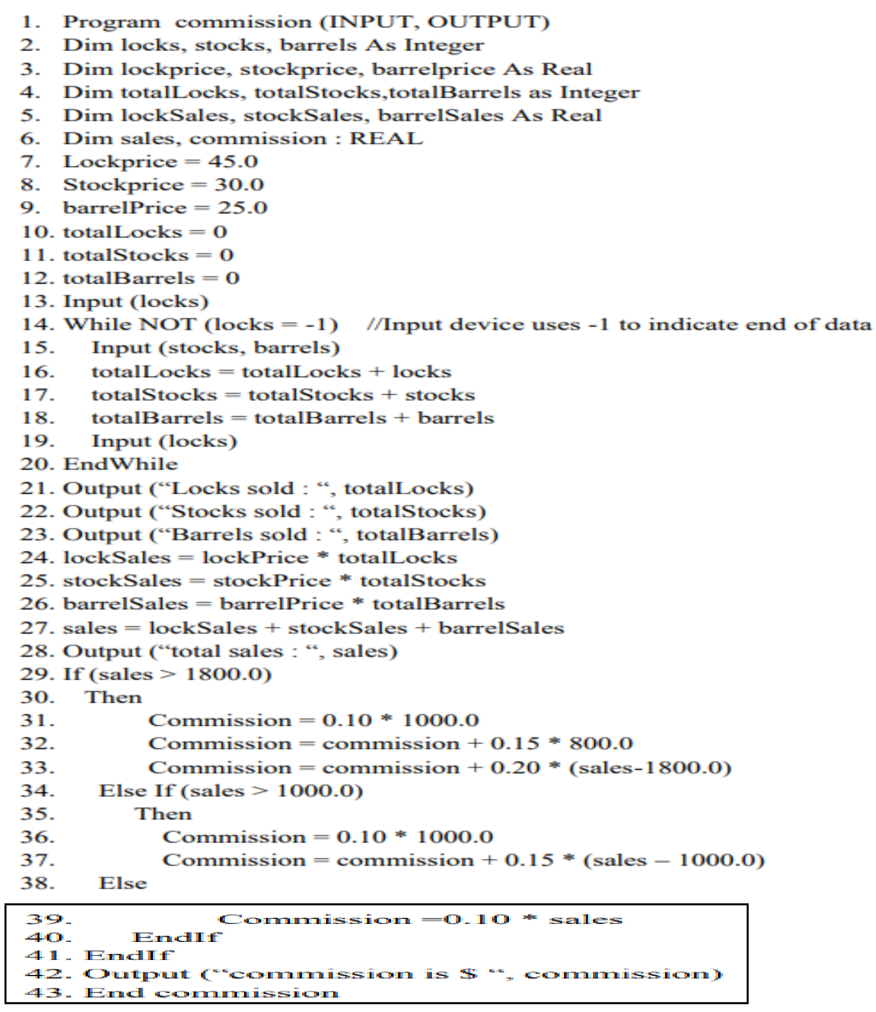
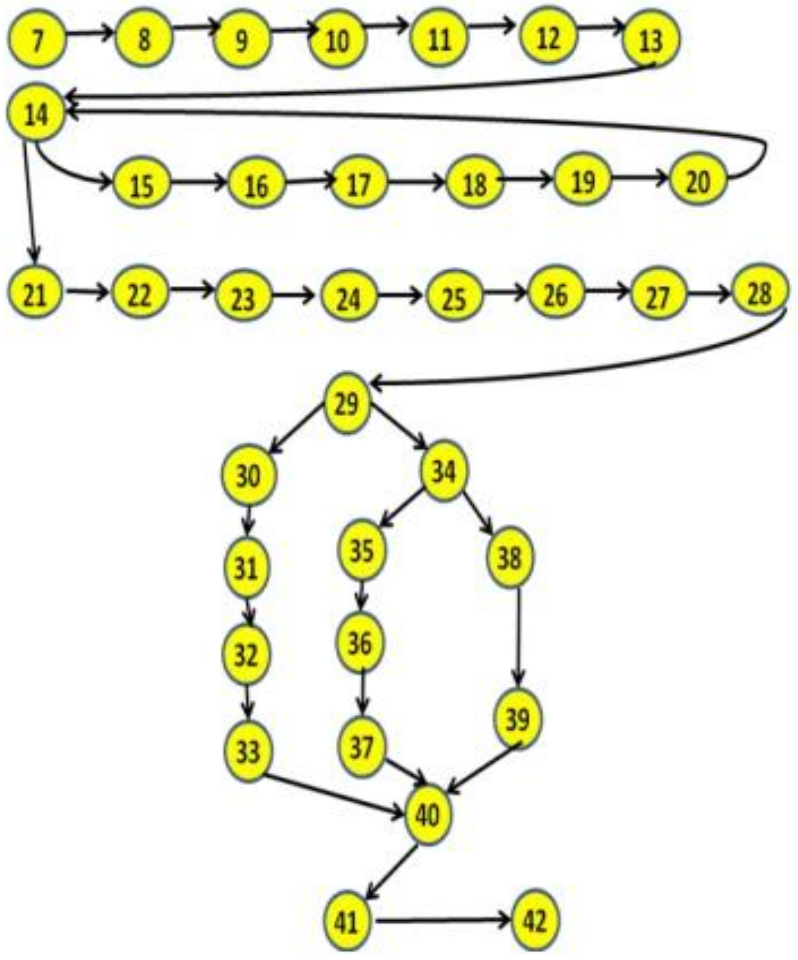
Internal Assesment Test – III, December 2020									
Sub:	Software Testing							Code:	18MCA351
Date:	16-12-2020	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA

