RV Educational Institutions®

RV Institute of Technology and Management

(Affiliated to VTU, Belagavi)

JP Nagar 8th Phase, Bengaluru - 560076

Department of

Information Science and Engineering

Course Name: Software Testing Laboratory

Course Code: 21ISL66

VI Semester

2021 Scheme

Prepared By :

Mrs. Swathi Darla
Assistant Professor,
Department of ISE
Email: swathidarla.rvitm@rvei.edu.in

	PART-A
SL.NO	PROGRAMS
1	Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.
2	Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.
3	Design, develop, code and run the program in any suitable language to solve the commissi problem. Analyze it from the perspective of decision table-based testing,derive different te cases, execute these test cases and discuss the test results.
4	Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of atriangle and determine if the three values represent an equilateral triangle, isoscelestriangl scalene triangle, or they do not form a triangle at all. Assume that the upper limitfor the siz of any side is 10. Derive test cases for your program based on boundary-value analysis, equivalence class partitioning and decision-table approach and execute thetest cases and discuss the results.
5	Design, develop, code and run the program in any suitable language to solve thecommissio problem. Analyze it from the perspective of dataflow testing, derive different test cases, execute these test cases and discuss the testresults.
6	Design, develop, code and run the program in any suitable language to implement the bina search algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.
	PART B – Practical Based Learning
	Develop a Mini Project with documentation of suitable test-cases and their results to perform automation testing of any e-commerce or social media web page. Suggested Guidelines: ● Create a WebDriver session. ● Navigate to a Web page. ● Locate the web elements on the navigated page. ● Perform an actions on the located elements. ● Assert the performed actions did the correct thing. ● Report the results of the assertions. ● End the session.

	Each inputs / data feeds (ex: website, username, password, mobile no, product name, etc.,)must be provided through a file linked with code and neither to be entered manually nor to be included in the code Use any software testing tool like selenium, Katalon, etc

Course Outcome (Course Skill Set)

At the end of the course the student will be able to:

CO 1. List out the requirements for the given problem and develop test cases for any given problem .

CO 2. Design and implement the solution for given problem and to design flow graph

CO 3. Use the appropriate functional testing strategies. Compare the different testing techniques.

CO 4. Classify and Compare the problems according to a suitable testing model applying the test coverage metrics.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

- CIE marks for the practical course is 50 Marks.
- The split-up of CIE marks for record/ journal and test are in the ratio 60:40.
- Each experiment to be evaluated for conduction with observation sheet and record write-up.
- Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment

- write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.

The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book The average of 02 tests is scaled down to 20 marks (40% of the maximum marks).The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination. (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated

for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

- Students can pick one experiment from the questions lot of PART A with equal choice to all the students in a batch.PART B : Student should develop a mini project and it should be demonstrated in the laboratory examination (with report and presentation).
- The weightage of marks for PART A is 60% and for PART B is 40%. General rubrics suggested to be followed for part A and part B.
- Change of experiment is allowed only once (in part A) and marks allotted to the procedure part to be made zero.
- The duration of SEE is 03 hours.

Lab Programs

1. Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.

```
#include<stdio.h>
main()
{
    int locks, stocks, barrels, t_sales;float commission;
    printf("Enter the total number of locks");scanf("%d",&lock);
    printf("Enter the total number of stocks");scanf("%d",&stock);
    printf("Enter the total number of barrelss");scanf("%d",&barrels);
    if ((locks <= 0) || (locks> 70)||(stocks <= 0) || (stocks > 80)|| (barrels <= 0) || (barrels >
    90))
    {
    if(lock==-1)
    Printf("program terminates");else
    printf("invalid input");
    }
    else
    {
    t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);if (t_sales <= 1000)
    {
    commission = 0.10 * t_sales;
    }
    else if (t_sales < 1800)
    {
    commission = 0.10 * 1000;
    commission = commission + (0.15 * (t_sales - 1000));
    }
    else
    {
    commission = 0.10 * 1000;
    commission = commission + (0.15 * 800); commission = commission + (0.20 * (t_sales -
    1800));
    }
    printf("The total sales is %d \n the commission is %f",t_sales, commission);
    }
```

```
return 0;
}
```

BOUNDARY VALUE ANALYSIS

Test Case Name: Boundary Value Analysis for Commission problem

Test Data: Enter the 3 Integer Value (locks, Stocks and barrels)

