JULY						
M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

AUGUST						
M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2/11/22

MEETINGS / APPOINTMENTS

Assignment - II

Ans 1 This concept of cyclomatic complexity involves using cyclomatic numbers of graph theory which has been redefined as cyclomatic complexity, this is nothing but the numbers of independent paths through a program. Mclabs introduced this concept and gave three methods to calculate it:-

(i) $V(G) = e - n + 2p$

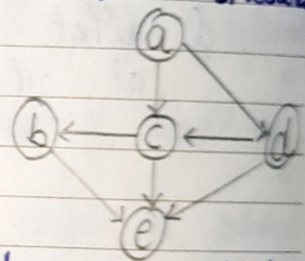
where; $V(G)$: Cyclomatic complexity
n : Numbers of nodes
e : Numbers of edges
p : Number of connected components

(ii) Equal to numbers of regions of program.

(iii) $V(G) = \pi + 1$

where; π : Numbers of predicate nodes in G,
But, the predicate node should only have two outgoing nodes (one for true and one for false) otherwise this is not valid.

Example



(i) $V(G) = e - n + 2p$
$= 7 - 5 + 2(1)$
$= 4$

(ii) There are 4 regions in graph inside and outside.

(iii) $V(G) = \pi + 1$
$= 3 + 1$
$= 4$

$\{\pi = 3$; i.e. a, c, d $\}$

NOTES

Ans 2

Testing	Debugging
• Testing is process to find bugs and errors.	Debugging is the process of correct bugs found in Testing
• It is display of error.	It is deductive process.

•	It is the process to identify the failture of implemented code.	It is the process to give the absolution to code failure.
•	It is done by testers.	It is either done by develop or programmer.
•	It is not needed to design knowledge in the testing proc.	It can't be done without proper degn knowledge.
•	It can be manual or automate	It can't be automated, it has to be manual.
•	It is based on different testing level ie. unit testing, integrate system etc.	It is based on different types of bugs.
•	It is a stage of software development life cycle (SDLC).	It is not an aspect of ~~of software~~ development life cycle, it occurs as a consequence of testing.
•	It is composed of validation and verification of software.	It seeks to match symptom to cause, by that it leads to the error correction.

Ans 3 This technique is a popular technique for small programs and considers the combinations of various input which were not available in techniques like Boundry Value Analysis and equivalence class testing. Such techniques do not allow combinations of inputs and considers all inputs as independent inputs. Two new terms are used here and these are cause & effect, which are nothing but input and output respectively.

Considered Example: Consider example of Keeping the record of material status and number of Children of a citizen. The value of marital status

NOTES

MEETINGS / APPOINTMENTS

must be 'y' or 'n'. The numbers of children must be digit or null, in case of a citizen is unmarried. If the information entered by the user is correct then an update is made. If the marital status of a user is incorrect, then error message1 is issued. Similarly, if the value of numbers of children is incorrect, then error message2 is issued.

The causes are:-

C_1: marital status is 'u'

C_2: marital status is 'm'

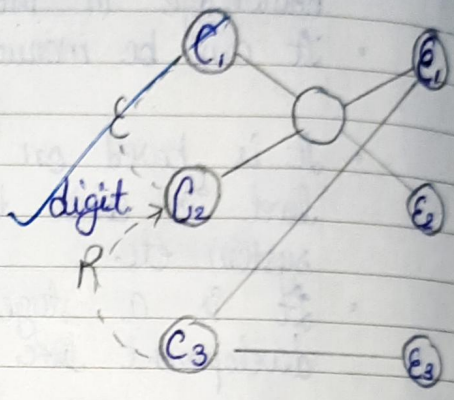
C_3: number of children is a digit

The effects are:-

E_1: updation made

E_2: error message1 is issued

E_3: error message2 is issued



The cause and effect graph is shown. There are two constraints exclusive, (between C1 & C2) and requires (between C3 and C2) which are placed at appropriate place in the graph. Cause C_1 & C_2 cannot occur simultaneously and for cause C_3 to be true, C_2 has to be true. However, there is no mask constraint in this graph.