	Answer Any 5 **QUESTION**s	Marks	OBE	
			CO	RBT
1	Briefly discuss the dependability properties in process framework.	10	CO5	L2
2	Write a program of the commission problem. The statement of the problem: a rifle salesperson in the former Arizona territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. The locks cost $45, stocks cost $30 and barrels cost $25. The salesperson had to sell at least one complete rifle per month and production limits were such that at the most the sales person could sell in a month was 70 locks, 80 stocks and 90 barrels. At the end of a month, the salesperson sent a very short telegram showing -1 lock sold, the gunsmith then knew the sales for the month were complete and computed the salesperson's commission as follows: 10% on sales up to $1000, 15% on the next $800 and 20% on any sales in excess of $1800. The commission program produced a monthly sales report that gave the total number of locks, stocks and barrels sold, the salesperson's total dollar sales, and finally, the commission. Construct the program graph and define use nodes for variable in the above problem.	10	CO4	L1
3	Why organizational factors are needed in process frame work.?	10	CO5	L2
4	What do you mean by Fault-Based testing? Explain what are the assumptions made in Fault-Based testing.	10	CO5	L1
5	What is Mutation Analysis? What are different types of Mutation Analysis? Explain them with example.	10	CO5	L3
6	Give Definition for All-defs, All uses, All P-uses/some C-uses, All C-uses/ some P-Uses, All Du-paths. Write hierarchy of dataflow coverage Metrics.	10	CO4	L2
7	What is cyclomatic complexity? Explain how to calculate cyclomatic complexity of a given program by considering the biggest of three number logic. Explain slice –based testing guidelines and observations in detail.	10	CO4	L4
8	Define DD-path. Draw DD-graph for triangle problem.	10	CO4	L3

NOTE: Question Number 2 must be answer as mandatory requirement

Question #	**Description**
1	Briefly discuss the dependability properties in process framework. Five key pillars of dependability in a system Security, Availability, Reliability, Safety and Resilience.
2	Write a program of the commission problem. The statement of the problem: a rifle salesperson in the former Arizona territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. The locks cost $45, stocks cost $30 and barrels cost $25. The salesperson had to sell at least one complete rifle per month and production limits were such that at the most the sales person could sell in a month was 70 locks, 80 stocks and 90 barrels. At the end of a month, the salesperson sent a very short telegram showing -1 lock sold, the gunsmith then knew the sales for the month were complete and computed the salesperson's commission as follows: 10% on sales up to $1000, 15% on the next $800 and 20% on any sales in excess of $1800. The commission program produced a monthly sales report that gave the total number of locks, stocks and barrels sold, the salesperson's total dollar sales, and finally, the commission. Construct the program graph and define use nodes for variable in the above problem. 

```
1.  Program  commission (INPUT, OUTPUT)
2.  Dim locks, stocks, barrels As Integer
3.  Dim lockprice, stockprice, barrelprice As Real
4.  Dim totalLocks, totalStocks,totalBarrels as Integer
5.  Dim lockSales, stockSales, barrelSales As Real
6.  Dim sales, commission : REAL
7.  Lockprice = 45.0
8.  Stockprice = 30.0
9.  barrelPrice = 25.0
10. totalLocks = 0
11. totalStocks = 0
12. totalBarrels = 0
13. Input (locks)
14. While NOT (locks = -1)    //Input device uses -1 to indicate end of data
15.     Input (stocks, barrels)
16.     totalLocks = totalLocks + locks
17.     totalStocks = totalStocks + stocks
18.     totalBarrels = totalBarrels + barrels
19.     Input (locks)
20. EndWhile
21. Output ("Locks sold : ", totalLocks)
22. Output ("Stocks sold : ", totalStocks)
23. Output ("Barrels sold : ", totalBarrels)
24. lockSales = lockPrice * totalLocks
25. stockSales = stockPrice * totalStocks
26. barrelSales = barrelPrice * totalBarrels
27. sales = lockSales + stockSales + barrelSales
28. Output ("total sales : ", sales)
29. If (sales > 1800.0)
30.    Then
31.            Commission = 0.10 * 1000.0
32.            Commission = commission + 0.15 * 800.0
33.            Commission = commission + 0.20 * (sales-1800.0)
34.      Else If (sales > 1000.0)
35.          Then
36.              Commission = 0.10 * 1000.0
37.              Commission = commission + 0.15 * (sales – 1000.0)
38.      Else
39.                  Commission =0.10 * sales
40.          EndIf
41. EndIf
42. Output ("commission is $ ", commission)
43. End commission
```