Pre-condition: 1<=Locks<=70 {1,2,35,70,69}

1<=Stocks<=80 {1,2,40,80,79}

1<=Barrels<=90 {1,2,45,90,89}

No of test cases=(4n+1, n is the number of inputs) {min, min+, norm, max, max-}

Case id V A)	Description	Input			Expected output		Actual output		status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1	Barrels is min	35	40	1	2800	420.00			
2	Barrels is min+	35	40	2	2825	425.00			
3	Barrels is norm	35	40	45	3900	640.00			
4	Barrels is max	35	40	90	5025	865.00			
5	Barrels is max-	35	40	89	5000	860.00			
6	Stocks is min	35	1	45	2730	406.00			
7	Stocks is min+	35	2	45	2760	412.00			
8	Stocks is norm	35	40	45	3900	640.00			
9	Stocks is max	35	80	45	5100	880.00			
10	Stocks is max-	35	79	45	5070	874.00			

Robust testing: Test Cases (min-, max+)

Case id	Description	Input data			Expected output		Actual output		Status
		Locks	Stocks	Barrels	Sales	commission	Sales	Commission	
1	Locks invalid	-2	40	45	Invalid Input				
2	Locks invalid	71	40	45	Invalid Input				
3	Stocks invalid	35	-2	45	Invalid Input				
4	Stocks invalid	35	81	45	Invalid Input				
5	Barrels invalid	35	40	-2	Invalid Input				
6	Barrels invalid	35	40	91	Invalid Input				

Special Value Testing: Test Cases(Validated test cases)

Case id	Description	Input			Expected output		Actual output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1	Less than 1800	1	40	1	1270	275.00			
2	Less than 1000	1	30	1	970	97.00			
3	Greater than 180(	1	1	90	2325	325.00			

Worst Case Testing: Test Cases(min,max, nom,min-,max+)

TestCaseid	Description	Input			Expected output		Actual output		status
		Lock	Stocks	Barrels	Sale	Commission	Sales	Commission	
1	Cartesian product of5 elementset.	1	2	1	130	13			
2		1	2	2	155	15.5			
3		1	2	45	123(	134.5			
4		1	2	90	235	331			
5		1	2	89	233(	326			
6	Cartesian product of5 elementset.	1	1	1	100	10			
7		1	1	2	125	12.5			
8		1	1	45	120(	130			
9		1	1	90	232	325			
10		1	1	89	230(	320			
11	Cartesian product of5 elementset.	1	40	1	127(	140.5			
12		1	40	20	174	211.75			
13		1	40	45	237(	334			
14		1	40	90	349	559			
15		1	40	89	347(	554			
16		1	80	1	247(	354			

17	Cartesian product of5 elementset.	1	80	2	2495	359			
18		1	80	45	3570	574			
19		1	80	90	4695	799			
20		1	80	89	4670	794			
21	Cartesian product of5 elementset.	1	79	1	2440	348			
22		1	79	2	2465	353			
23		1	79	45	3540	568			
24		1	79	90	4665	793			
25		1	79	89	4640	788			

2.Design, develop, code and run the program in any suitable language to implement theNext Date function. Analyze it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>
#include <stdlib.h>
void main( )
{
int month[12]={31,28,31,30,31,30,31,31,30,31,30,31};
int i,d,m,y,nd,nm,ny,ndays;
printf("enter the date,month,year"); scanf("%d%d%d",&d,&m,&y);
if((y%100!=0)&&(y%4==0)||(y%400==0))
{
month[i]=29;
}
ndays =month [m-1];
if((y<1812) || ( y>2012) || (d<1) || (d>ndays)|| (m<1) || ( m>12))
{
```

