Page No.
Date / /

Software Testing ~~Tools~~

ch 2.
Software Testing strategies & Techniques

tester programmer
→ white box technique → code checking
user → Black box technique → Functionality

1) Test ability
- James Bach - "How easily computer program can be tested".
- The Test ability of the program is the degree to which a program facilitate and measurement of some performance criteria.

* Characteristic of Test ability
1) Operability
2) Controll ability
3) Observability
4) Simplicity
5) Understandability
6) Stability
7) Decomposability

Page No.
Date / /

Test case Design (→ set of conditions

1) It is the set of conditions under which a tester will determine whether the functionality of software is working correctly or not.

→ Test case id
→ Test case name
→ Test case description
→ Test priority
→ Design by
→ created by
→ Executed by
→ steps Action selection
→ Input
→ Expected o/p
→ Actual o/p
→ Status
→ comment

* White box Testing / structed Testing / glass box testing / transparent testing. (why it is called glass box or transparent
Ans→ (because the code is visible to tester)

Testing strategies

1) Flow graph ~~Notion~~ Notation

2) Compound logic

3) Independent Programs path

4) Cyclomatic complexity

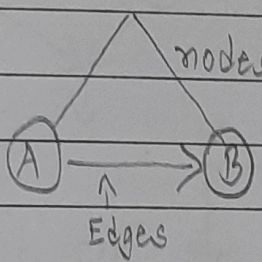
Page No.
Date / /

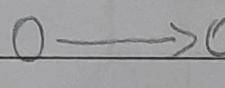
5) Graph matrix

6) Deriving test case

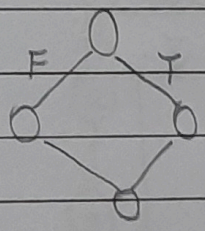
1) Flow graph ~~Notation~~ Notation
It is the collection of number of edges & number of nodes.