3

Why organizational factors are needed in process frame work.?

1.Review plans and objectives.

Objectives are the specific activities that must be completed to achieve goals. Plans shape the activities needed to reach those goals. Managers must examine plans initially and continue to do so as plans change and new goals are developed.

2.Determine the work activities necessary to accomplish objectives.

Although this task may seem overwhelming to some managers, it doesn't need to be. Managers simply list and analyze all the tasks that need to be accomplished in order to reach organizational goals.

3.Classify and group the necessary work activities into manageable units.

A manager can group activities based on four models of departmentalization: functional, geographical, product, and customer.

4.Assign activities and delegate authority.

Managers assign the defined work activities to specific individuals. Also, they give each individual the authority (right) to carry out the assigned tasks.

5.Design a hierarchy of relationships.

A manager should determine the vertical (decision-making) and horizontal (coordinating) relationships of the organization as a whole. Next, using the organizational chart, a manager should diagram the relationships.

4

What do you mean by Fault-Based testing? Explain what are the assumptions made in Fault-Based testing.

A model of potential program faults is a valuable source of information for evaluating and designing test suites. Some fault knowledge is commonly used in functional and structural testing. Fault-based testing uses a fault model directly to hypothesize potential faults in a program under test, as well as to create or evaluate test suites based on its efficacy in detecting those hypothetical faults.

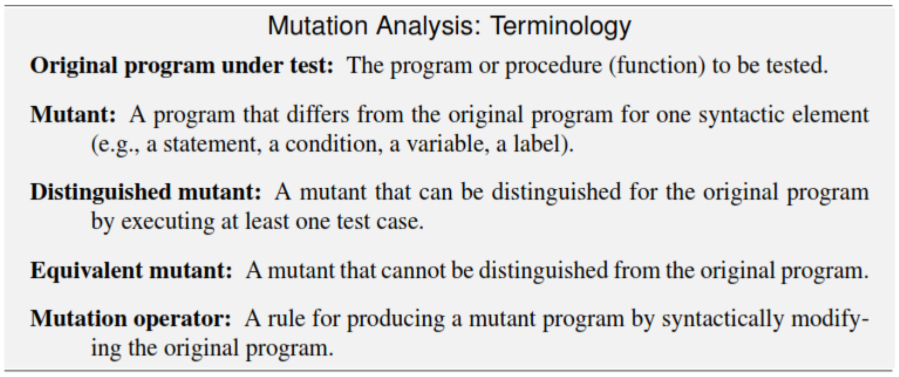
The effectiveness of fault-based testing depends on the quality of the fault model and on some basic assumptions about the relation of the seeded faults to faults that might actually be present. In practice, the seeded faults are small syntactic changes, like replacing one variable reference by another in an expression, or changing a comparison from < to <=. We may hypothesize that these are representative of faults actually present in the program.

If the program under test has an actual fault, we may hypothesize (assume) that it differs from another corrected program by only a small textual change. If so, then we need merely distinguish the program from all such small variants to ensure detection of all such faults. This is known as the competent programmer hypothesis, an assumption that the program under test is close to a correct program.

5

What is Mutation Analysis? What are different types of Mutation Analysis? Explain them with example.

