CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Scheme and Solutions of Internal Assesment Test – II

Sub:	SOFTWARE TESTING							Code:	10IS65
Date:		Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ISE

Answer Any **FIVE FULL** Questions

		Marks	OBE CO	OBE RBT
1 (a)	Define various data flow testing criteria	[05]	CO3	L1
(b)	Explain different structural test coverage metrics	[05]	CO3	L4

Data flow testing criterias are:

Consider a program P which has a program graph G(P), has set of variables V

Definition 1:

If the node $n \in G(P)$ is said to be defining node of variable $v \in V$ ~~such that~~ iff the variable v is defined in the statement n

Definition 2:

If the node $n \in G(P)$ is said to be usage node variable $v \in V$ iff the variable v is used in the statement n

Definition 3:

If n is a predicate node then outdegree of n ≥ 2 or it is compute node (outdegree ≤ 1)

Definition 4:

definition / use path in PATHS(P) such that some $v \in V$. definition node DEF(v,m) and usage node USE(v,n) such that m and n are the initial and terminal nodes of path

Definition 5:

definition clear path is a du path in PATHS(P). iff DEF(v,m) and USE(v,n) are ~~the~~ initial and final nodes ~~have~~ and no node in ~~these~~ path should ~~be~~ define the variable v.

Structural test coverage metrics

Test coverage metrix is a tool to measure the extent to which the testcases cover the program.

Some of metrixes are:

Metrix	Description of metrix
C_0	Every statement
C_1	Every du path
C_{1p}	predicate value in each outcome
C_2	C1 coverage + loop coverage
C_d	C1 Coverage + all dependable pairs of du path
C_{stat}	statistically significant paths
C_∞	All executable paths

If test cases satisfy C_1 metrix in which every du paths passes, then 85% of testcases satisfy means almost no faults in program

2 (a)	Define the following. 1. DD Path 2. Predicate Node 3. DU Path 4. DC Path	[04]	CO3	L1

1. DD path :

DD paths is a path in PATHS(P) such that some $v \in V$. DEF(v,m) and USE(v,n) are definition and usage node such that m and n are the initial and terminal nodes of graph

2. Predicate Node

If the node $n \in G(P)$ is said tobepredicate node iff the node n is having outdegree ≥ 2

3. DU path :

Du path is a path in PATHS(P) such that some $v \in V$. The definition node DEF(v,m) and usage node USE(v,n) such that m and n are the initial and terminal nodes of graph

4. DC Path :

DC path is a du path in PATH(P) iff DEF(v,m) and USE(v,n) are initial and final nodes and no node in these path should define the vaelable v

Commission problem

```
1. program commission
2. DEM locks, stocks, barrels as integer
3. DEM lockprice, stockprice, barrelprice as real
4. DEM totallocks, totalstocks, total barrel as integer
5 DEM locksales, stocksales, barrelsales as real
6 DEM sales, comm as real
7. lockprice = 35
8 stockprice = 95
9 barrelprice = 40
10. total locks = 0
11. total stocks = 0
12. total barrels = 0
13. Input(locks)
14. while NOT(locks <> -1)
15. Input (stocks, barrels)
16. totallocks += locks
17. totalstocks += stocks
18. totalbarrels += barrels
19. Input(locks)
```

```
20.  END while
21.  output ( "total locks)
22.  output (total stocks)
23.  output (total barrels)
24.  locksales = lockprice * total locks
25.  Stocksales = stockprice * total stocks
26.  barrelsales = barrelprice * total barrels
27.  sales = locksales + stocksales + barrel sales
29.  Output (sales)
30.  If (sales > 1800)
31   Then
32.      comm = 0.1 *1000
33.      comm = comm + 0.15 (800)
34.      comm = comm + 0.2 (sales-1800
35   Else If (sales > 1000)
36    Then
37.       comm = 0.1 *1000
38        comm = comm + 0.15 (sales - 1000)
39    Else
40        Comm = sales *0.1
41   EndIf
42 EndIf
43.  output (comm)
44 End.
```

(b)	List out the du paths for the variables locks, total locks and sales with the help of commission problem pseudocode.	[06]	CO3	L1

dupaths for locks :

DEF (locks, 13)
DEF (locks, 19)
USE (locks, 14)
USE (locks, 16)

P_1 = <13, 14> → dc path

P_2 = <13, 14, 15, 16> → dc path

P_3 = <19, 20, 14> → dc path

P_4 = <19, 20, 14, 15, 16> → dc path

dupaths for totallocks :