Identification of all cause & effects

NOTES

Design the cause-effect graph

Apply constraints (if any)

Design limited entry decision table from graph

Write test cases

09 Ans 4 This is a simple but popular functional testing technique. Here, we concentrate on input values that are on or close
10 to boundary value. Experience has shown that such cases show a higher probability of detecting a fault in the
11 software. Suppose we have a programm which take input from 1-100. So, there is no need to check the program
12 for all the input from 1-100. The number of input selected by this technique is (4n+1) where 'n' is no. of inputs.
13 We also consider 'single fault' assumption theory of reliability which says that failure are rarely the result of simult-
14 aneous occurance of two(or more) faults. Normally, one fault is responsible for one failure. With this theory in mind,
15 we select one input value on boundary (minimum), just above boundary (maximum), nominal(average) and other n-1 inputs.
16 Therefore we select from 1-100:

- Minimum value (1),
17 - Just above minimum value (2),
- Maximum value (100),
18 - Just below maximum value (99), and
- Average (50).

19 Boundary value test for largest among 3 numbers [1-100]

Test Case	x	y	z	Expect Output
20 1	1	50	50	50
2	2	50	50	50
NOTES 3	50	50	50	50
4	99	50	50	99
5	100	50	50	100
6	50	1	50	50
7	50	2	50	50
8	50	99	50	99
9	50	100	50	100
10	50	50	1	50

MEETINGS / APPOINTMENTS

	11	50	50	2	50
	12	50	50	99	99
	13	50	50	100	100

Total cases are : 4(n)+1 {where n = input}
: 4(3)+1
: 13

Ans 5 CODE :-

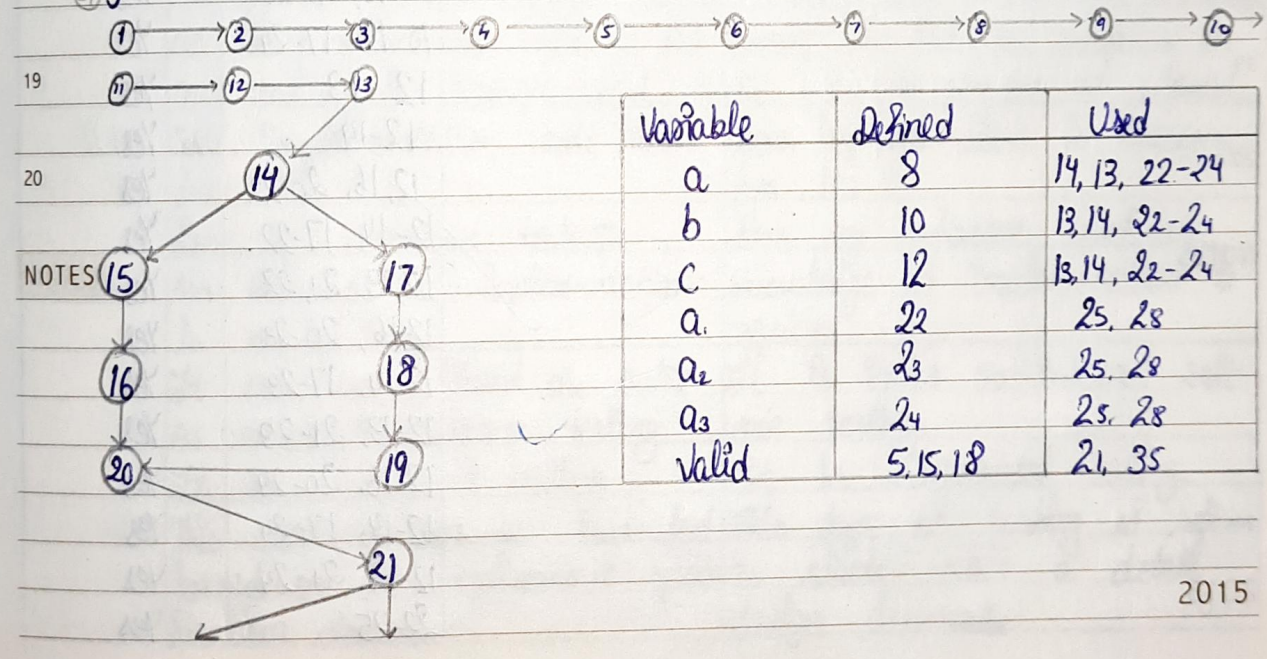
```
#include <stdio.h>
#include <conio.h>
#include <math.h>
① Void main()
② {
③ Double a, b, c;
④ double a1, a2, a3;
⑤ int valid = 0;
⑥ clrscr();
⑦ printf("Enter first side of the triangle");
⑧ scanf("%lf, &a);
⑨ printf("Enter second side of the triangle");
⑩ scanf("%lf, &b);
⑪ printf("Enter third side of the triangle");
⑫ scanf("%lf, &c);
⑬ if (a>0 && a<=100 && b>0 && b<=100 && c>0 && c<100)
⑭ {if [(a+b)>c && (b+c)>a && (c+a)>b)
⑮ { valid = 1
⑯ }
⑰ else{
⑱ valid = -1
⑲ }
⑳ }
```