Mutation analysis is the most common form of software fault-based testing. A fault model is used to produce hypothetical faulty programs by creating variants of the program under test. Variants are created by "seeding" faults, that is, by making a small change to the program under test following a pattern in the fault model. The patterns for changing program text are called mutation operators, and each variant program is called a mutant.



Mutation Analysis: Terminology

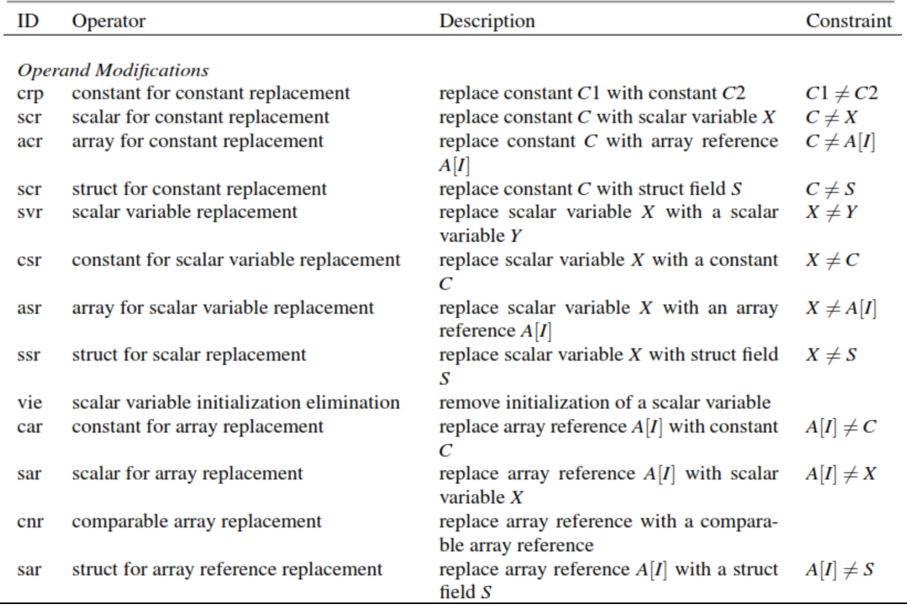
Original program under test: The program or procedure (function) to be tested.

Mutant: A program that differs from the original program for one syntactic element (e.g., a statement, a condition, a variable, a label).

Distinguished mutant: A mutant that can be distinguished for the original program by executing at least one test case.

Equivalent mutant: A mutant that cannot be distinguished from the original program.

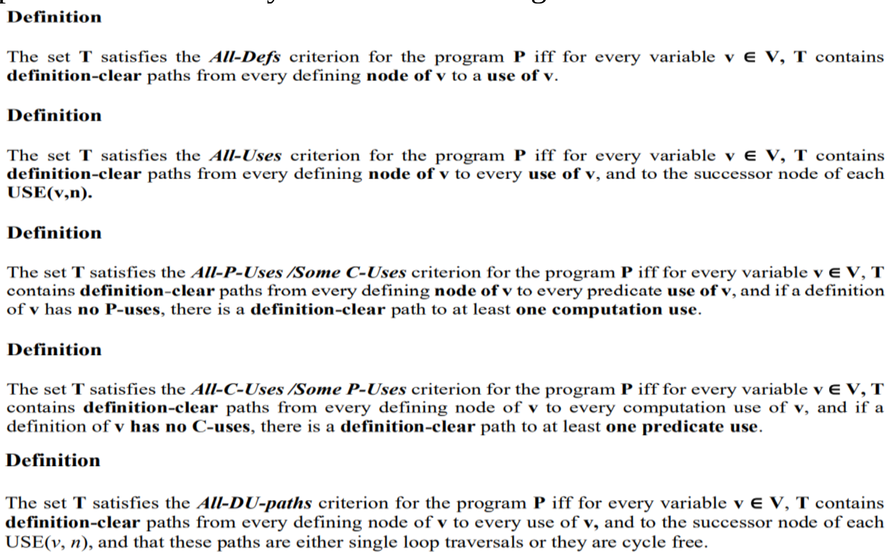
Mutation operator: A rule for producing a mutant program by syntactically modifying the original program.