DEF (totallocks, 10)
DEF (totallocks, 16)
USE (totallocks, 16)
USE (totallocks, 21)
USE (totallocks, 24)

P_5 = <10, 11, 12, 13, 14, 15, 16> → dc path

P_6 = <P_5, 17, 18, 19, 20, 14, 21> → not dc path

P_7 = <P_6, 22, 23, 24> → not dc path

P_8 = <16, 17, 18, 19, 20, 14, 21> → not dc

P_9 = <P_8, 22, 23, 24> → not dc path

du paths for sales :

DEF (sales, 27), USE (sales, 29)
USE (sales, 30)
USE (sales, 34)
USE (sales, 35)
USE (sales, 38)
USE (sales, 40)

P_{10} = <27, 29> → du

P_{11} = <27, 29, 30> → du

P_{12} = <P_{11}, 31, 32, 33, 34> → du

P_{13} = <P_{11}, 35> → du

P_{14} = <P_{13}, 36, 37, 38> → du.

P_{15} = <P_{13}, 39, 40> → du

3 (a)	Explain McCabe's basis path method. Apply basis path method on triangle problem and explain.	[10]	CO3	L4

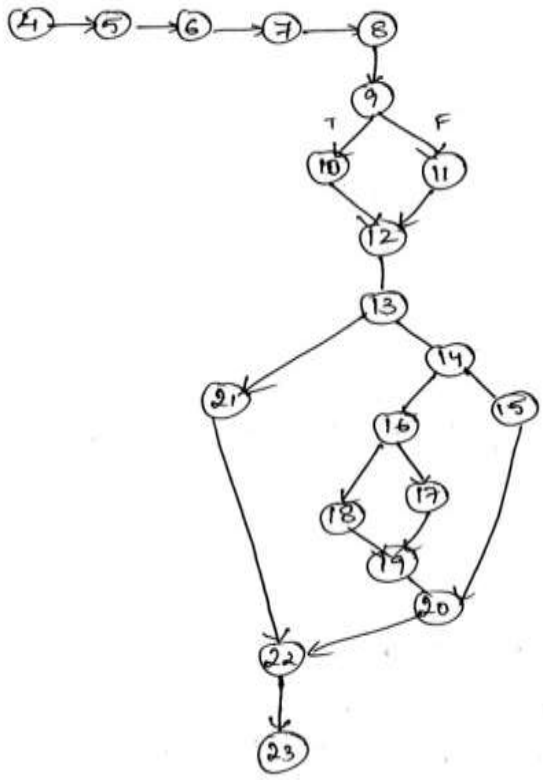
Mccabe's basis path method

It is a directed graph or the program graph or DD path graph. Mccabe's basis on testing is Cyclomatic complexity which $e - n + p$ where e is number of edges, n is number of nodes, p is no. of connection

Triangle problem

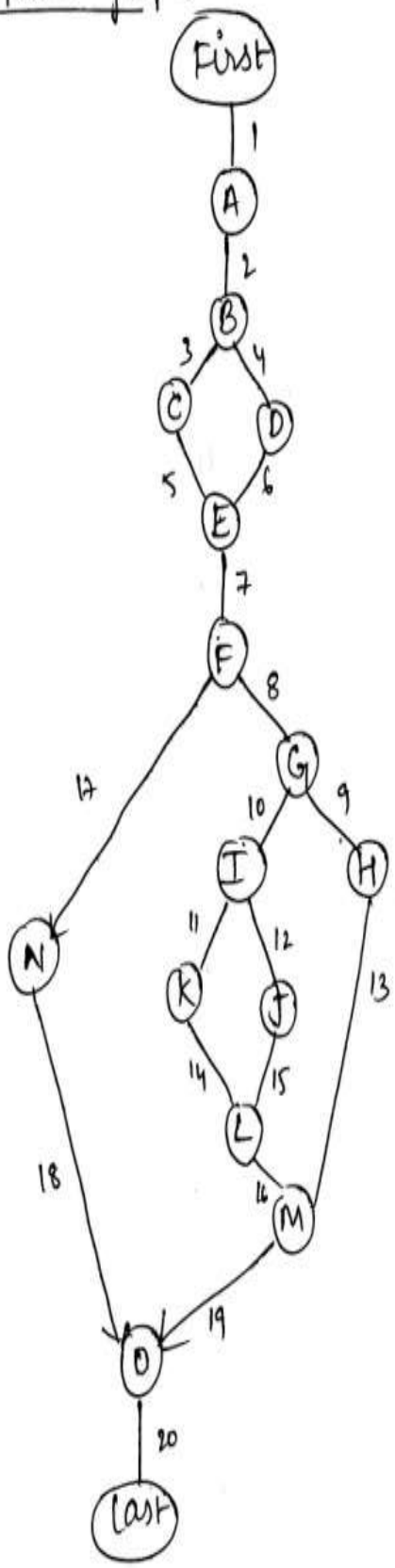
```
1.  program
2.  DIM  a, b, c  as  integer
3   DIM  ISAvariable as  boolean
4.  Output (" enter 3 sides of A^le")
5.  Input (a, b, c)
6.  Output (a)
7.  Output (b)
8.  Output (c)
9.  If  (a<b+c) AND  (b<a+c)  AND  (c<a+b)
10  THEN   ISAtriangle = true
11  Else   ISAtriangle = False
12. EndIf
13. If  ISA triangle
14   THEN  If (a=b) and (b=c)
15       then  output (" equilatral")
16        ElseIf (a≠b)  AND  (b≠c) AND  (c≠a)
17          Then  output (" scalene")
18        Else  output (" Isoselus)
19   EndIf
20  EndIf
21  Else output (" NOt a A^le");
22 endIf
23  End
```

