Program 1 and 4 (Boundary Value and Equivalence Class Analysis program)

/* 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 triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Assume that the upper limit for the size of any side is 10. Derive test cases for your program based on boundary value analysis, execute the test cases and discuss the results */

```
#include<stdio.h>
int main()
{
int a,b,c,c1,c2,c3;
char istriangle;
do
{
printf("\n enter 3 integers which are sides of triangle\n");
scanf("%d%d%d", &a, &b, &c);
printf("\n a=%d\t b=%d\t c=%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");
return 0;
}
```

Test Case Name :Boundary Value Analysis for triangle problem

Experiment Number : 1

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 a Equilateral, Isosceles , Scalene triangle or can't form a triangle

Triangle Problem -Boundary value Test cases for input data

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	b	c				
1	Keep a and b at nominal value and vary c	5	5	1	Should display the message Isosceles triangle			
2	Keep a and b at nominal value and vary c	5	5	2	Should display the message Isosceles triangle			
3	Keep a and b at nominal value and vary c	5	5	5	Should display the message Equilateral triangle			
4	Keep a and b at nominal value and vary c	5	5	9	Should display the message Isosceles triangle			
5	Keep a and b at nominal value and vary c	5	5	10	Should display the message Not a triangle			
6	Keep a and cat nominal value and vary b	5	1	5	Should display the message Isosceles triangle			
7	Keep a and c at nominal value and vary b	5	2	5	Should display the message Isosceles triangle			
8	Keep a and c at nominal value and vary b	5	5	5	Should display the message Equilateral triangle			

9	Keep a and c at nominal value and vary b	5	9	5	Should display the message Isosceles triangle			
10	Keep a and c at nominal value and vary b	5	10	5	Should display the message Not a triangle			
11	Keep b and cat nominal value and vary a	1	5	5	Should display the message Isosceles triangle			
12	Keep b and c at nominal value and vary a	2	5	5	Should display the message Isosceles triangle			
13	Keep b and c at nominal value and vary a	5	5	5	Should display the message Equilateral triangle			
14	Keep b and c at nominal value and vary a	9	5	5	Should display the message Isosceles triangle			
15	Keep b and c at nominal value and vary a	10	5	5	Should display the message Not a triangle			

Triangle Problem Worst-Case-Test Cases (one corner of a triangle)

Case	Description	a	b	c	Expected Output	Actual Output	Status	Comments
1	Enter the **min value** for a , b and c	**1**	1	**1**	Should display the message as Equilateral triangle			
2	Enter the **min value** for 2 items and **min +1** for any one item	**1**	1	**2**	Should display the message as Not a Triangle			
3	Enter the **min value** for 2 items and **Average value** for any one item	**1**	1	**5**	Should display the message as Not a Triangle			
4	Enter the **min value** for 2 items and **Max -1** for any one item	**1**	1	**9**	Should display the message as Not a Triangle			
5	Enter the **min value** for 2 items and **Max** for any one item	**1**	1	**10**	Should display the message as Not a Triangle			
6	Enter the **min value** for 2 items and **min +1** for any one item	**1**	2	**1**	Should display the message as Not a Triangle			
7	Enter the **min+1 value** for 2 items and **min** for any one item	**1**	2	**2**	Should display the message as Isosceles			
8	Enter the **min value** for **1 items, min+1** and **Average value** for any one item	**1**	2	**5**	Should display the message as Not a Triangle			
9	Enter the **min value** for **1 items, min+1** and **max-1** for any one item	**1**	2	**9**	Should display the message as Not a Triangle			
10	Enter the **min value** for **1 items, min+1** and **max** for any one item	**1**	2	**10**	Should display the message as Not a Triangle			

11	Enter the **min value** for 2 **items, average value** for any one item	**1**	5	**1**	Should display the message as Not a Triangle			
12	Enter the **min value** for **1 items, min+1** and **average** for any one item	**1**	5	**2**	Should display the message as Not a Triangle			
13	Enter the **min value** for **1 items,** and **average** for any 2 items	**1**	5	**5**	Should display the message as Isosceles			
14	Enter the **min value** for **1 items, max-1** and **average** for any one item	**1**	5	**9**	Should display the message as Not a Triangle			
15	Enter the min value for **1 items, max** and **average** for any one item	**1**	5	**10**	Should display the message as Not a Triangle			
16	Enter the **min value** for 2 items and **max -1** for any one item1	**1**	9	**1**	Should display the message as Not a Triangle			
17	Enter the **min value** for **1 items, min+1** and **max-1** for any one item	**1**	9	**2**	Should display the message as Not a Triangle			
18	Enter the **min value** for **1 items, max-1**and **Average value** for any one item	**1**	9	**5**	Should display the message as Not a Triangle			
19	Enter the **min value** for**1 items, max-1**for 2 items	**1**	9	**9**	Should display the message as Isosceles			
20	Enter the **min value** for **1 items, max-1**and **Max value** for any one item	**1**	9	**10**	Should display the message as Not a Triangle			
21	Enter the **min value** for **2 items** and **max** for any one item	**1**	10	**1**	Should display the message as Not a Triangle			