Formula
$G = (V, E)$

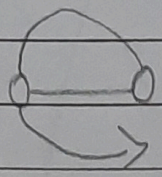




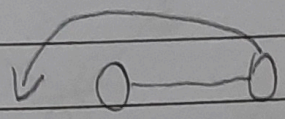
sequence



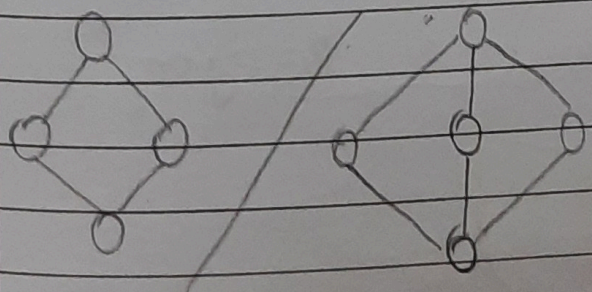
if



while



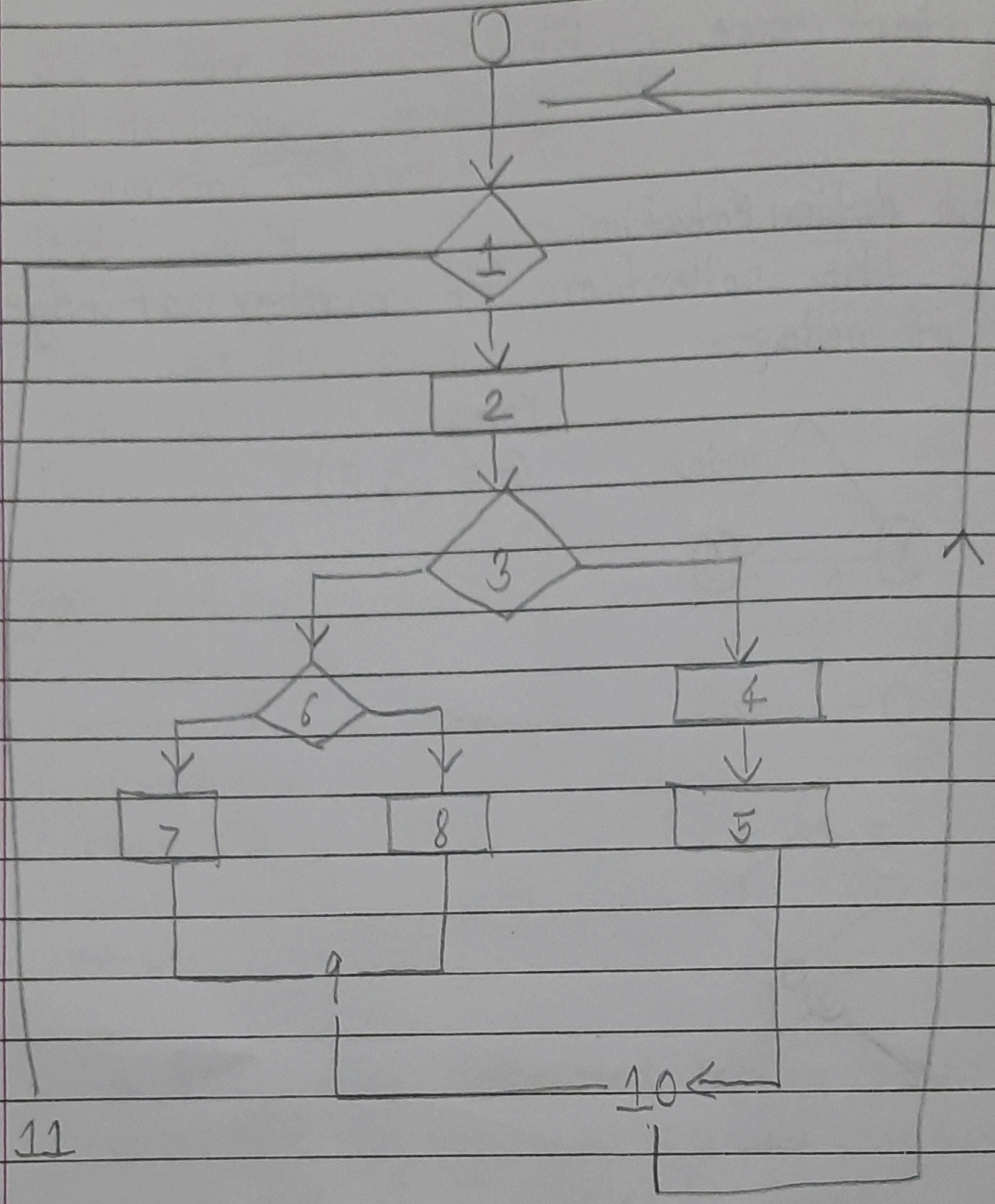
until



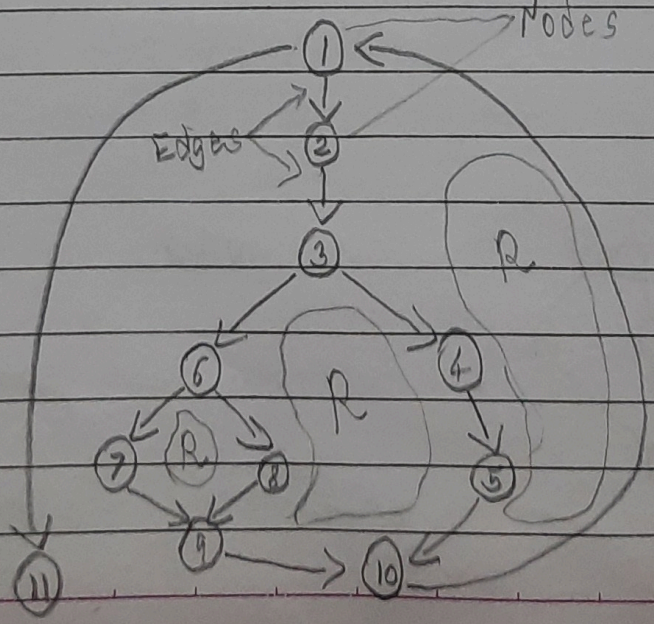
case switch

Page No.
Date / /

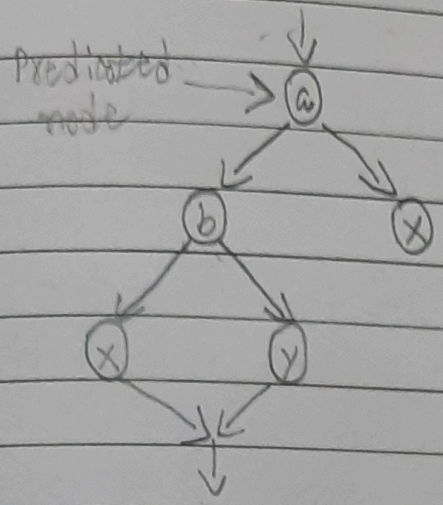
Flowchart



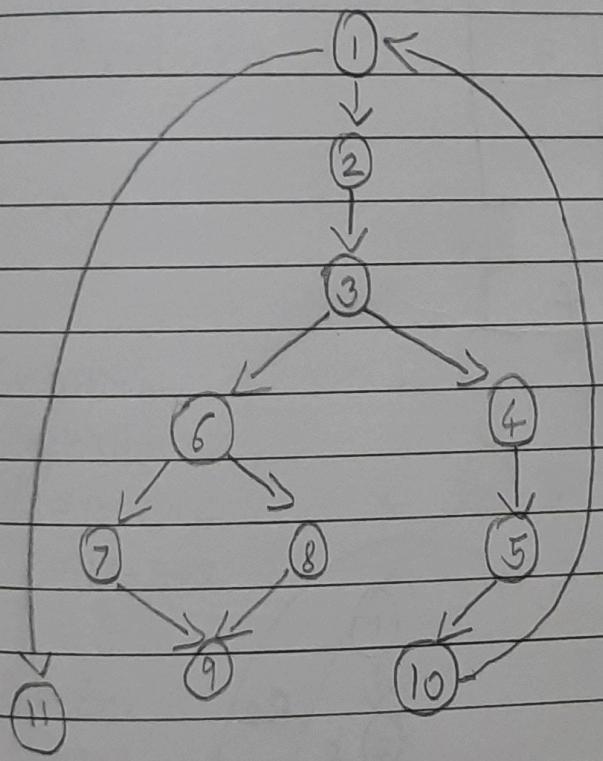
Flow Graph



2) Compound Logic
AND, OR, NOT, NAND, NOR, X-OR, XNOR



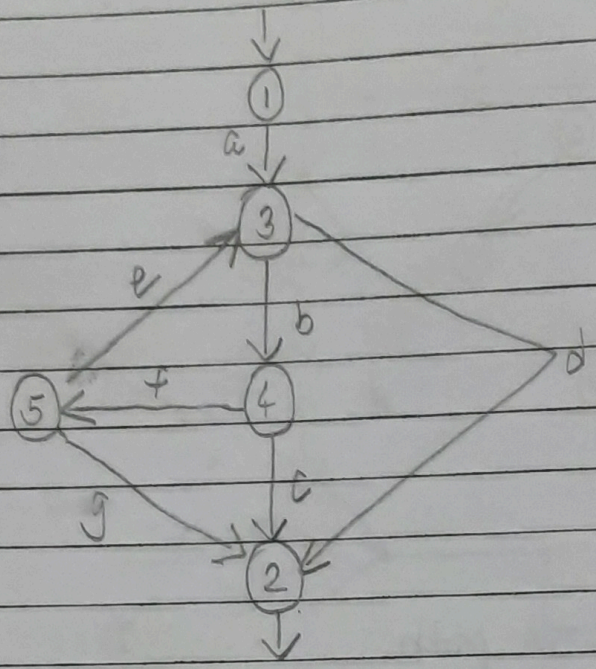
3) Independent programs path



1st path = 1-11
2nd = 1-2-3-4-5-10-1-11
3rd = 1-2-3-6-7-9-10-1-11
4th = 1-2-3-6-8-9-10-1-11

Predicated Node = 1, 3, 6