program graph



program graph node	DD Path node	cases
4	First	1
5-8	A	5
9	B	3
10	C	4
11	D	4
12	E	3
13	F	3
14	G	3
15	H	4
16	I	3
17	J	4
18	K	4
19	L	3
20	M	3
21	N	4
22	O	3
23	last	2

DD path graph



Cyclomatic Complexity =

$e - n + 1$

$20 - 17 + 1 = 4$

(4 paths)

P_1 : A B C E F G H M O

P_2 : A B C E F G I J K L M O

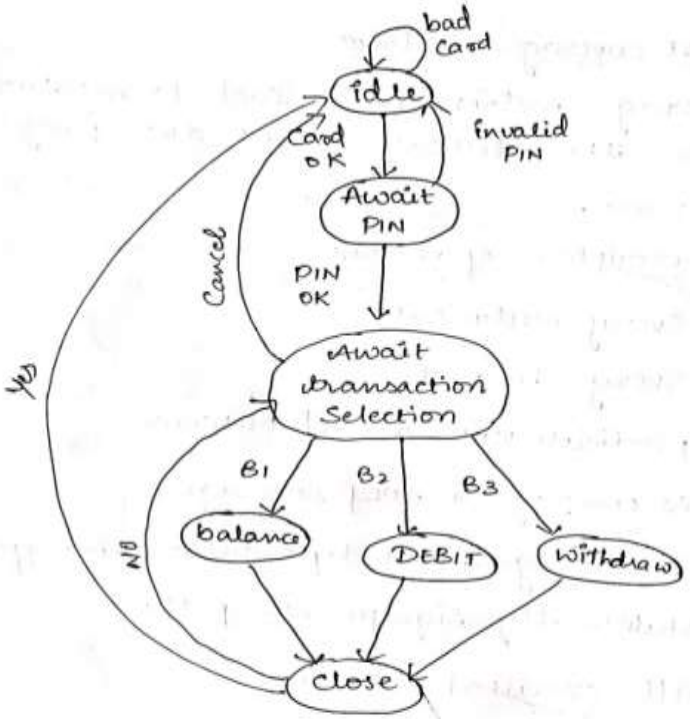
P_3 : A B C E F G I K L M O

P_4 : A B D E F N O

4 (a)	With neat diagrams, explain the upper level finite state machine and context diagram of Simple ATM Application.	[10]	CO1	L1

multices are provided by lattice. Lattice is important because at each level many faults are detected.