```
printf ("invalid input");
}
else
{
if(m==2)
{ if(y%100==0)
{ if(y%400==0)
ndays=29;
}
else if(y%4==0) ndays=29;
}
nd=d+1; nm=m; ny=y; if(nd>ndays)
{
nd=1; nm++;
}
if(nm>12)
{
nm=1; ny++;
}
printf("\n Given date is %d:%d:%d",d,m,y); printf("\n Next day date is %d:%d:%d",nd,nm,ny);
}
return 0;
}
```

EQUIVALENCE CLASS TESTING

First Attempt

A first attempt at creating an equivalence relation might produce intervals Such as these:

Valid Equivalence Classes

M1 ={month: 1 ≤month ≤ 12} D1 ={day: 1 ≤day ≤31}Y1 = {year: 1812 ≤ year ≤2012}

Invalid Equivalence Classes

M2 = {month: month < 1} M3 = {month: month > 12} D2 =
{day: day < 1}D3 = {day: day > 31}
Y2 = {year: year < 1812} Y3 = {year: year > 2012}

At a first glance it seems that everything has been taken into account and our day, month and year intervals have been defined well. Using these intervals we produce test cases using the four different types of Equivalence Class testing.

Weak Normal and Strong Normal Equivalence Classes Test Cases

Case id	Description	Input			Expected output	Actual output	status
		Day	Month	Year			
1	Enter the M1, D1 and Y1 valid cases	15	6	1912	16-6-1912		

Weak Robust Equivalence Classes Test Cases

Case	Description	Input			Expected	Actual output	status
id		Day	Month	Year	output		
1	Valid input	15	6	1912	Invalid input		
2	Day is invalid	-1	6	1912	Invalid input		
3	Day is invalid	32	6	1912	Invalid input		
4	Month is invalid	15	-1	1912	Invalid input		
5	Month is invalid	15	13	1912	Invalid input		
6	Year is invalid	15	6	1811	Invalid input		
7	Year is invalid	15	6	2013	Invalid input		

Strong Robust Equivalence Classes Test Cases

T C ID	TC Description	Input			Expected output	Actual output	status
		Day	Month	Year			

SR1	Invalid input	-1	6	1912	Invalid input		
SR2	Invalid input	15	-1	1912	Invalid input		
SR3	Invalid input	15	6	-1	Invalid input		
SR4	Invalid input	-1	-1	1912	Invalid input		
SR5	Invalid input	-1	6	1912	Invalid input		
SR6	Invalid input	15	-1	-1	Invalid input		
SR7	Invalid input	-1	-1	-1	Invalid input		

Second Attempt

As said before the equivalence relation is vital in producing useful test cases and more time must be spent on designing it. If we focus more on the equivalence relation and consider moregreatly what must happen to an input date we might produce the following equivalence classes:

M1 = {month: month has 30 days} M2 = {month: month has 31 days} M3 = {month: month isFebruary}

Here month has been split up into 30 days (April, June, September and November), 31 days (January,March, April, May, July, August, October and December) and February.

D1 = {day: $1 \leq$ day ≤ 28}
D2 = {day:day = 29}
D3 = {day: day =30}
D4 = {day:day = 31}

Day has been split up into intervals to allow months to have a different number of days; wealso have the special case of a leap year (February 29days).

Y1 = {year: year = 2000}
Y2 = {year: year is a leap year}
Y3 = {year: year is a common year}

Year has been split up into common years, leap years and the special case the year 2000 so we can determine the date in the month of February. Here are the test cases for the new equivalence relation using the four types of Equivalence Class testing.