4) Graph Matrix



	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5		g	e		

5) Deriving Test Case

a, b, c $a > b - c = a$

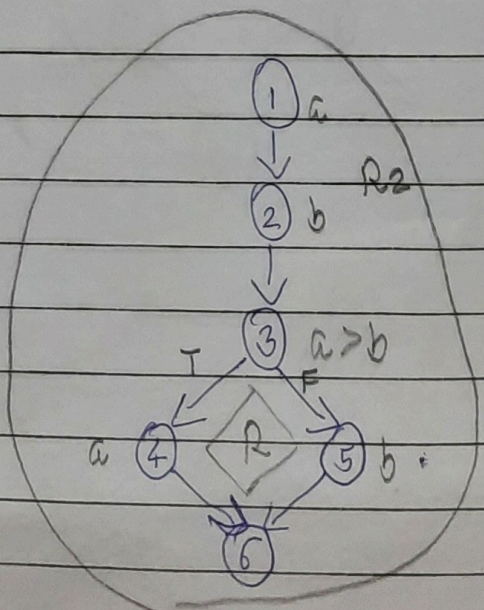
$b > c - c = b$

Nodes = 6

Edges = 6

Predicated = 1 (Node 3)

Node



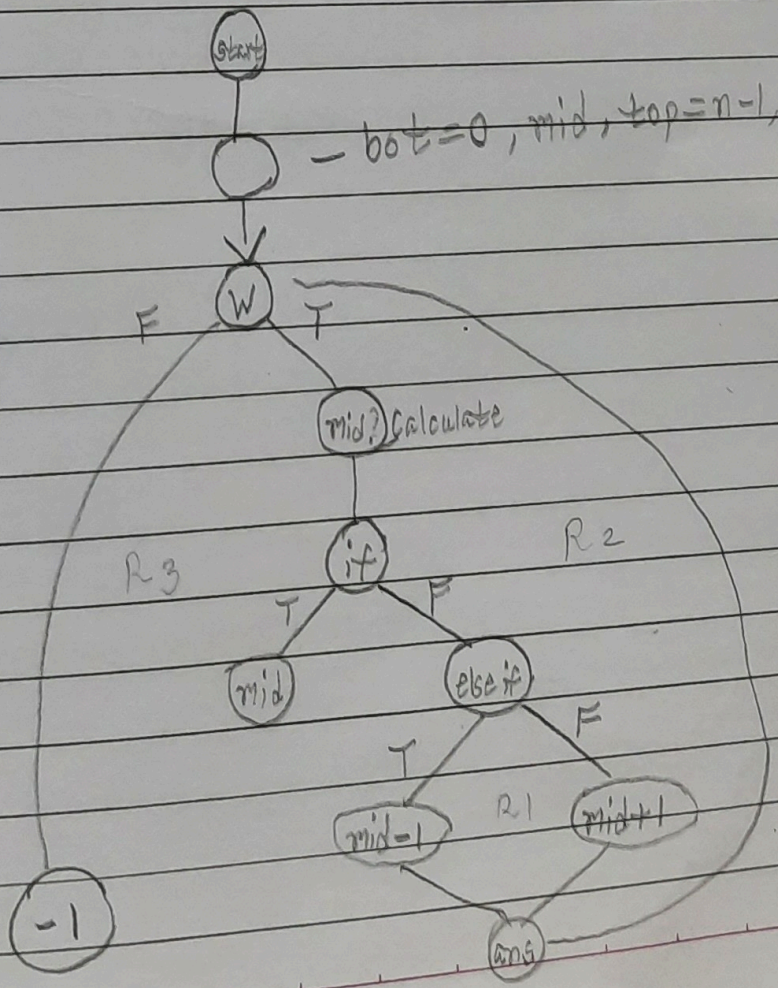
6) Cyclomatic complexity

```
int binarySearch (char *item, char * table[], int n)
{
    int bot = 0, top = n-1, int mid, cmp;
    while (bot <= top)
    {
      mid = (bot + top)/2;
    if (table [mid] == item)
          return mid;
    else if (compare (table [mid], item < 0)
          top = mid-1;
    else
          bot = mid + 1;
    }return -1;                    //not found
```