ID	Operator	Description	Constraint
Operand Modifications			
crp	constant for constant replacement	replace constant $C1$ with constant $C2$	$C1 \neq C2$
scr	scalar for constant replacement	replace constant C with scalar variable X	$C \neq X$
acr	array for constant replacement	replace constant C with array reference $A[I]$	$C \neq A[I]$
scr	struct for constant replacement	replace constant C with struct field S	$C \neq S$
svr	scalar variable replacement	replace scalar variable X with a scalar variable Y	$X \neq Y$
csr	constant for scalar variable replacement	replace scalar variable X with a constant C	$X \neq C$
asr	array for scalar variable replacement	replace scalar variable X with an array reference $A[I]$	$X \neq A[I]$
ssr	struct for scalar replacement	replace scalar variable X with struct field S	$X \neq S$
vie	scalar variable initialization elimination	remove initialization of a scalar variable	
car	constant for array replacement	replace array reference $A[I]$ with constant C	$A[I] \neq C$
sar	scalar for array replacement	replace array reference $A[I]$ with scalar variable X	$A[I] \neq X$
cnr	comparable array replacement	replace array reference with a comparable array reference	
sar	struct for array reference replacement	replace array reference $A[I]$ with a struct field S	$A[I] \neq S$

6

Give Definition for All-defs, All uses, All P-uses/some C-uses, All C-uses/ some P-Uses, All Du-paths. Write hierarchy of dataflow coverage Metrics.



Definition

The set **T** satisfies the ***All-Defs*** criterion for the program **P** iff for every variable **v ∈ V**, **T** contains **definition-clear** paths from every defining **node of v** to a **use of v**.

Definition

The set **T** satisfies the ***All-Uses*** criterion for the program **P** iff for every variable **v ∈ V**, **T** contains **definition-clear** paths from every defining **node of v** to every **use of v**, and to the successor node of each **USE(v,n).**

Definition

The set **T** satisfies the ***All-P-Uses /Some C-Uses*** criterion for the program **P** iff for every variable **v ∈ V**, **T** contains **definition-clear** paths from every defining **node of v** to every predicate **use of v**, and if a definition of **v** has **no P-uses**, there is a **definition-clear** path to at least **one computation use**.

Definition

The set **T** satisfies the ***All-C-Uses /Some P-Uses*** criterion for the program **P** iff for every variable **v ∈ V**, **T** contains **definition-clear** paths from every defining node of **v** to every computation use of **v**, and if a definition of **v has no C-uses**, there is a **definition-clear** path to at least **one predicate use**.

Definition

The set **T** satisfies the ***All-DU-paths*** criterion for the program **P** iff for every variable **v ∈ V**, **T** contains **definition-clear** paths from every defining node of **v** to every use of **v**, and to the successor node of each USE(v, n), and that these paths are either single loop traversals or they are cycle free.

7

What is cyclomatic complexity? Explain how to calculate cyclomatic complexity of a given program by considering the biggest of three number logic. Explain slice –based testing guidelines and observations in detail.

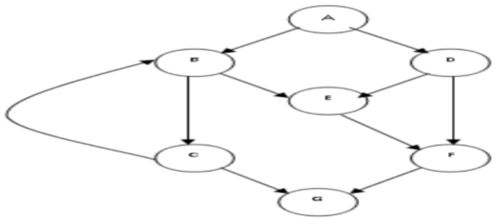
Introduced in 1976 by McCabe, is one of the most commonly used metrics in software development.

• Provides a quantitative measure of the logical complexity of a program in terms of Cyclomatic number

• Defines the number of independent paths in the basis set

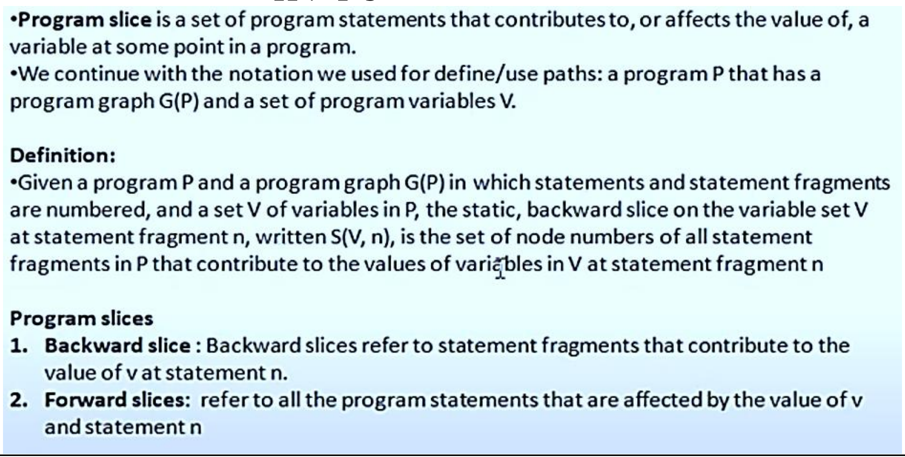
• Provides an upper bound for the number of tests that must be conducted to ensure all statements have been executed at least once

The Cyclomatic Complexity of the program V(G), can be computed from its Control Flow Graph (CFG) G in two ways,
– V(G)=e-n+2p
– V(G)=e-n+p
where e=no. of edges, n=no. of nodes, p=no. of connected regions.



