Test Case ID	Test Description	Assumptions & Pre conditions	Test Data	Expected Result	Pass / Fail
11.	Giving email as input for email ID	—@—.— Format of email	S@gmail.in (or) S@gmail.edu	valid valid	Pass Pass
12.	Giving email as input for email ID	—@—.— format of email	S@g.i (or) @g.in	invalid invalid	fail fail
13.	Uploading Resume.	~~Size of~~ file should be in pdf format. size should be 50 KB.	uploading pdf with 42 KB	valid	Pass
14.	Uploading Resume.	file should be in pdf format. size should be 50 KB.	uploading docx.	Invalid.	fail.
15.	uploading photo.	Photo should be in PNG (or) JPEG format and size should be of 50 KB.	uploading photo of 40 KB in PNG Format	valid	Pass
16.			uploading photo of 53 KB in JPEG format	invalid	fail

Test ID	Conditions			Expected output	Preconditions / classes covered by the test
	DD	MM	YYYY		
1.	12	12	2003	Valid	I_0, I_1, I_2
2.	12	0	2003	invalid	I_3
3.	01	13	2003	invalid	I_4
4.	0	8	2002	invalid	I_5
5.	32	8	2002	invalid	I_6.
6.	1	12	1899	invalid	I_7
7.	1	4	2026	invalid	I_8

EXPERIMENT-3

(2) A program determines the next date in the calendar. Its input is entered in the form of <ddmmyyyy> with the following range.

$1 \leq mm \leq 12$
$1 \leq dd \leq 31$
$1900 \leq yyyy \leq 2025$.

Its output would be the next date or an error message 'invalid date'. Design test cases using equivalence class partitioning method.

Pre-conditions & classes :

I_0 $1 \leq mm \leq 12$
I_1 $1 \leq dd \leq 31$
I_2 $1900 \leq yyyy \leq 2025$
I_3 $1 > mm$
I_4 $mm > 12$
I_5 $1 > dd$
I_6 $dd \geq 31$
I_7 $1900 > yyyy$
I_8 $yyyy > 2025$

Test case ID	A	B	C	Expected result	classes covered by the test cases.
1	13	25	36	C is greatest	I_1, I_2, I_3
2	0	13	45	Invalid input	I_4
3	51	34	17	Invalid input	I_5
4.	29	0	18	Invalid input	I_6
5.	36	53	32	Invalid input	I_7
6.	27	42	0	Invalid input	I_8
7.	33	21	51	Invalid input	I_9

(2) First we partition the domain of input as valid input values and invalid values, getting the following classes D^x

(2) A program reads three numbers, A, B and C with a range [1, 50] and prints the largest number. Design test cases for this program using equivalence class testing technique.

Soln:

$I_1 = \{\langle A, B, C\rangle : 1 \le A \le 50\}$

$I_2 = \{\langle A, B, C\rangle : 1 \le B \le 50\}$

$I_3 = \{\langle A, B, C\rangle : 1 \le C \le 50\}$

$I_4 = \{\langle A, B, C\rangle : A < 1\}$

$I_5 = \{\langle A, B, C\rangle : A > 50\}$

$I_6 = \{\langle A, B, C\rangle : B < 1\}$

$I_7 = \{\langle A, B, C\rangle : B > 50\}$

$I_8 = \{\langle A, B, C\rangle : C < 1\}$

$I_9 = \{\langle A, B, C\rangle : C > 50\}$

Test case ID	Angle	Expected results	Classes Covered by the test Case
1	50	1 Quadrant	I_1
2	-1	Invalid Input	I_2
3	361	Invalid input	I_3

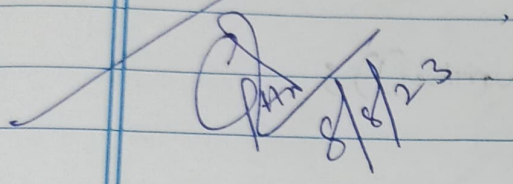
③ A program takes an angle as input within the range [0,360] and determines in which quadrant the angle lies. Design test cases using equivalence class partitioning method.