Cyclomatic complexity

V(G) = E - N + 2
= 12 - 11 + 2
= 3

V(G) = P + 1
= 3 + 1
= 4



Page No.
Date / /

* Control structure Testing

1) Condition Testing / Branching Testing :- if, else, switch

2) Data Flow Testing :- sequence

3) Loop Testing :- while, do while, for

i) Simple loop
ii) Nested loop
iii) Concatenated loop
iv) Unstructured loop

black box = user
white box = programmer

Page No.
Date / /

Software Testing Tool

* Test cases for simple programs
Specification based → black box
structure based → white box
Experience based → experience of the tester/white box + black box

1) write simple program, make a use of loops and control structure with test cases.

code coverage = $\frac{\text{no. of lines executed}}{\text{Total no. of lines/code of the program}} \times 100$

The percentage of a program statement that can be involve or executed during the phase of testing is called code coverage. It comes under white box testing

Types
1) Statement Coverage Testing

```
eg:- if (s>1 && t==0)
     {
        x=9;
     }
```

Test case - s=2; t=0

2) Sequencial Statement

```
Read X;
Read Y;
Z = X+Y;
```

= 100%

3) Decision or Selection control structure

```
1  If P>10
2  THEN
3     X=X+Y;
4  ELSE
5     X=X-Y;
6  ENDIF
```

Test case = P=11

Code coverage = $\frac{4^2}{6^3} \times 100$

$= \frac{200}{3}$

$= 66.3$

for P=9

$= \frac{4^2}{6^3} \times 100$

$= 66.3$

Ex:-

```
1  If P>10
2  THEN
3     X=X+Y
4  ENDIF
```

Test case P=12

code coverage = $\frac{4}{4} \times 100 = 100\%$

4) Loop structure

```
1   x = 10
2   count = 0
3   WHILE  X<=20  DO
4      X = X+1
5      count = count + 1
6   END DO
```

Code coverage $= \frac{6}{6} \times 100 = 100\%$

* EX:-

1) Take i/p of 2 values x & y. 2) Find the sum
3) If the sum > 0 then print it is positive
4) If the sum < 0 then print it is negative

Code structure

```
1 input (int x, int y) {
2   sum = x+y;
3   if (sum > 0)
4   print "positive"
5   else
6   print "negative"
7 }
```

Test case ①
x=6, y=2

```
1 input (6, 2) {
2 sum = 6+2 = 8;
3 if (8 > 0)
4 print "positive"
5 }
```

Code coverage $= \frac{5}{7} \times 100 = \frac{500}{7}$
$= 71.4$

Test case ②
x=-4, y=-3

```
1 input (-4, -3)
2 sum = -4+-3 = -7
3 if (-7 > 0)
4 else
5 print "negative"
6 }
```

code coverage $= \frac{6}{7} \times 100 = 85\%$

Page No.
Date / /

* Example 2 :-

Code Structure

```
1  input (int x, int y) {
2  int z = ((x+y/200) * 100
3     if (z > 50)
4     print "pass"
5  else
6     print "fail"
7  }
```

Test case ①

X = 20, Y = 30

```
1 input(20,30) {
2 int z = ((20+30)/200) * 100 = 25
3 ~~else~~ if (25 > 50)
4 ~~print~~ else
5        print "fail"
6  }
```

Code coverage = $\frac{6}{7} \times 100$

= 85%

Test case ②

X = 100, Y = 75

```
1  input (100, 75)
2  int z = ((100+75/200) * 100
3     if (87.5 > 50)
4     print "pass"
5  }
```