– V(G)=e-n+2p
– V(G)=e-n+p
where e=no. of edges,
n=no. of nodes,
p=no. of connected regions.

- The number of linearly independent paths for the above graph is V(G)=e-n+2p
  =10-7+2(1)=5
- The number of linearly independent circuits for the graph in figure below is V(G) = e-n+p
  =11-7+1=5



•**Program slice** is a set of program statements that contributes to, or affects the value of, a variable at some point in a program.
•We continue with the notation we used for define/use paths: a program P that has a program graph G(P) and a set of program variables V.

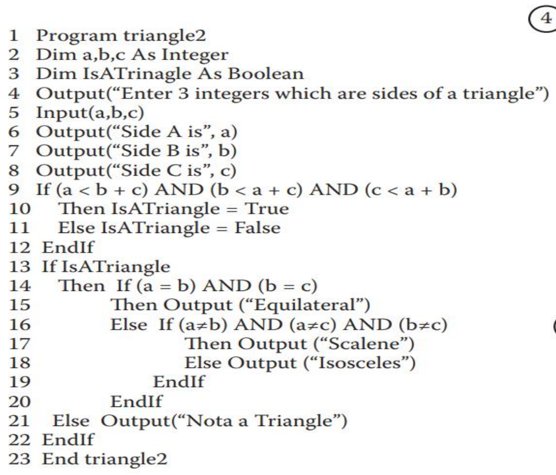
Definition:
•Given a program P and a program graph G(P) in which statements and statement fragments are numbered, and a set V of variables in P, the static, backward slice on the variable set V at statement fragment n, written S(V, n), is the set of node numbers of all statement fragments in P that contribute to the values of variables in V at statement fragment n

Program slices
1. **Backward slice :** Backward slices refer to statement fragments that contribute to the value of v at statement n.
2. **Forward slices:** refer to all the program statements that are affected by the value of v and statement n

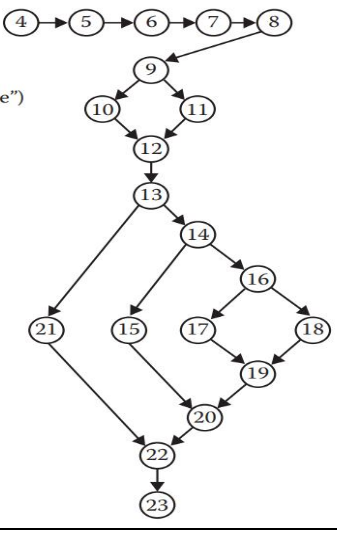
8

Define DD-path. Draw DD-graph for triangle problem.

The best-known form of structural testing is based on a construct known as a Decision-to-Decision path (DD-Path).
• The name refers to a sequence of statements that, in Miller's words begins with the "outway" of a decision statement and ends with the "inway" of the next decision statement.
• We will define DD-Paths in terms of paths of nodes in a directed graph. We might call these paths as chains, where a chain is a path in which the initial and terminal nodes are distinct, and every interior node has indegree = 1 and outdegree = 1.
• Notice that the initial node is 2-connected to every other node in the chain, and no instances of 1- or 3-connected nodes occur.



```
1   Program triangle2
2   Dim a,b,c As Integer
3   Dim IsATrinagle As Boolean
4   Output("Enter 3 integers which are sides of a triangle")
5   Input(a,b,c)
6   Output("Side A is", a)
7   Output("Side B is", b)
8   Output("Side C is", 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 IsATriangle
14    Then  If (a = b) AND (b = c)
15            Then Output ("Equilateral")
16            Else  If (a≠b) AND (a≠c) AND (b≠c)
17                    Then Output ("Scalene")
18                    Else Output ("Isosceles")
19                EndIf
20            EndIf
21    Else  Output("Nota a Triangle")
22  EndIf
23  End triangle2
```


4
5
6
7
8
9
10
11
12
13
14
16
21
15
17
18
19
20
22
23