upper level finite state machine of Simple ATM



→ In this simple ATM is implemented as a set of states

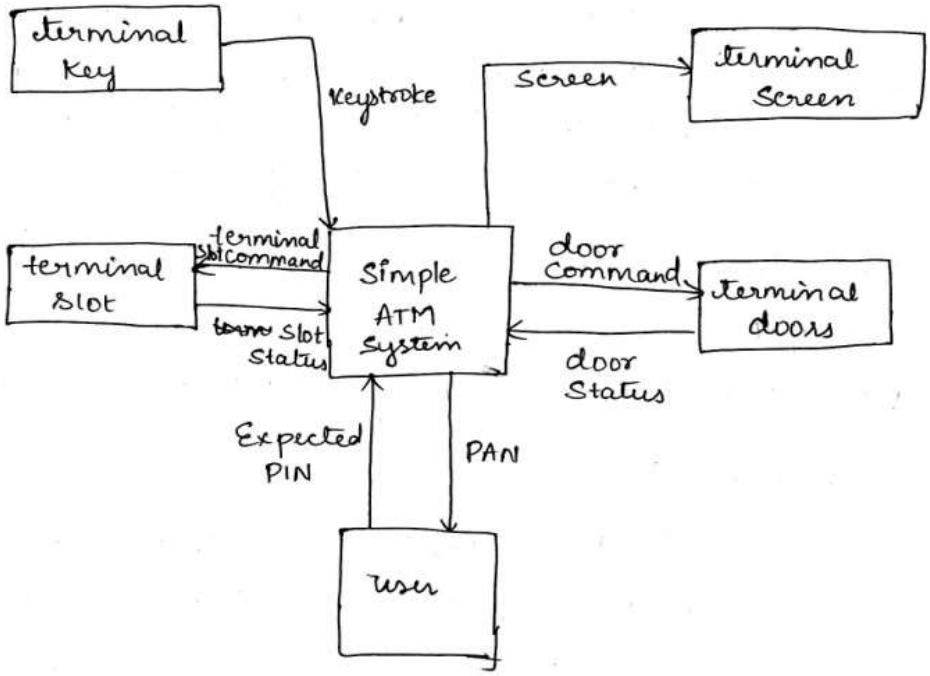
→ There is change in state when either user click on terminal ^ data flows

→ In this if the card is valid it will ask for PIN, If it is valid PIN then user has to select transaction like balance, debit, withdraw or he can cancel from transaction selection

→ If trasaction is over, then if user presses close it go to idle state otherwise transaction selection sta

→ If PIN is invalid for 3 trials it will go idle st

Context diagram of SATM



→ In the above diagram terminal key, termi screen, terminal slot, terminal door are th separate blocks which do specific task

→ This is not detailed diagram, this is simple to illustrate the function of SATM

→ dataflow diagram for data and finite sta machine for control operation of SATM.

6 (a)	Describe the decomposition based integration testing and distinguish between top down integration and bottom up integration.	[10]	CO1	L1

Decomposition based testing :

⇒ The testing is based on the decomposition tree o
In the type of testing, it specifies the order in which units should be tested.

⇒ It mainly tests interfaces between two individu components.

⇒ It can be 4 types top down integration testing bottom up, bigbang etc

Top down integration testing :

⇒ It starts testing the main function first. main function calls the program sub parts called stubs

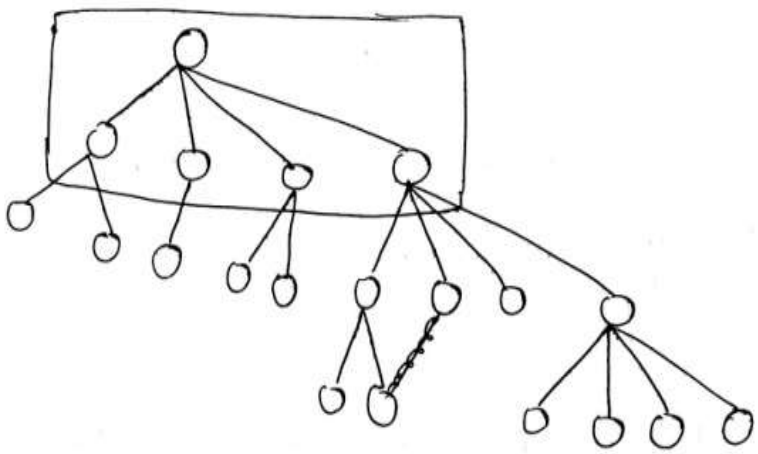
⇒ Stubs are developed to test the main function

⇒ For example in SATM system if we want to do top d integration testing, we need to develop stubs first validate PIN etc