22	Enter the **min value** for **1 items, min+1** and **max** for any one item	**1**	10	**2**	Should display the message as Not a Triangle			
23	Enter the **min value** for **1 items, max** and **Average value** for any one item	**1**	10	**5**	Should display the message as Not a Triangle			
24	Enter the **min value** for **1 items, max-1** , and **max** for 1 items	**1**	10	**9**	Should display the message as Not a Triangle			
25	Enter the **min value** for **1 items,** and **Max value** for 2 items	**1**	10	**10**	Should display the message as Isosceles			

Special Value Test Cases

Case	Description	a	b	c	Expected Output	Actual Output	Status	Comments
1	Enter the **values** for a , b and c	**5**	8	**6**	Should display the message as Scalene triangle			
2	Enter the **out of boundary value** for a and b and **normal** value for c	**11**	0	**5**	Should display the message as value of a and b not in the permitted range			
3	Enter the **negative value** for a, b and c	**-1**	-4	**-6**	Should display the message as value of a, b and c not in the permitted range			
4	Enter the **values** for a , b and c	**5**	**1**	**10**	Should display the message as Not a Triangle			

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	✓	✓	✓	✓						✓			✓	✓	✓

Test Case Name :Equivalence Class Analysis for triangle problem

Experiment Number : 4

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

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

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

Triangle Problem - Equivalence Class Test cases

Weak and Strong Normal Equivalence class Testing								
Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	b	C				
WN1 /SN1	Enter the nom value for a , b and c	5	5	5	Should display the message Equilateral triangle			
WN2 /SN2	Enter the valid value for a , b and c	2	2	3	Should display the message Isosceles triangle			
WN3 /SN3	Enter the valid value for a , b and c	3	4	5	Should display the message Scalene triangle			
WN4 /SN4	Enter the valid value for a , b and c	4	1	2	Message should be displayed can't form a triangle			

Weak Robust Equivalence Class Testing								
WR1	Enter one invalid input and two valid value for a , b and c	-1	5	5	Should display value of a is not in the range of permitted values			
WR2	Enter one invalid input and two valid value for a , b and c	5	-1	5	Should display value of b is not in the range of permitted values			
WR3	Enter one invalid input and two valid value for a , b and c	5	5	-1	Should display value of c is not in the range of permitted values			
WR4	Enter one invalid input and two valid value for a , b and c	11	5	5	Should display value of a is not in the range of permitted values			
WR5	Enter one invalid input and two valid value for a , b and c	5	11	5	Should display value of b is not in the range of permitted values			
WR6	Enter one invalid input and two valid value for a , b and c	5	5	11	Should display value of c is not in the range of permitted values			

Strong Robust Equivalence class Testing								
SR1	Enter one invalid input and two valid value for a , b and c	-1	5	5	Should display value of a is not in the range of permitted values			
SR2	Enter one invalid input and two valid value for a , b and c	5	-1	5	Should display value of b is not in the range of permitted values			
SR3	Enter one invalid input and two valid value for a , b and c	5	5	-1	Should display value of c is not in the range of permitted values			
SR4	Enter two invalid input and one valid value for a , b and c	-1	-1	5	Should display value of a and b are not in the range of permitted values			
SR5	Enter two invalid input and one valid value for a , b and c	5	-1	-1	Should display value of b and c are not in the range of permitted values			
SR6	Enter two invalid input and one valid value for a , b and c	-1	5	-1	Should display value of a and c are not in the range of permitted values			
SR7	Enter all invalid inputs	-1	-1	-1	Should display value of a, b and c are not in the range of permitted values			

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO2	✓	✓	✓	✓						✓			✓	✓	✓

Program 2, 5 and 8 (Boundary, Equivalence and Decision Table for Commission Problem)

