	—
Actions												
a1: Impossible										X	X	X
a2: Increment day	X	X	X	X		X	X					
a3: Reset day					X			X	X			
a4: Increment month								X	X			
a5: Reset month					X							
a6: Increment year					X							

Case ID	Month	Day	Year	Expected Output
1–3	4	15	2001	4/16/2001
4	4	30	2001	5/1/2001
5	4	31	2001	Invalid input date
6–9	1	15	2001	1/16/2001
10	1	31	2001	2/1/2001
11–14	12	15	2001	12/16/2001
15	12	31	2001	1/1/2002
16	2	15	2001	2/16/2001
17	2	28	2004	2/29/2004
18	2	28	2001	3/1/2001
19	2	29	2004	3/1/2004
20	2	29	2001	Invalid input date
21, 22	2	30	2001	Invalid input date

b. Describe Mutual Exclusive property for Triangle Problem. [4]

Scheme: Definition +Test case Table – 1+3 marks

Solution:

When conditions refer to equivalence classes, decision tables have a characteristic appearance and Conditions in the decision table in Table 7.4 are from the NextDate problem; they refer to the **mutually exclusive possibilities** for the month variable. Because a month is in exactly one equivalence class, we cannot ever have a rule in which two entries are true.

The don't care entries (—) really mean "**must be false**."

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
Rule count	32	16	8	1	1	1	1	1	1	1	1
a1: Not a triangle	X	X	X								
a2: Scalene											X
a3: Isosceles							X		X	X	
a4: Equilateral				X							
a5: Impossible					X	X		X			

3. Compute the following for the triangle problem. A. Flow Graph B. Cyclomatic Complexity C. Path Testing Strategy and coverage table. D. test cases. [10] 3 L3

Scheme: code+ flow graph+ Cyclomatic complexity+ Path testing +Test cases – 2+3+1+2+2 marks

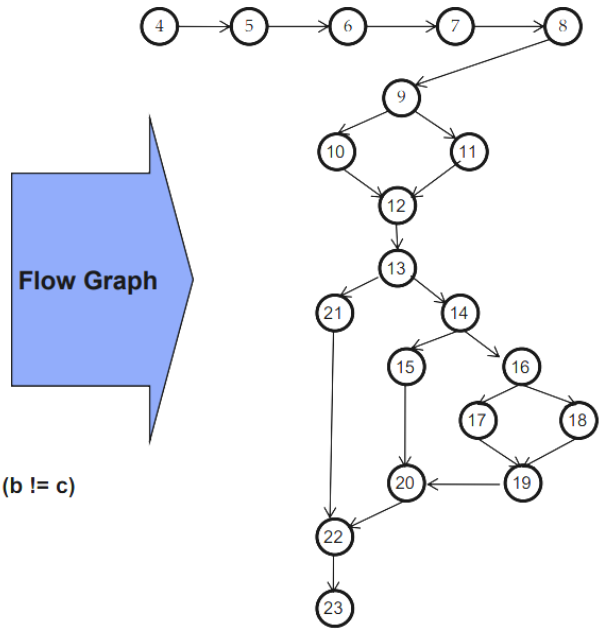
Solution:

```
1. Program Triangle
2. Dim a, b,c  As  Integer
3. Dim IsTriangle As Boolean

4. Output ( "enter a,b, and c integers")
5. Input (a,b,c)
6. Output ("side 1 is", a)
7. Output ("side 2 is", b)
8. Output ("side 3 is", c)

9. If (a<b+c) AND (b<a+c) And (c<b+a)
10.  then  IsTriangle = True
11.  else   IsTriangle = False
12. endif

13. If IsTriangle
14.    then if (a=b) AND (b=c)
15.          then Output ("equilateral")
16.          else if (a != b) AND (a != b) AND (b != c)
17.                 then Output ( "Scalene")
18.                 else Output ("Isosceles")
19.               endif
20.        endif
21.    else  Output ("not a triangle")
22. endif
23. end Triangle2
```