Code coverage = $\frac{5}{7} \times 100$

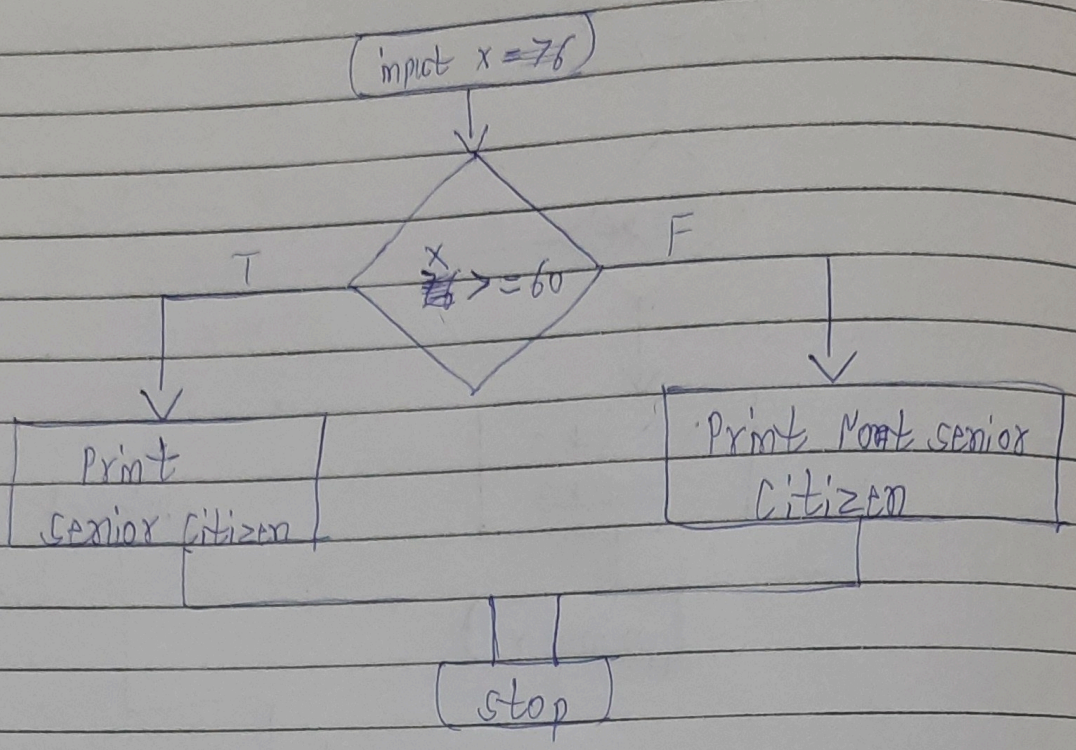
= 71%

* Example 3 :-

```
1 int compute-gcd (x,y)
2    int x, y
3    while (x != y)
4    {
5       if (x > y) then
6          x = x - y;
7       else
8          y = y - x;
9    }
10   return x;
11 }
```

Page No.
Date / /

Test case ① X=76, X=45



For 76

Decision Coverage $= \frac{1}{2} \times 100$

$= 50\%$

For 45

$= \frac{1}{2} \times 100$

$= 50\%$

Test case	Value of x	O/p	Decision coverage
1	76	Senior	50%
2	45	Not senior citizen	50%

Ex 4)