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

/* Assumption price for lock=45.0, stock=30.0 and barrels=25.0, production limit that could be sold in a month is 70 locks, 80 stocks and 90 barrels. Commission on sales = 10 % on sales <= 1000 and 15 % on 1001 to 1800 and 20 % on above 1800*/

```
#include<stdio.h>
int main()
{
        Int locks, stocks, barrels, tlocks, tstocks, tbarrels;
        float lprice, sprice, bprice, sales, comm;
        int c1,c2,c3,temp;
        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)
        {
              c1= (locks<=0 || locks>70);
              printf("enter the number of stocks and barrels\n");
              scanf("%d%d", &stocks, &barrels);
              c2=(stocks<=0 || stocks>80);
              c3=(barrels<=0 || barrels>90);
              if(c1)
                    printf("value of locks not in the range 1..70  ");
              else
              {
                    temp=tlocks+locks;
                    if(temp>70)
                          printf("new total locks =%d not in the range 1..70 ", temp);
                    else
                           tlocks=temp;
              }
               printf("total locks = %d\n", tlocks);

               if(c2)
                    printf("value of stocks not in the range 1..80  ");
               else
               {
                    temp=tstocks+stocks;
```

```
                if(temp>80)
                    printf("new total stocks =%d  not in the range 1..80 ", temp);

                 else
                        tstocks=temp;
            }
            printf("total stocks=%d\n", tstocks);

            if(c3)
                printf("value of barrels not in the range 1..90  ");
            else
              {
                temp=tbarrels+barrels;
                if(temp>90)
                        printf("new total barrels =%d not in the range 1..90 ", temp);
                else
                         tbarrels=temp;
               }
            printf("total barrels=%d", tbarrels);
            printf("\n enter the number of locks and to exit the loop enter -1 for locks \n");
            scanf("%d", &locks);
    }
    printf("\n total locks = %d\n total stocks =%d\n total barrels =%d\n", tlocks, tstocks, tbarrels);
    sales = lprice*tlocks + sprice*tstocks + bprice*tbarrels;
    printf("\n the total sales=%f\n", sales);
    if(sales > 0)
    {
          if(sales > 1800.0)
          {
              comm=0.10*1000.0;
              comm=comm+0.15*800;
              comm=comm+0.20*(sales-1800.0);
          }
         else if(sales > 1000)
         {
               comm =0.10*1000;
               comm =comm+0.15*(sales-1000.0);
         }
          else
                comm=0.10*sales;
                 printf("the commission is=%f\n", comm);
     }
    else
           printf("there is no sales\n");
           return 0;
}
```

Test Case Name : Boundary Value for Commission Problem
Experiment Number : 2
Test data : price for lock = 45.0 , stock = 30.0 and barrel = 25.0
sales = total locks * lock price + total stocks * stock price + total barrels * barrel price
commission : 10% up to sales Rs 1000 , 15 % for the next Rs 800 and 20 % on any 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.

Commission Problem Boundary Value Analysis Test Cases

Case Id	Description	Input Data			Expected Output		Actual output		Status	Comment
		Total Locks	Total Stocks	Total Barrels	Sales	Comm-ission	Sales	Comm-ission		
1	Set locks and stocks as nominal value and vary barrels value.	35	40	1	2800					
2	Set locks and stocks as nominal value and vary barrels value.	35	40	2	2825					
3	Set locks and stocks as nominal value and vary barrels value.	35	40	45	3900					
4	Set locks and stocks as nominal value and vary barrels value.	35	40	89	5000					
5	Set locks and stocks as nominal value and vary barrels value.	35	40	90	5025					
6	Set locks and barrels as nominal value and vary stocks value	35	1	45	2730					
7	Set locks and barrels as nominal value and vary stocks value	35	2	45	2760					
8	Set locks and barrels as nominal value and vary stocks value	35	40	45	3900					
9	Set locks and barrels as nominal value and vary stocks value	35	79	45	5070					
10	Set locks and barrels as nominal value and vary stocks value	35	80	45	5100					

11	Set stocks and barrels as nominal value and vary locks value	1	40	45	2370					
12	Set stocks and barrels as nominal value and vary locks value	2	40	45	2415					
13	Set stocks and barrels as nominal value and vary locks value	35	40	45	3900					
14	Set stocks and barrels as nominal value and vary locks value	69	40	45	5430					
15	Set stocks and barrels as nominal value and vary locks value	70	40	45	5475					

Commission Problem Output Boundary Value Analysis Test Cases

Case Id	Description	Input Data			Expected Output		Actual output		Status	
		Total Locks	Total Stocks	Total Barrels	Sales	Comm-ission	Sales	Comm-ission		Comment
1	Enter the min value for locks, stocks and barrels	1	1	1	100	10				output minimum
2	Enter the min value for 2 items and min +1 for any one item	1	1	2	125	12.5				output minimum +
3		1	2	1	130	13				output minimum +
4		2	1	1	145	14.5				output minimum +
5	Enter the value sales approximately mid value between 100 to 1000	5	5	5	500	50				Midpoint
6	Enter the values to calculate the commission for sales nearly less than 1000	10	10	9	975	97.5				Border point -
7		10	9	10	970	97				Border point -
8		9	10	10	955	95.5				Border point -
9	Enter the values sales exactly equal to 1000	10	10	10	1000	100				Border point