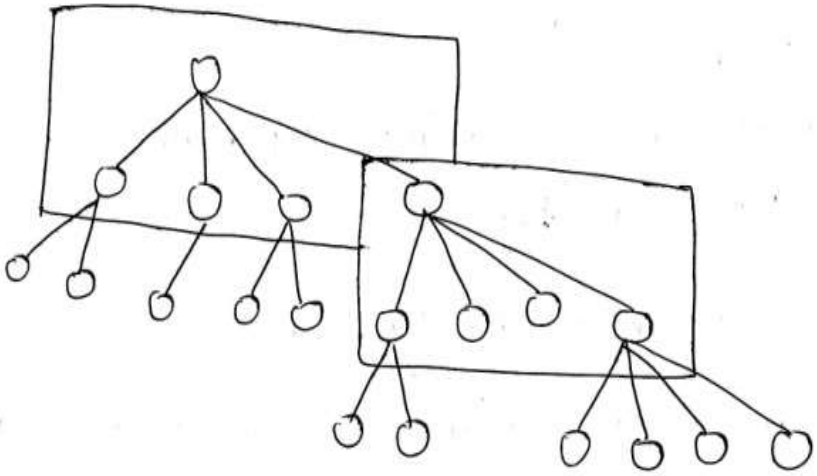
⇒ After developing stubs we test the main function stand alone. If it has no faults the stubs are replaced with actual code

⇒ It follows breadth first search traversal

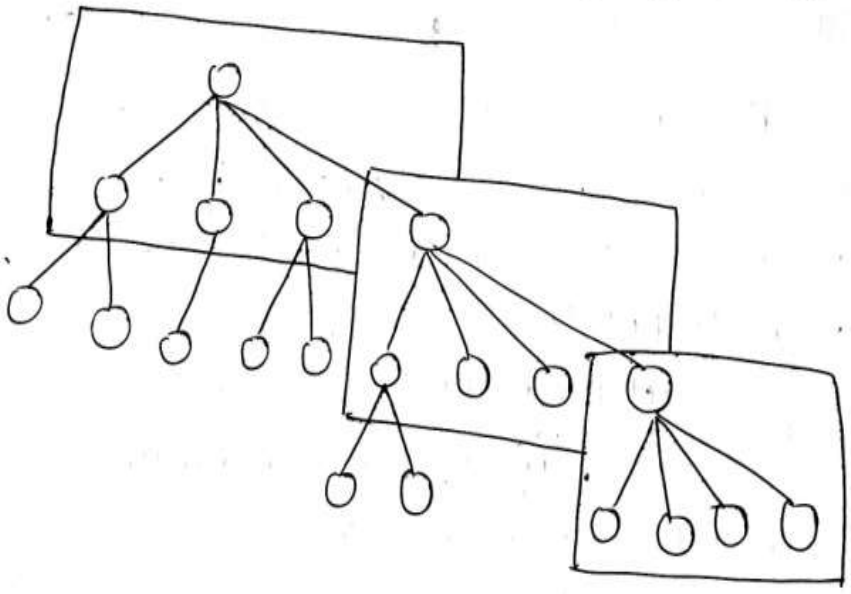
Step1



Step2:



Step3:



Bottom up integration testing:

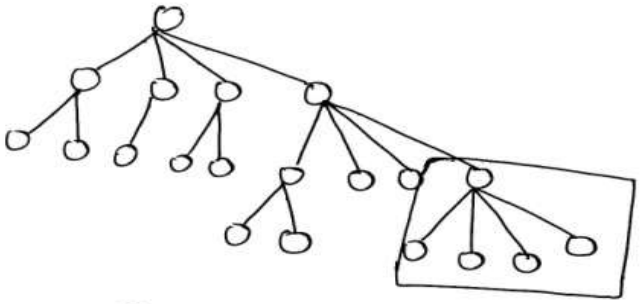
→ It is the mirror image of top down integration testi
but only difference is instead of stubs, driver mo
are used to emulate next step

→ In top down integration testing, for each child stub shoul
be created but in bottom up less number of
driver modules are required

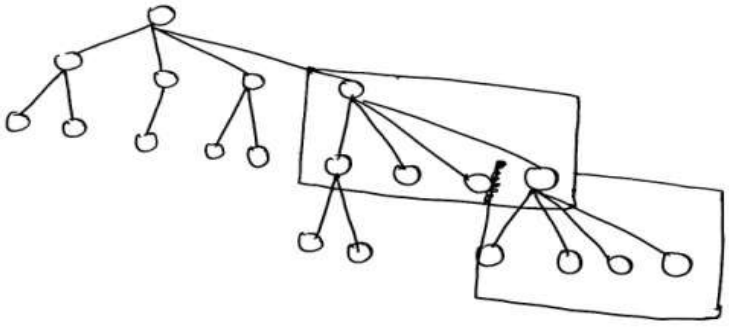
→ driver modules are complicated

→ It starts from decomposition leaves to main pro

Stage 1:



Stage 2



Stage 3.