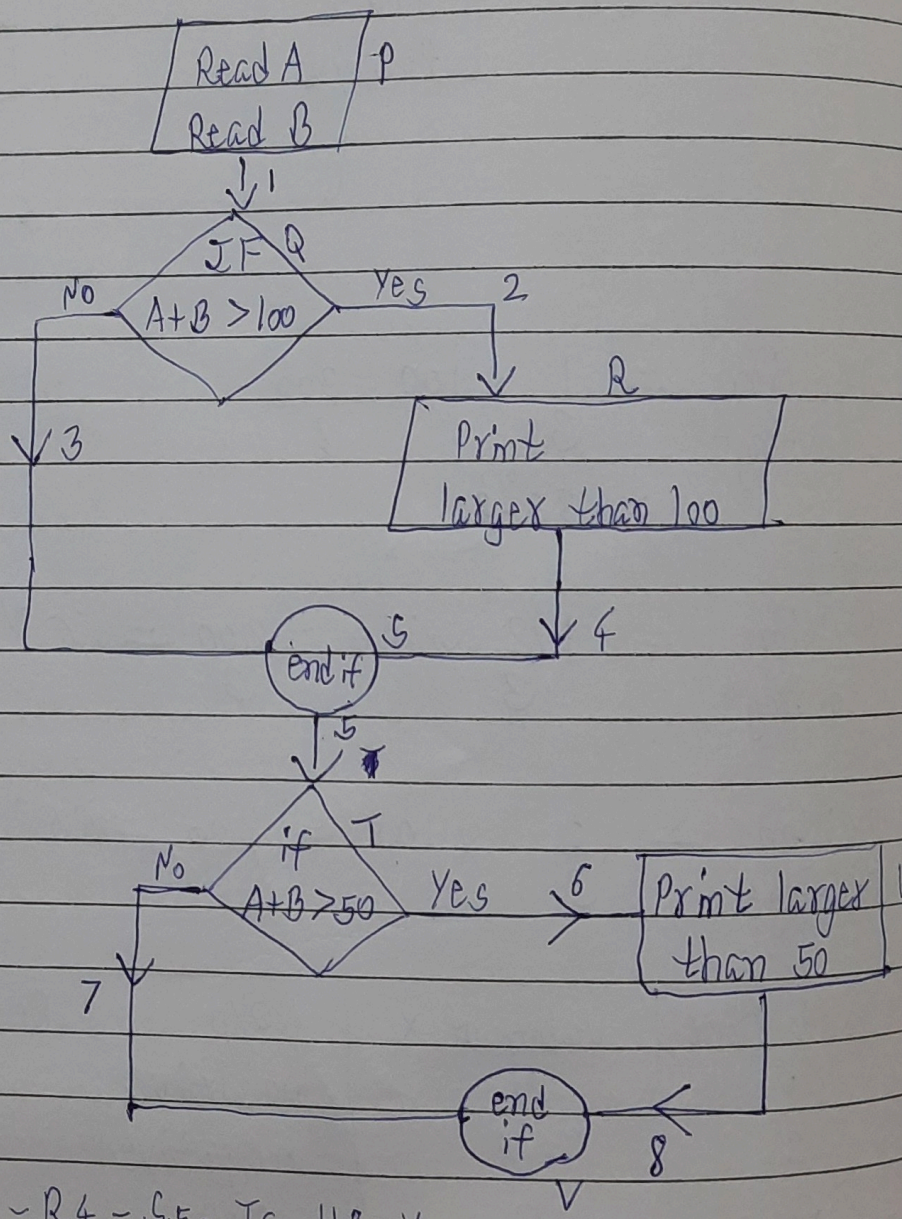
```
1   int num(x){
2     input x;
3     if (x>0)
4       print positive;
5     else if (x<0)
6       print negative;
7     else
8       zero
9   }
```

Page No.
Date / /

* Branch Coverage Testing

Ex 1)

```
1  Read A
2  Read B
3  IF A+B >100 THEN
4       print "larger than 100"
5  ENDIF
6  IF A+B >50 THEN
7       print "Larger than 50"
8  ENDIF
```



Yes - $P_1 - Q_2 - R_4 - S_5 - T_6 - U_8 - V$

No - $P_1 - Q_3 - S_5 - T_7 - V$

Number of path = 2

Page No
Date / /

* Branch coverage = $\frac{\text{no. of executed branches}}{\text{Total no. of branches}} \times 100$

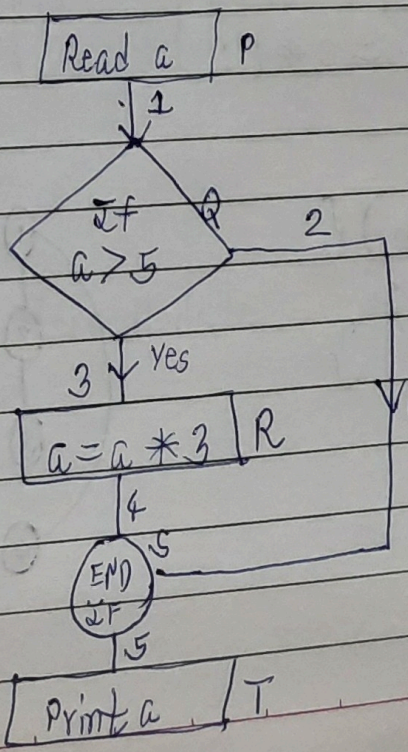
Branch Coverage (Yes) $= \frac{5}{8_2} \times \frac{100^{25}}{} = \frac{125}{2} = 62.5$

Branch Coverage (No) $= \frac{3}{8_2} \times \frac{100^{25}}{} = \frac{75}{2} = 37.5$

case	Covered branches	Path	Branch Coverage
Yes	2, 4, 5, 6, 8	P-Q2-R4-S5-T6-U8-V	62.5 %
No	3, 5, 7	P-Q3-S5-T7-V	37.5 %

2)

```
1  read a;
2  if (a > 5)
3      a = a * 3;
4  end if;
5     print(a);
```



Case	Covered branches	Path	Branch Coverage
Yes	3, 4, 5	P1 - Q3 - R4 - S5 - T	60%
No	2, 5	P1 - Q2 - S5 - T	40%

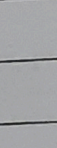
Branch coverage (Yes) $= \frac{3}{5} \times 100 = 60\%$