Weak Normal Equivalence Classes Test Cases

Test case id	Description	Input			Expected output	Actual output	status
		Day	Month	Year			
WN1	Valid input	14	6	2000	14-6-2000		
WN2	Valid input	29	7	1996	29-7-1996		
WN3	Valid input	30	2	2000	30-2-2000		
WN4	Valid input	31	6	2000	31-6-2000		

Strong Normal Equivalence Classes Test Cases

(No. of test cases =3*4*3=36 Where 3= no of month class; 4= no of day class; 3=no of yearclass respectively)

Test case id	Description	Input			Expected output	Actual output	status
		Day	Month	Year			
SN1	Valid input	14	6	2000	Valid input		
SN2	Valid input	14	6	1996	Valid input		
SN3	Valid input	14	6	2002	Valid input		
SN4	Valid input	29	6	2000	Valid input		
SN5	Valid input	29	6	1996	Valid input		
SN6	Valid input	29	6	2002	Valid input		
SN7	Valid input	30	6	2000	Valid input		
SN8	Valid input	30	6	1996	Valid input		
SN9	Valid input	30	6	2002	Valid input		
SN10	Valid input	31	6	2000	Valid input		

SN11	Valid input	31	6	1996	Valid input		
SN12	Valid input	31	6	2002	Valid input		
SN13	Valid input	14	7	2000	Valid input		
SN14	Valid input	14	7	1996	Valid input		
SN15	Valid input	14	7	2002	Valid input		
SN16	Valid input	29	7	2000	Valid input		
SN17	Valid input	29	7	1996	Valid input		
SN18	Valid input	29	7	2002	Valid input		
SN19	Valid input	30	7	2000	Valid input		
SN20	Valid input	30	7	1996	Valid input		
SN21	Valid input	30	7	2002	Valid input		
SN22	Valid input	31	7	2000	Valid input		
SN23	Valid input	31	7	1996	Valid input		
SN24	Valid input	31	7	2002	Valid input		
SN25	Valid input	14	2	2000	Valid input		
SN26	Valid input	14	2	1996	Valid input		
SN27	Valid input	14	2	2002	Valid input		
SN28	Valid input	29	2	2000	Valid input		
SN29	Valid input	29	2	1996	Valid input		
SN30	Valid input	29	2	2002	Valid input		
SN31	Valid input	30	2	2000	Valid input		
SN32	Valid input	30	2	1996	Valid input		
SN33	Valid input	30	2	2002	Valid input		

SN34	Valid input	31	2	2000	Valid input		
SN35	Valid input	31	2	1996	Valid input		

Invalid Equivalence Classes

M4 = {month: month < 1} M5 = {month: month > 12}
D5={day: day < 1}D6 = {day: day > 31}
Y4 = {year: year < 1812} Y5 = {year: year > 2012}

Weak Robust Equivalence Classes Test Cases

Test case id	Description	Input data			Expected output	Actual output	status
		Day	Month	Year			
WR1	Valid input	15	6	1912	Invalid input		
WR2	Day is invalid	-1	6	1912	Invalid input		
WR3	Day is invalid	32	6	1912	Invalid input		
WR4	Month is invalid	15	-1	1912	Invalid input		
WR5	Month is invalid	15	13	1912	Invalid input		
WR6	Year is invalid	15	6	1811	Invalid input		
WR7	Year is invalid	15	6	2013	Invalid input		

Strong Robust Equivalence Classes TestCases

Test case id	Description	Input data			Expected output	Actual output	status
		Day	Month	Year			
SR1	Invalid input	-1	6	1912	Invalid input		
SR2	Invalid input	15	-1	1912	Invalid input		
SR3	Invalid input	15	6	-1	Invalid input		
SR4	Invalid input	-1	-1	1912	Invalid input		
SR5	Invalid input	-1	6	1912	Invalid input		
SR6	Invalid input	15	-1	-1	Invalid input		
SR7	Invalid input	-1	-1	-1	Invalid input		

3.Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of decision table-based testing,derive different test cases, execute these test cases and discuss the test results.