NOTES

2015

MEETINGS / APPOINTMENTS

```
09  21  if (valid==1)
    22      {a1 = (a*a + b*b) | (c*c);
10  23       a2 = (b*b + c*c) | (a*a);
    24       a3 = (a*a + c*c) | (b*b);
11  25       if (a1<1 || a2<1 || a3<1)
    26           {printf("obtuse angled triangle");
12  27           }
    28       else
13  29           {printf("acute angled triangle");
    30           }
14  31      }
    32  else if (valid==-1)
15  33      {printf("Invalid triangle");
    34      }
16  35  else
    36      {printf("The input value are out of Range");
17  37      }
    38  getch();
18  39 }
```



Variable	Defined	Used
a	8	14, 13, 22-24
b	10	13, 14, 22-24
c	12	13, 14, 22-24
a_1	22	25, 28
a_2	23	25, 28
a_3	24	25, 28
valid	5, 15, 18	21, 35

NOTES

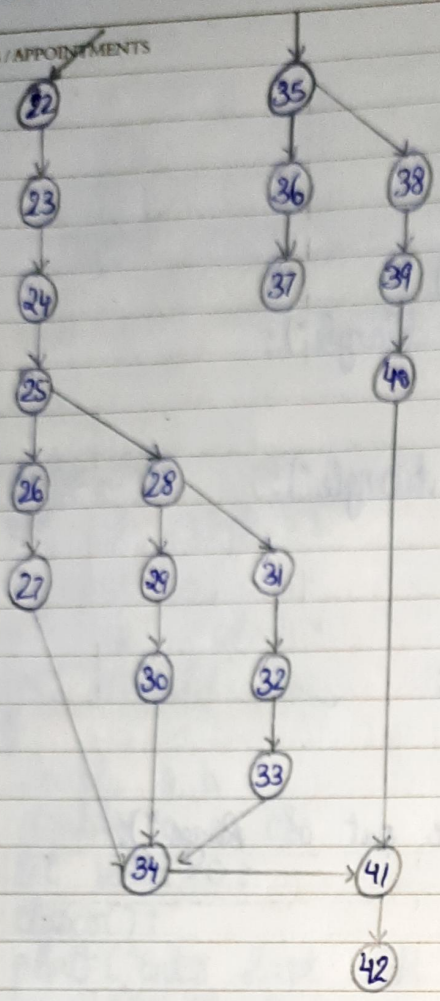
24 JULY
FRIDAY
Wk. 30-205-160

JULY
M T W T F S S
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

AUGUST
M T W T F S S
31 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

MEETINGS / APPOINTMENTS

09
10
11
12
13
14
15
16
17
18
19
20



All du-paths	Definition Class
8-13	Yes
8-14	Yes
8-16, 20-22	Yes
8-14, 11-22	Yes
8-13, 21-22	Yes
8-16, 20-23	Yes
8-14, 17-23	Yes
8-13, 21-23	Yes
8-16, 20-24	Yes
8-14, 17-24	Yes
8-13, 21-24	Yes
10-16, 20-22	Yes
10-14, 17-22	Yes
10-13, 21-22	Yes
10-16, 20-23	Yes
10-14, 17-22	Yes
10-13, 21-23	Yes
10-16, 20-24	Yes
10-13, 21-24	Yes
10-14, 17-24	Yes
12, 13	Yes
12-14	Yes
12, 16, 20-22	Yes
12-14, 17-22	Yes
12, 13, 21, 22	Yes
12-16, 20-23	Yes
12-14, 17-23	Yes
12-13, 21-23	Yes
12-16, 20-24	Yes
12-14, 17-24	Yes
12-13, 21-24	Yes
22-25	Yes

NOTES

2015

JULY

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

AUGUST

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MEETINGS / APPOINTMENTS

09

10

11

12

13

14

15

16

22-25, 28	Yes
23-25	Yes
23-25, 28	Yes
24, 25	Yes
24, 25, 28	Yes
5-16, 20, 21	No
5-14, 17-21	No
5-13, 21	Yes
5-16, 20, 21, 35	Yes
5-14, 17-21, 35	No
5-13, 21, 35	No
5, 16, 20, 21	Yes
5, 16, 20, 21, 35	Yes
18-21	Yes
18-21, 35	Yes

17 Ans 6 @

Black Box	White Box