① 4-5-6-7-8-9-10-12-13-21-22-23

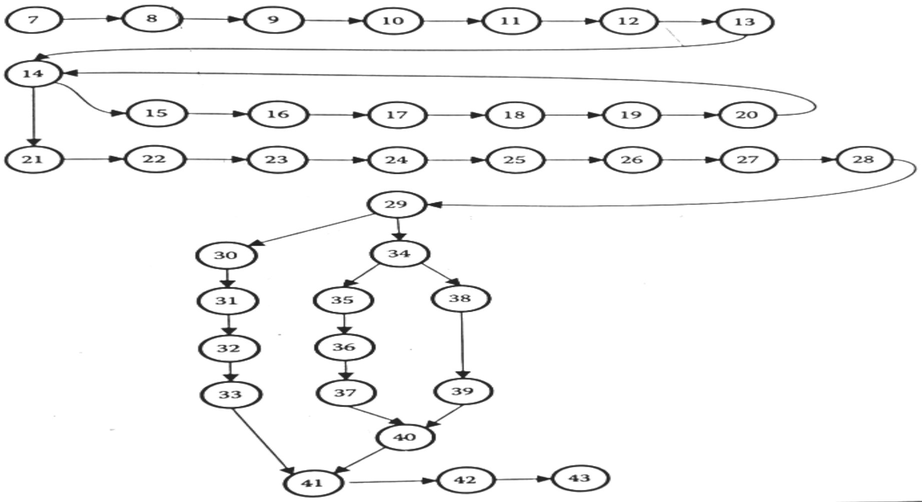
② 4-5-6-7-8-9-11-12-13-14-15-20-22-23

③ 4-5-6-7-8-9-11-12-13-14-16-17-19-20-22-23

④ 4-5-6-7-8-9-11-12-13-14-16-18-19-20-22-23

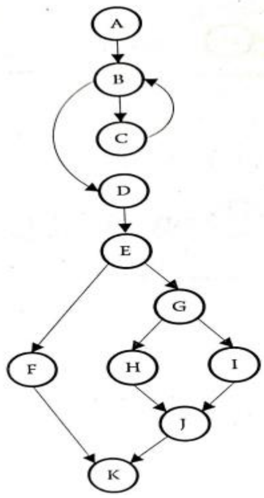
Path	Decision				Test case			Expected Results
	9	13	14	16	a	b	c	
①	T	F			100	100	200	Not A triangle
②	F	T	T		100	100	100	Equilateral
③	F	T	F	T	100	50	60	Scalene
④	F	T	T	F	100	100	50	Isosceles

	Cyclomatic Complexity = V(G)=e-n+2p = 23-20+2(1)=5			
4.	For Commission problem, generate DD and DU Path Table using Data Flow Testing. Scheme: Program+Flow graph+DDPath+DU Path-3+2+2+3 marks Solution: Program for Commission Problem in any language	[10]	3	L3

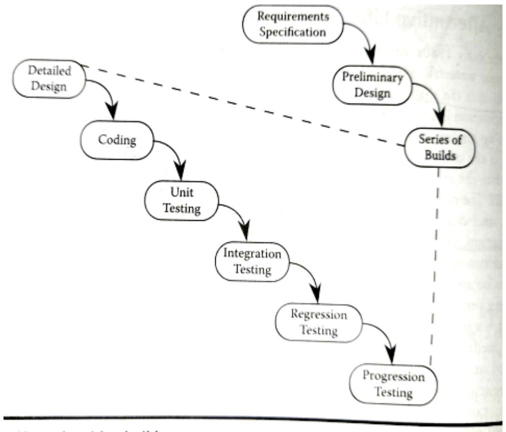


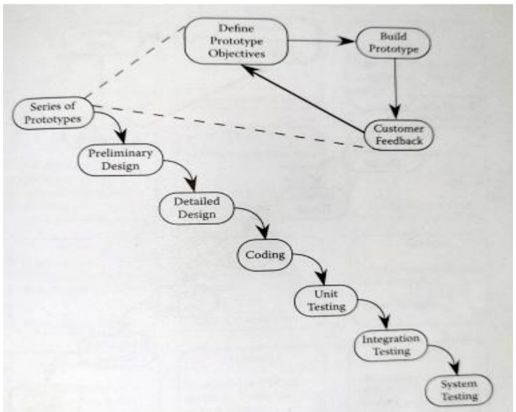
DD-Path	Nodes
A	7, 8, 9, 10, 11, 12, 13
B	14
C	15, 16, 17, 18, 19,20
D	21, 22, 23, 24, 25, 26, 27, 28
E	29
F	30, 31, 32, 33
G	34
H	35, 36, 37
I	38, 39
J	40
K	41, 42, 43