```
#include <stdio.h>
#inclde <stdlib.h>
int main()
{
int locks, stocks, barrels, t_sales; float commission;
printf("Enter the total number of locks"); scanf("%d",&locks);
printf("Enter the total number of stocks"); scanf("%d",&stocks);
printf("Enter the total number of barrelss"); scanf("%d",&barrels);
if ((locks <= 0) || (locks> 70)||(stocks <= 0) || (stocks > 80)|| (barrels <= 0) || (barrels > 90))
{
if(locks==-1) printf("program terminates"); else
printf("invalid input");
}
else
{
t_sales = (locks * 45) + (stocks * 30) + (barrels * 25); if (t_sales <= 1000)
{
commission = 0.10 * t_sales;
}
else if (t_sales < 1800)
{
commission = 0.10 * 1000;
commission = commission + (0.15 * (t_sales - 1000));
}
else
{
commission = 0.10 * 1000;
commission = commission + (0.15 * 800); commission = commission + (0.20 * (t_sales - 1800));
}
printf("The total sales is %d \n the commission is %f",t_sales, commission);
}
return 0;  }
```

DECISION TABLE

Test data: price Rs for lock - 45.0, stock - 30.0 and barrel - 25.0
Sales = total lock * lock price + total stock * stock price + total barrel * barrel price
Commission: 10% up to sales Rs 1000, 15 % of the next Rs 800 and
20 % onany sales in excess of 1800.

Pre-condition: lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

Brief Description: The salesperson had to sell at least one complete rifle per month.

RULES		R1	R2	R3	R4	R5	R6	R7	R8	R9
Conditions	C1: Locks = -1	T	F	F	F	F	F	F	F	F
	C2 : 1 ≤ Locks ≤ 70	-	T	T	F	T	F	F	F	T
	C3 : 1 ≤ Stocks ≤ 80	-	T	F	T	F	T	F	F	T
	C4 : 1 ≤ Barrels ≤ 90	-	F	T	T	F	F	T	F	T
Actions	a1 : Terminate the input loop	X								
	a2 : Invalid locks input				X		X	X	X	
	a3 : Invalid stocks input			X		X		X	X	
	a4 : Invalid barrels input		X			X	X		X	
	a5 : Calculate total locks, stocks and barrels		X	X	X	X	X	X		X
	a5 : Calculate Sales	X								
	a6: proceed to commission decision table	X								

Decision table for commission problem (Precondition: lock = -1)

Rules		R1	R2	R3	R4
Conditions	C1:sales=0	T	F	F	F
	C2: sales>0 and sales< =1000	-	T	F	F
	C3: sales>1001 and sales< =1800	-	-	T	F
	C4: sales >= 1801	-	-	-	T
Actions	A1:Terminate the program	X			
	A2:comm =10% * sales		X		
	A3:comm= 10%*1000+(sales-1000)*15%			X	
	A4:comm = 10%*1000+15%*800+(sales-1800)*20%				X

Precondition : Initial Value Total Locks= 0 , Total Stocks=0 and Total Barrels=0

Precondition Limit : Total locks, stocks and barrels should not exceed the limit 70,80and 90 respectively.

Commission problem decision table for input data test cases

Case Id	Description	Input data			Expected output	Actual output	status
		Locks	stocks	barrels			
1	Enter the value for locks =-1	-1			Terminate the input loop check for sales if(sales=0) exit from program else calculate commission		
2	Enter valid inputs for locks and stocks and invalid input for barrels	20	30	-5	Total of locks stocks is updated if it is with in a precondition limit and should be display value of barrels is not in range1..90		
3	Enter valid inputs for locks and barrels and invalid input for stocks	15	-2	45	Total of locks barrels is updated if it is with in a precondition limit and should be display value of stocks is not in range1..80		
4	Enter valid inputs for barrels and stocks and invalid input for locks	-4	15	16	Total of barrels stocks is updated if it is with in a precondition limit and should be display value of locks is not in range1..70		
5	Enter valid inputs for locks and invalid input for barrels and stocks	15	80	100	Total of locks is updated if it is with in a precondition limit and (i) should be display value of barrels is not in range1..90 (ii) should be display value of stocks is not in range1..80		
6	Enter valid inputs for stocks and invalid input	88	20	99	Total of stocks is updated if it is with in a precondition limit and (i)should be display		

	for barrels and locks				value of barrels is not in range1..90(ii) should be display value of locks is not in range1..70		
7	Enter valid inputs for barrels and invalid input for locks and stocks	100	200	25	Total of barrels is updated if it is with in a precondition limit and (i) should be display value of stocks is not in range1..80(ii) should be display value of locks is not in range1..70		
8	Enter invalid input for barrels and stocks and locks	-5	400	-9	(i) should be display value of stocks is not in range1..80(ii) should be display value of locks is not in range1..70(iii) should be display value of barrels is not in range1..90		
9	Enter valid inputs for locks and barrels and stocks	15	20	25	Total of locks,stocks and barrels is updated if it is with in a precondition limit and calculate sales and commission		

Commission problem - decision table for pre condition locks=-1

Case id	**Description**	**Input**	**Expected output**		**Actual output**	**Status**
		Sales	**commission**	**values**		
1	Check values for sales	0	Terminates the program and commission =0	0		
2	Sales>0 and sales< =1000	900	Comm =10% * sales	900		

3	Sales>1000 and sales< =1800	1400	Comm= 10%*1000+(sales-1000)*15%	1600		
4	Sales >= 1800	2500	Comm=10%*1000+15%*800+(sales-1800)*20%	3400		

4. Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of atriangle and determine if the three values represent an equilateral triangle, isoscelestriangle, scalene triangle, or they do not form a triangle at all. Assume that the upper limitfor the size of any side is 10. Derive test cases for your program based on boundary-value analysis, equivalence class partitioning and decision-table approach and execute thetest cases and discuss the results.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c,c1,c2,c3; char istriangle; do
{
printf("\nenter 3 integers which are sides of triangle\n");
scanf("%d%d%d",&a,&b,&c); printf("\na=%d\tb=%d\tc=%d",a,b,c);
c1 = a>=1 && a<=10; c2= b>=1 && b<=10; c3= c>=1 && c<=10; if(!c1)
printf("\n the value of a=%d is not the range of permitted value",a); if(!c2)
printf("\n the value of b=%d is not the range of permitted value",b); if(!c3)
printf("\n the value of c=%d is not the range of permitted value",c);
}
while(!(c1 && c2&& c3));
{
// to check is it a triangle or not if( a<b+c && b<a+c && c<a+b ) istriangle='y';
```

```
else
istriangle ='n';
if(istriangle=='y')
if((a==b)&&(b==c))
 printf("equilateral triangle\n");
else if((a!=b) && (a!=c) && (b!=c))
 printf("scalene triangle\n");
else
printf("isosceles triangle\n");
else
printf("Not a triangle\n");
 getch();
}
}
```