• It is the way of software testing in which internal structure or the program or code is hid and nothing is known about it.	It is way of testing the software in which the tester has knowledge about the internal structure or the code or program of software.
• Code implementation isn't needed for it.	Code implementation is necessary for it.
• Done by software testers.	Done by software developer.
• No knowledge of implementation is needed.	Knowledge of implementation is required.
• It can be referred as outer or external software testing.	It is inner or internal software testing.
• It is functional testing.	It is structural testing.
• This testing can be initiated based on the requirement specification document.	This type of testing of software is started after a detail design document.

NOTES

JULY

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

AUGUST

M	T	W	T	F	S	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

MEETINGS / APPOINTMENTS

09 • Knowledge of programing isn't requied.
10 • It is behaviours testing of software
• It is appliable to higher level of testing of software.
11

It is mandatoory to have the programing knowledge.
It is logical testing of software.
It is general applicable to the lowers level of software testing

12

(b)
```
#include <stdio.h>
#include <conio.h>
void main()
{
float A, B, C;
clrscr();
printf("Enter Number 1 : \n");
scanf("%f, &A);
printf("Enter number 2 : \n");
scanf("%f, &B);
printf("Enter number 3 : \n");
scanf("%f, &C);
if (A>B)
    {if (A>C)
        {printf("The largest number is %f \n", A);
        }
    else
        {printf("The largest number is %f \n", C);
        }
    }
else
    {if (B<C)
        {printf("The largest number is %f \n", C);
        }
    else
```

NOTES

2015

MEETINGS / APPOINTMENTS

```
printf("The largest number is %d\n", B);
}
getch();
```

S.No.	A	B	C	Expected Output	Remarks
1	9	8	7	9	6-11
2	9	8	7	9	6-12
3	9	8	7	9	6-13
4	7	9	8	9	8-11
5	7	9	8	9	8-11, 19, 20
6	7	9	8	9	8-11, 19, 20, 23, 24
7	8	7	9	9	10-12
8	8	7	9	9	10-12, 15, 16
9	8	7	9	9	10, 11, 19, 20