Branch coverage (No) $= \frac{2}{5} \times 100 = 40\%$

* Path Coverage :-

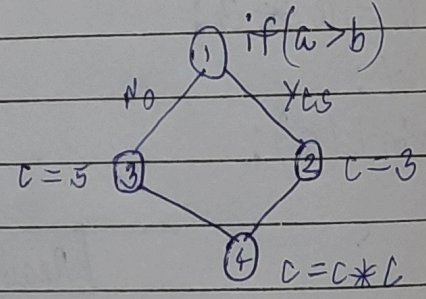
1) Sequence

```
1  a = 5;
2  b = a * b - 1;
```



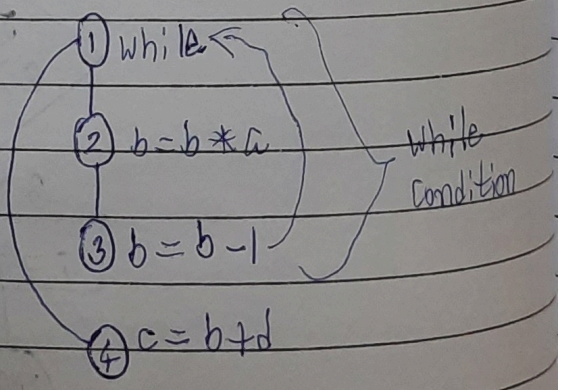
2) Selection Path

```
1  if (a > b) then
2      c = 3;
3  else c = 5;
4      ~~c = 5;~~ c = c * c;
```



3) Iteration

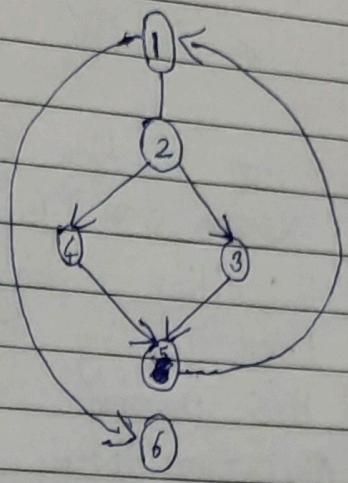
```
1  while (a > b) {
2      b = b * a;
3      b = b - 1; }
4  c = b + d;
```



Page No
Date

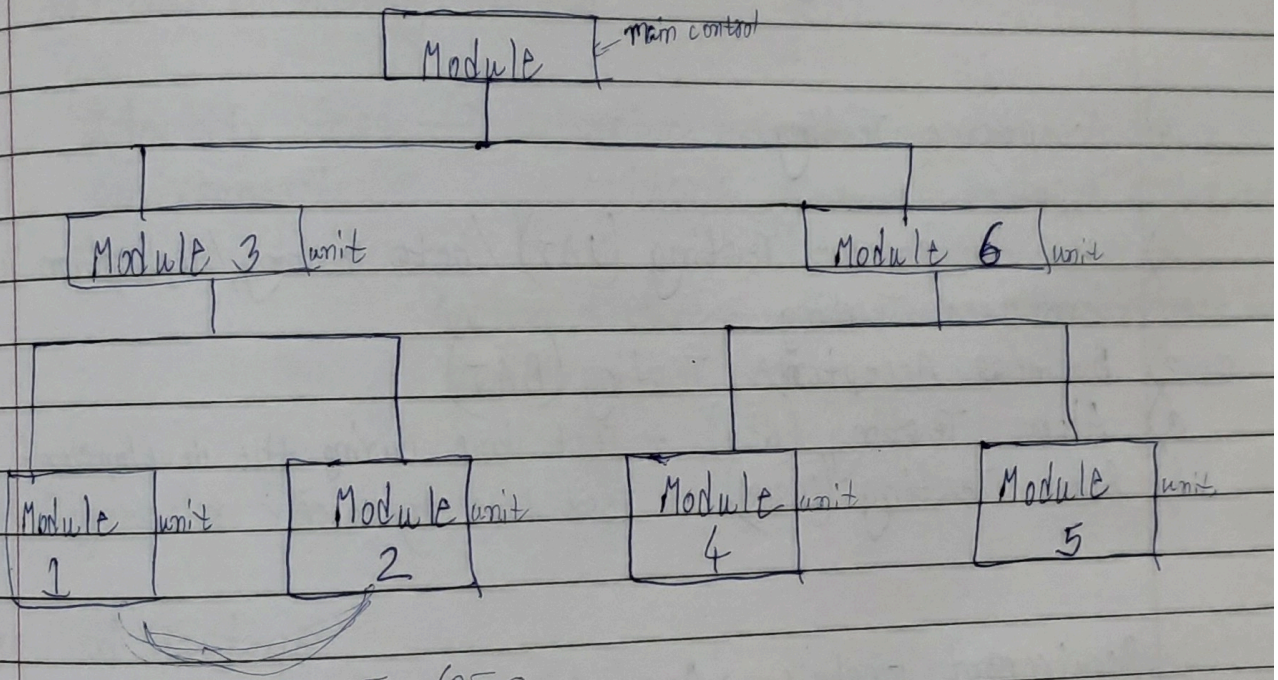
* Example for CFG (Euclid's gcd Algorithms)

```
int f1 (int x, int y) {
1  while (x != y) {
2      if (x > y) then
3          x = x - y
4      else y = y - x;
5  } print ans; }
6  return x; }
```