BOUNDARY VALUE ANALYSIS

Test Case Name: Boundary Value Analysis for triangle problem

Test Data: Enter the 3 Integer Value (a , b and c)

Pre-condition: $1 \leq a \leq 10$, $1 \leq b \leq 10$ and $1 \leq c \leq 10$ and $a < b + c$, $b < a + c$ and $c < a + b$ **Conditions evaluated for:** whether given value for a Equilateral, Isosceles, Scalene triangle or can't form a triangle.

Case id(BVA)	Description	Input data			Expected output	Actual output	status
		a	b	c			
1.	Enter min value for a b and c	1	1	1	Equilateral triangle		Pass/Fail
2.	Enter min value for two items and min+1 for any one item	1	1	2	Triangle Formationnot possible		
3.	Enter valid values for a b and c	5	5	5	Equilateral triangle		
4.	Enter valid values for two items and max values for one item	5	5	10	Triangle Formationnot possible		
5.	Enter valid values for two items and max-1 values for one item	10	9	10	Isosceles triangle		
6.	Enter max values for a b and c	10	10	10	Equilateral triangle		
7.	Enter different values for a b and c	1	2	3	Triangle Formation not possible		
8.	Enter different values for a b and c	2	3	1	Triangle Formation not possible		

Note : Also Perform robustness testing , worst case testing and special value testing

EQUIVALENCE CLASS TESTING

Test Case Name : Equivalence class Analysis for triangle

Test Data : Enter the 3 Integer Value(a , b And c)

Pre-condition : $1 \leq a \leq 10$, $1 \leq b \leq 10$ and $1 \leq c \leq 10$ and $a < b + c$, $b < a + c$ and $c < a + b$

Brief Description : Check whether given value for Equilateral, Isosceles ,Scalene triangle or can't form a triangle

Weak equivalence class testing

Case id	Description	Input data			Expected output	Actual output	status
		a	b	c			
1	Enter min value for a b and c	5	5	5	Equilateral triangle		
2	Enter min value for a b and c	2	2	3	Isosceles triangle		
3	Enter min value for a b and c	3	4	5	Scalene triangle		
4	Enter min value for a b and c	4	1	2	Cannot form a triangle		

Weak robust equivalence class testing

Case id	Description	Input data			Expected output	Actual output	status
		A	b	C			
1	Enter one invalid input and two valid inputs	-1	5	5	Value of a is not in range of permitted values		
2	Enter one invalid input and two valid inputs	5	-1	5	Value of b is not in range of permitted values		
3	Enter one invalid input and two valid inputs	5	5	-1	Value of c is not in range of permitted values		
4	Enter one in valid input and two valid inputs	11	5	5	Value of a is not in range of permitted values		
5	Enter one in valid input and two valid inputs	5	11	5	Value of b is not in range of permitted values		
6	Enter one in valid input and two valid inputs	5	5	11	Value of c is not in range of permitted values		

Strong robust equivalence class testing

Case id	Description	Input data			Expected output	Actual output	status
		a	b	c			
1	Enter one invalid input and two valid inputs	-1	5	5	Value of a is not in range of permitted values		
2	Enter one invalid input and two valid inputs	5	-1	5	Value of b is not in range of permitted values		
3	Enter one invalid input and two valid inputs	5	5	-1	Value of c is not in range of permitted values		
4	Enter two in valid input and one valid inputs	-1	-1	5	Value of a and b I are not in range of permitted values		
5	Enter two in valid input and one valid inputs	-1	5	-1	Value of b and a are not in range of permitted values		
6	Enter two in valid input and one valid inputs	5	-1	-1	Value of c and b are not in range of permitted values		
7	Enter all three invalid inputs	-1	-1	-1	Input values are not in range of permitted values		

DESCISION TABLE TESTING

Test Case Name: Decision table for triangle problem

Test Data: Enter the 3 Integer Value (a, b And c)

Pre-condition: a < b + c , b < a + c and c < a + b

Conditions evaluated for: whether given value for a equilateral, isosceles , Scalene triangle orcan't form a triangle

Rules	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
Conditions:											
C1: a<(b+c)?	F	T	T	T	T	T	T	T	T	T	T
C2: b<(a+c)?	-	F	T	T	T	T	T	T	T	T	T
C3: c<(b+a)?	-	-	F	T	T	T	T	T	T	T	T
C4: a=b?	-	-	-	F	T	T	T	F	F	F	T
C5: b=c?	-	-	-	T	F	T	F	T	F	F	T
C6: c=a?	-	-	-	T	T	F	F	F	T	F	T
Actions:											
A1: not a triangle	X	X	X								