Sol:

$I_1 = \{ <Angle> : 0 \leq Angle \leq 360 \}$

$I_2 = \{ <A,B,C> : Angle < 0 \}$

$I_3 = \{ <A,B,C> : Angle > 0 \}$

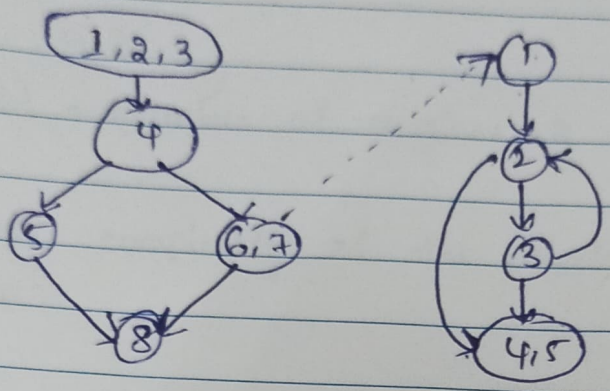


```
main()
{
    int number;
    int fact();
1.  clrscr()
2.  printf("Enter the number whose factorial is to be found out");
3.  scanf("%d", &number);
4.  if (number<0)
5.      printf("factorial can't be defined for this number");
6.  else
7.      printf("factorial is %d", fact(number));
8. }
9. int fact(int number)
   {
1.     int index;
2.     int product = 1;
3.     for (index = 1; index <= number; index++)
           product = product * index;
4.     return(product);
5. }
```

Experiment - 4

Consider the program for calculating the factorial of a number. It consists of main() program and the module fact(). Calculate the individual cyclomatic complexity number for main() & fact() and then, the cyclomatic complexity for the whole program.

Sol: DD graph.



a) Flow graph for main() b) Flow graph for fact()

cyclomatic complexity of main()

(a) V(m) = e - n + 2
= 5 - 5 + 2
= 2.

(b) V(M) = d + p
= 1 + 1
= 2.

(c) V(M) = Number of regions = 2.

cyclomatic complexity of fact()

(a) V(R) = e - n + 2p
= 4 - 4 + 2
= 2.

(b) V(R) = Number of predicate nodes + 1
= 1 + 1
= 2.

(c) V(R) = Number of regions = 2.

Cyclomatic complexity of the whole graph considering the full program:

(a) V(G) = e - n + 2p
= 9 - 9 + 2 × 2
= 4
= V(M) + V(R)

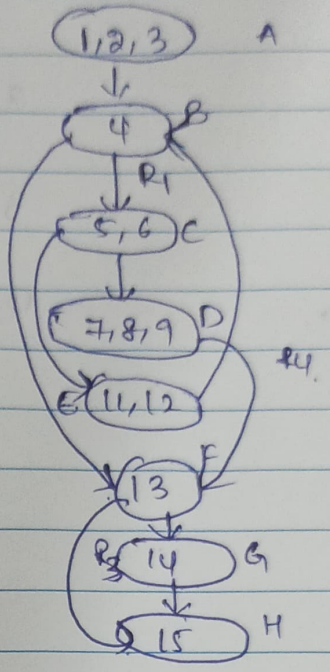
(b) V(G) = d + p
= 2 + 2
= 4
= V(M) + V(R)

(c) V(G) = Number of regions
= 4
= V(M) + V(R).

```
main()
{
    int number, index;
1.  printf("Enter a number");
2.  scanf("%d", &number);
3.  index = 2;
4.  while (index <= number-1)
5.  {
6.      if (number % index == 0)
7.      {
8.          printf("Not a prime number");
9.          break;
10.     }
11.     index ++;
12.     }
13.     if (index == number") {
14.             printf("prime number");
15.     } //end main.
```

Consider the program segment

1. DD graph
2. Cyclomatic complexity



① V(G) = e - n + 2 * p
= 10 - 8 + 2
= 4

② V(G) = Number of predicate nodes + 1
= 3 + 1
= 4.

③ V(G) = Number of regions
= 4 (R1, R2, R3, R4)

3. Independent path

Since cyclomatic complexity is 4, test cases are 4.

1. A-B-F-H
2. A-B-F-G-H
3. A-B-C-E-B-F-G-H
4. A-B-C-D-F-H

Test cases:

Test case ID	Input form	Expected result	path covered
1.	1.	No output displayed	A-B-F-H
2.	2	prime number	A-B-F-G-H
3.	3	Not a prime	A-B-C-D-F-
4.	4.	prime number	A-B-C-E-B- G-H

INDEX

Serial No.	Date	Name of the Experiment	Page No.	Marks Awarded	Remarks
1.	25/7/23	BVA, RT, WCT	1 -		
2.	1/8/23	Webpage test cases			
3.	8/8/23	Equivalence partitioning testing			
4.	29/8/23	Cyclomatic complexity			