White Box Testing

S.No.	A	B	C	Expected Output	Remark
1	1	50	50	50	
2	2	50	50	50	
3	50	50	50	50	
4	99	50	50	99	
5	100	50	50	100	
6	50	1	50	50	
7	50	2	50	50	
8	50	99	50	99	
9	50	100	50	100	
10	50	50	1	50	
11	50	50	2	50	
12	50	50	99	99	
13	50	50	100	100	

NOTES

MEETINGS / APPOINTMENTS

Ans 7 Program slicing was introduced by mark weise where we prepare various subsets (called) (slice) of a program with respect to its variable and their selected locations, in the program. Each variable with one of its location will give us a program slice.
"Program slicing is a technique for restricting the behaviour of a program to some specified subset of interest. A slice S(V, n) of program P on variable V, or set of variables, at statement n yeilds the portion of the program that contributed to the value of V just before statement n is executed."
Example:-

```
void main()
{
  int a, b, c, d, e
  printf("Enter the value of a, b and c \n");
  scanf("%d %d %d", &a, &b, &c);
  d = a + b
  e = b + c
  printf(%d, d);
  printf(%d, e);
}
```

s(e, 10) = (1, 2, 3, 4, 5, 7, 9, 10)
s(e, 7) = (1, 2, 3, 4, 5, 7, 10)
s(d, 10) = (1, 2, 3, 4, 5, 6, 8, 10)
s(d, 6) = (1, 2, 3, 4, 5, 6, 10)
s(a, 5) = (1, 2, 3, 4, 5, 10)

NOTES

Ans 8 Decision table are used in many engineering disciplines to represent complex logical relationships. An output may be dependent on many input conditions and

2015

MEETINGS / APPOINTMENTS

09 Decisions tables give a pictorial view of various combinations of input conditions. These are few portion of the decision table.

10 1. Condition study
2. Condition entries
3. Action stubs
11 4. Action entries

Stubs	Entries
C_1	
C_2	
C_3	
a_1	
a_2	
a_3	
a_4	

12 13 14 Let consider the program "Classification of Triangle".

15

C_1: $a<b+c$	F	T	T	T	T	T	T	T	T	T	T	T
C_2: $b<c+a$	m	F	T	T	T	T	T	T	T	T	T	T
C_3: $c<a+b$	m	m	F	T	T	T	T	T	T	T	T	T
C_4: $a=b$	m	m	m	T	T	T	F	F	F	F	F	F
C_5: $b=c$	m	m	m	T	F	F	T	T	F	F	T	T
C_6: $c=a$	m	m	m	T	F	F	T	F	T	F	T	F
Rule Count	32	16	8	1	1	1	1	1	1	1	1	1
a_1: not a triangle	X	X	X									
a_2: Scalene												X
a_3: Isosceles						X	X	X	X	X	X	
a_4: Equilateral				X								
a_5: Impossible					X	X	X	X				

16 17 18 19 20

NOTES

MEETINGS / APPOINTMENTS

09 The 'do not condition' are represented by the '-' sign.
A "do not care" condition has no effect on the output
10 If we refers to column one of the decision table
where condition $C_1: a < b + c$ is false, then other entry
11 become "do not care" entries. If C_1 is false the output
will be invalid triangle irrespective of any state (true & false)
12 These condition become do not care condition and are
represented by (-) sign. If we don't do so and repre-
13 sented all true and false entries of every condition, the
decision table will unnecessarily increase.
14 This is nothing but a representation facility in the
decision table to reduce the number of columns and
15 avoid redundancy. Ideally each column has one rule and
that leads to a test case.
16 The term rule count is used with "do not care" entries
in the decision table and has a value 1, if "do not
17 care" condition is not there, but doubles for each
"do not care" entry.
18 Rule Count = 2 (Number of 'do not care' condition)

19

20

NOTES

Good

15/11/22