A2: equilateral											X
A3: isosceles							X	X	X		
A4: scalene										X	
A5: impossible				X	X	X					

Triangle Problem -Decision Table Test cases for inputdata

Case id	Description	Input data			Expected output	Actual output	Status
		A	b	c			
1	Enter the value of a,b and c such that a is notless than sum of two sides	20	5	5	Cannot form a triangle		
2	Enter the value of a,b and c such that b is notless than sum of two sides and a is less than sum of two sides	3	15	11	Cannot form a triangle		
3	Enter the value of a , b and c such that c is not less than sum of two sides and a and b is lessthan sum of other	4	5	20	Cannot form a triangle		

5. Design, develop, code and run the program in any suitable language to solve thecommission problem. Analyze it from the perspective of dataflow testing, derive different test cases, execute these test cases and discuss the testresults.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int locks, stocks, barrels, tlocks, tstocks, tbarrels;
float lprice,sprice,bprice,lsales,ssales,bsales,sales,comm; lprice=45.0;
sprice=30.0; bprice=25.0; tlocks=0; tstocks=0; tbarrels=0;
printf("\n enter the number of locks and to exit the loop enter -1 for locks\n");
 scanf("%d", &locks);
while(locks!=-1)
{
printf("enter the number of stocks and barrels\n");
scanf("%d%d",&stocks,&barrels);
 tlocks=tlocks+locks;
tstocks=tstocks+stocks;
 tbarrels=tbarrels+barrels;
printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
 scanf("%d",&locks);
}
printf("\ntotal locks = %d\n",tlocks);
printf("total stocks =%d\n",tstocks);
printf("total barrels =%d\n",tbarrels);
lsales = lprice*tlocks;
ssales=sprice*tstocks;
bsales=bprice*tbarrels;
```

```
sales=lsales+ssales+bsales;
printf("\nthe total sales=%f\n",sales);
 if(sales > 1800.0)
{
comm=0.10*1000.0;
comm=comm+0.15*80;
 comm=comm+0.20*(sales-1800.0);
}
else if(sales > 1000)
{
comm =0.10*1000;
 comm=comm+0.15*(sales-1000);
}
else comm=0.10*sales;
printf("the commission is=%f\n",comm);
return 0;
}
```

DATA FLOW TESTING

Test Case Name: Data Flow Testing for Commission Program

Precondition: Enter -1 for locks to exit from input loop Brief **Description:**

Enter the locks, stocks and barrels > 0

Define /Use nodes for variables in the commission problem

Variable name	Defined at node	Used at node
Lprice	5	26
Sprice	6	27
Bprice	7	28
Tlocks	8,16	16
Tstocks	9,17	17
Tbarrels	10,18	18
Locks	12	21

Selected Define/use paths for commission problem

Test case id	Description	Variable path (beginning and end nodes)	Du paths	Definition clear?
1	Check for lock price variable DEF(lprice,5) and USE(lprice,26)	(5,26)	<5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26>	Yes
2	Check for lock variable DEF(locks ,12) and USE(locks,)	(12,22)	<12-13-14-15-16-17-18-19-20-**21**-22>	No

Note: Find path for all variables used and identify Definition Clear and non-DefinitionClear paths.

6.Design, develop, code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different testcases, execute these test cases and discuss the test results

```
#include <stdio.h>
#include <stdlib.h>
int binsrc(int x[],int low,int high,int key)
{
int mid; while(low<=high)
{
mid=(low+high)/2;
 if(x[mid]==key)
return mid;
if(x[mid]<key)
```

```
low=mid+1;
else
high=mid-1;
return -1;
}
}
int main()
{
int a[20],key,i,n,succ;
printf("Enter the n value");
scanf("%d",&n);
if(n>0)
{
printf("enter the elements in ascending order\n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("enter the key element to be searched\n");
scanf("%d",&key);
succ=binsrc(a,0,n-1,key);
if(succ>=0)
printf("Element found in position = %d\n",succ+1);
 else
printf("Element not found \n");
printf("Number of element should be greater than zero\n");
return 0;
```