Software Testing (continue)

* Integration Testing



BFS/DFS

Note * Driver - connects two Module

#1) Topdown
#2) Bottom up

3) System Testing
→ H/w testing
→ S/w testing

★ System Testing Types
1) Regression Testing
2) Usability Testing - ★ simple to understand, easy to use.
3) Load Testing
4) H/s Testing
5) Migration Testing
6) Functional Testing
7) Recovery Testing

4) Acceptance Testing
Types
1) User Acceptance Testing (UAT) / beta Testing / Application Testing / end user Testing
2) Business Acceptance Testing (BAT)
3) Alpha Testing (dev) - Test done during the development
4) Beta Testing (user) - Test done by user

Requirement analysis → Acceptance Testing

High level design → System Testing

Low level design → Integration Testing

understand by machine

Coding → unit Testing

Fig. Placement of Acceptance Testing

5) Performance Testing

-> Scalability
-> stability
-> Speed

1) Load Testing
2) Stress Testing
3) Spike Testing
4) Configuration Testing
5) Soak Testing ~ accept all type of data

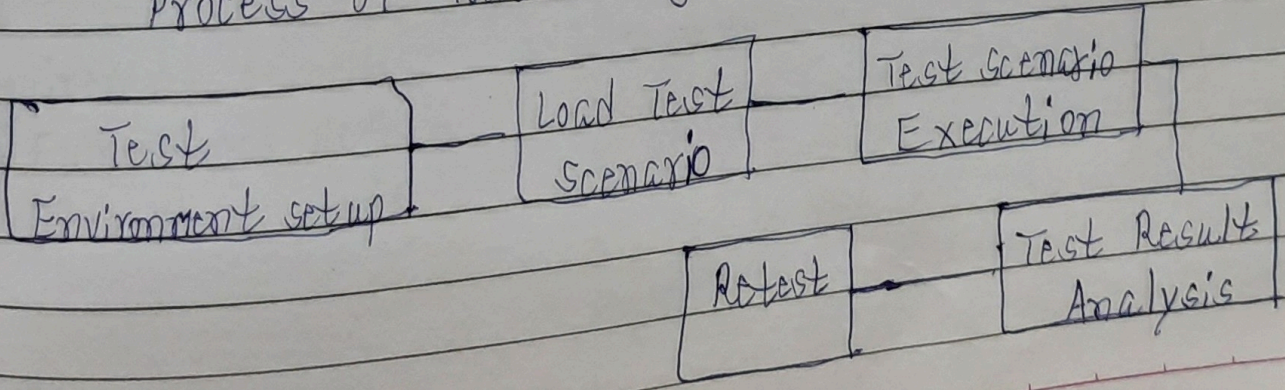
* Performance Testing Process

Identify test environment → Determine performance criteria (check Allocate, i/p, o/p) → Plan & Design → Configure test environment → Implement test Design → Run test → Analyze, finetune and retest

6) Regression Testing

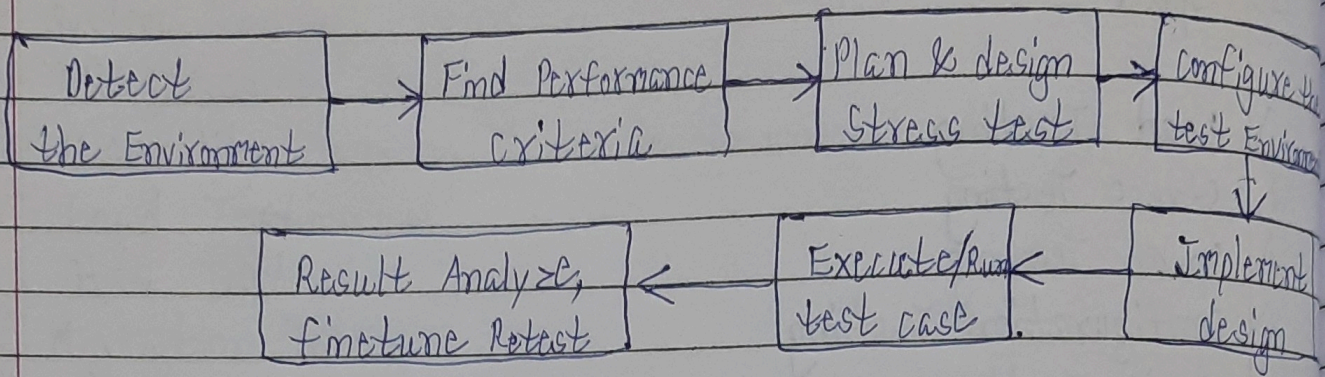
7) Load Testing

Process of load Testing



8) Stress Testing

Process of stress Testing



9) Security Testing

1) Confidentiality
2) Integrity
3) Authentication
4) Availability
5) Authorization
Non-repudiation