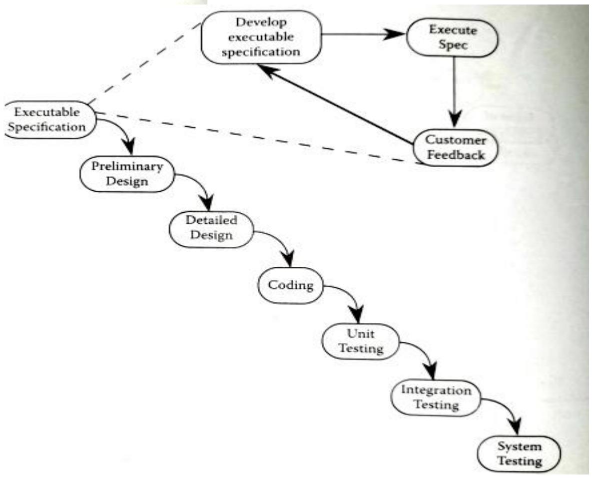
DD-Path graph



Variable	Defined at Node	Used at Node
lockPrice	7	24
stockPrice	8	25
barrelPrice	9	26
totalLocks	10, 16	16, 21, 24
totalStocks	11, 17	17, 22, 25
totalBarrels	12, 18	18, 23, 26
locks	13, 19	14, 16
stocks	15	17
barrels	15	18
lockSales	24	27
stockSales	25	27
barrelSales	26	27
sales	27	28, 29, 33, 34, 37, 39
commission	31, 32, 33, 36, 37, 39	32, 33, 37, 42

<table>
<tr><td>5.</td><td>Explain different models of waterfall spin-offs in testing.
Scheme: Definition+Diagram+Explanation-2+3+5 marks
Solution:




Life cycle with a build sequence.

</td><td>[10]</td><td>4</td><td>L2</td></tr>
<tr><td>6.</td><td>Explain slice-based testing in detail.
Scheme: Definition+Explanation+Example- 2+3+5 marks
Solution:
Given a program P and a set V of variables in P, a slice on the variable set V at statement n, written S(V, n) is the set of all statements in P prior to node n that contribute to the values of variables in V at node n

• Five forms of usage nodes
– P-use (used in a predicate (decision))
– C-use (used in computation)
– O-use (used for output, e.g. printf())
– L-use (used for location, e.g. pointers, subscripts)
– I-use (iteration, e.g. internal counters)

• Two forms of definition nodes
– I-def (defined by input, e.g. scanf())
– A-def (defined by assignment)</td><td>[10]</td><td>3</td><td>L2</td></tr>
</table>

GUIDELINES FOR CHOOSING SLICES

- If statement fragment n in S(V, n) is a ***defining node*** for v, then ***n is included in the slice.***
- If statement fragment n is a ***usage node***, then it is ***included in the slice.***
- If a statement is both a ***defining*** and a ***usage node***, then it is ***included in the slice***.
- In a static slice, ***P-uses and C-uses*** of other variables (not the v in the slice set V) are included to the ***extend that their execution affects the value*** of the variable v.
- If the value of v is the same whether a statement fragment is included or excluded, ***exclude*** the statement fragment.
- ***O-use, L-use, and I-use*** nodes are **excluded from slices**

SLICE ON LOCK VARIABLE

In the program fragment

```
13. Input(locks)
14. While NOT(locks = -1)
15.     Input(stocks, barrels)
16.     totalLocks = totalLocks + locks
17.     totalStocks = totalStocks + stocks
18.     totalBarrels = totalBarrels + barrels
19.     Input(locks)
20.EndWhile
```

There are these slices on locks (notice that statements 15, 17, and 18 do not appear):

S1: S(locks, 13) = {13} DEFINING NODE I-DEF
S2: S(locks, 14) = {13, 14, 19, 20}
S3: S(locks, 16) = {13, 14, 19, 20}
S4: S(locks, 19) = {19} DEFINING NODE I-DEF

SLICE ON STOCKS AND BARRELS

S_5: S(stocks,15) = {13,14,15,19,20}
S_6: S(stocks,17) = {13,14,15, 19,20}
S_7: S(barrels,15) = {13,14,15,19,20}
S_8: S(barrels,18) = {13,14,15, 19,20}

SLICE ON TOTAL LOCKS

S_9: S(totallocks,10) = {10} (A-def)
S_{10}: S(totallocks,16) = {10,13,14,16,19,20} (A-def & C-use)
S(totallocks,21)= {10,13,14,16,19,20} 21 is an O-Use of totallocks, excluded
S_{11}: S(totallocks,24) = {10,13,14,16,19,20} (24 is a C-use of total locks)

Faculty Signature CCI Signature HOD Signature