```
}
}
```

PATH TESTING

Binary Search function with line number

```
int binsrc(int x[],int low, int high, int key)
{
int mid; while(low<=high)
{
mid=(low+high)/2; if(x[mid]==key)
return mid; if(x[mid]<key)
low=mid+1; else high=mid-1;
}
return -1;
}
```

PATH TESTING

Independent paths :

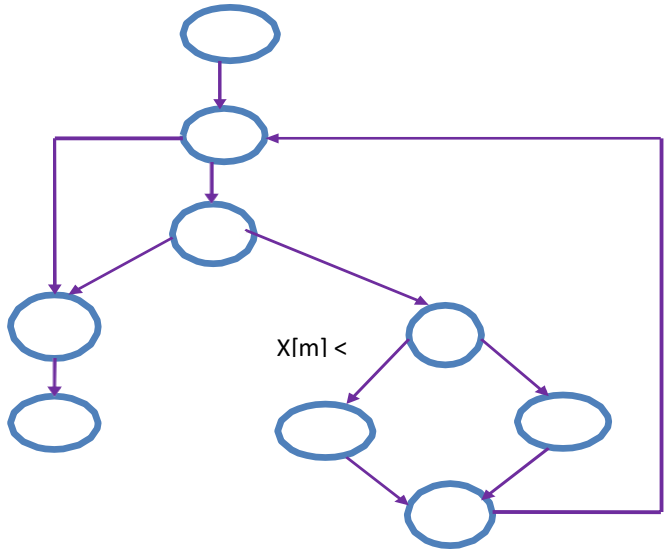
#Edges =11 ,#nodes= 9, #p =1 V(G)=E-N+2P=11-9+2=4

P1: 1-2-3-8-9

P2: 1-2-3-4-5-7-2

P3: 1-2-3-4-6-7-2

P4: 1-2-8-9



PRE CONDITIONS:

Array element is in ascending order

True /False Key element is in the array,

True /False Array has ODD number of elements True /False

Paths	Inputs		Expected output	Remarks
	X[ ]	Key		
P1: 1-2-3-8-9	{10,20,30,40,50}	30	Success	Key ∈X[ ] & Key==X[mid]
P2: 1-2-3-4-5-7-2	{10,20,30,40,50}	20	Repeat and success	Key < X[mid] search 1st half
P3: 1-2-3-4-6-7-2	{10,20,30,40,50}	40	Repeat and success	Key >X[mid] search 2nd half
P4: 1-2-8-9	{10,20,30,40,50}	60 or 50	Repeat and Failure	Key∉X[mid]
P5: 1-2-8-9	Empty	Any key	Failure	Empty list

Mini Project Ideas

- Combine Manual and Automation Testing
- Software Testing in Internet of Things
- Automated Network Security Testing Tool
- E-commerce website Testing
- Using Faulty Injection Testing Application Vulnerabilities.
- Testing the Angular Software
- AI and ML to enhance the automated Software testing
- DevOps and Agile in Software testing
- Data leakage detection system
- Credit card fraud detection
- AI shopping system
- Camera motion sensor system
- Bug tracker
- e-Learning platform
- Smart health prediction system
- Software piracy protection system
- Face detector
- Voice Recognition
- Chatbots

Not limited to these.

VIVA QUESTIONS

1. Define the following
i. error
ii. Fault
iii. Failure
iv. Incident
v. Test
vi. Test cases
2. What are pre conditions?
3. What is post Conditions?
4. What is Functional (Black box) testing?
5. What is structural testing (Clear box, White box testing)
6. Different types of faults?
7. List different types of functional testing?
8. List different types of structural testing?
9. What is boundary value analysis?
10. Number of test cases for boundary value analysis for n numbers?
11. What type of variables we are using boundary value analysis?
12. What are the 5 boundary value analysis values of a variable?
13. What is Robust testing?
14. What is worst case casting?
15. Number of test cases for worst case testing?
16. What is robust worst case testing?
17. Number of test cases for robust worst case testing?
18. What is special value testing (skirt testing)?
19. What is the limitation of boundary value analysis?
20. What is equivalence class?
21. What is equivalence class testing?

22. What is weak normal equivalence class testing?
23. What is strong normal equivalence class testing?
24. What is weak robust equivalence class testing?
25. What is strong robust equivalence class testing?
26. When equivalence class is appropriate
27. What is decision table based testing?
28. What are the different positions of decision table?
29. What is a rule?
30. What are the different types of decision tables?
31. What are limited entry decision tables?
32. What is extended entry decision tables?
33. When decision table based testing is appropriate?
34. What is a program graph?
35. What is DD (decision to decision) path?
36. What is test coverage metrics?
37. What is basis path testing?
38. What is data flow testing?
39. What is Def (v.n)?
40. What is Use (v,n)?
41. What is definition use path?
42. What is a slice?
43. Different types of use nodes?
44. Different types of definition nodes?
45. What is a system thread?
46. What is automic system function?
47. What is unit testing?
48. What is system testing?
49. What is Integration testing?
50. What is interaction testing?