10	Enter the values to calculate the commission for sales nearly greater than 1000	10	10	11	1025	103.75				Border point +
11		10	11	10	1030	104.5				Border point +
12		11	10	10	1045	106.75				Border point +
13	Enter the value sales approximately mid value between 1000 to 1800	14	14	14	1400	160				Midpoint
14	Enter the values to calculate the commission for sales nearly less than 1800	18	18	17	1775	216.25				Border point -
15		18	17	18	1770	215.5				Border point -
16		17	18	18	1755	213.25				Border point -
17	Enter the values sales exactly equal to 1800	18	18	18	1800	220				Border point
18	Enter the values to calculate the commission for sales nearly greater than 1800	18	18	19	1825	225				Border point +
19		18	19	18	1830	226				Border point +
20		19	18	18	1845	229				Border point +
21	Enter the value sales approximately mid value between 1800 to 7800	48	48	48	4800	820				Midpoint
22	Enter the max value for 2 items and max - 1 for any one item	70	80	89	7775	1415				Output maximum -
23		70	79	90	7770	1414				Output maximum -
24		69	80	90	7755	1411				Output maximum -
25	Enter the max value for locks, stocks and barrels	70	80	90	7800	1420				Output maximum

Output Special Value Test Cases

Case Id	Description	Input Data			Expected Output		Actual output			
		Total Locks	Total Stocks	Total Barrels	Sales	Comm-ission	Sales	Comm -ission	Status	Comment
1	Enter the random values such that to calculate commission for sales nearly less than 1000	11	10	8	995	99.5				Border point -
2	Enter the random values such that to calculate commission for sales nearly greater than 1000	10	11	9	1005	100.75				Border point +
3	Enter the random values such that to calculate commission for sales nearly less than 1800	18	17	19	1795	219.25				Border point -
4	Enter the random values such that to calculate commission for sales nearly greater than 1800	18	19	17	1805	221				Border point +

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	✓	✓	✓	✓						✓			✓	✓	✓

Test Case Name :Equivalence Class for Commission Problem

Experiment Number : 5

Test data : price for lock = 45.0 , stock = 30.0 and barrel = 25.0

sales = total locks * lock price + total stocks * stock price + total barrels * barrel price

commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any 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 has rto sell at least one complete rifle per month.

Valid Classes

L1 ={ Locks :1 <= Locks <=70}

L2 ={Locks=-1}(occurs if locks=-1 is used to control input iteration)

L3 ={stocks : 1<=stocks<=80}

L4= {barrels :1<=barrels<=90}

Invalid Classes

L3 ={locks: locks=0 **OR** locks<-1}

L4 ={locks: locks> 70}

S2 ={stocks : stocks<1}

S3 ={stocks : stocks >80}

B2 ={barrels : barrels <1}

B3 =barrels : barrels >90}

Commission Problem Output Equivalence Class Testing

Weak & Strong Normal Equivalence Class

Case Id	Description	Input Data			Expected Output		Actual output		Status	Comment
		Total Locks	Total Stocks	Total Barrels	Sales	Commission	Sales	Commission		
WN1 /SN1	Enter the value within the range for locks, stocks and barrels	35	40	45	3900	640				

Weak Robustness Equivalence Class

Case Id	Description	Input Data			Expected Output	Actual output	Status	Comment
		Locks	Stocks	Barrels				
WR1	Enter the valid values for locks, stocks and barrels	10	10	10	$100			
WR2	Enter the value locks = -1	-1	40	45	Terminates the input loop and proceed to calculate sales and commission (if Sales > 0)			
WR3	Enter the value less than -1 or equal to **zero** for locks and other valid inputs	-2	40	45	Value of Locks not in the range 1..70			
WR4	Enter the value greater than 70 for locks and other valid inputs	71	40	45	Value of Locks not in the range 1..70			
WR5	Enter the value less than or equal to 0 for stocks and other valid inputs	35	-1	45	Value of stocks not in the range 1..80			
WR6	Enter the value greater than 80 for stocks and other valid inputs	35	81	45	Value of stocks not in the range 1..80			
WR7	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	-1	Value of Barrels not in the range 1..90			
WR8	Enter the value greater than 90 for barrels and other valid inputs	35	40	91	Value of Barrels not in the range 1..90			

Strong Robustness Equivalence Class

Case Id	Description	Input Data			Expected Output	Actual output	Status	Comment
		Locks	Stocks	Barrels				
SR1	Enter the value less than -1 for locks and other valid inputs	-2	40	45	Value of Locks not in the range 1..70			
SR2	Enter the value less than or equal than 0 for stocks and other valid inputs	35	-1	45	Value of stocks not in the range 1..80			
SR3	Enter the value less than or equal 0 for barrels and other valid inputs	35	40	-1	Value of Barrels not in the range 1..90			

SR4	Enter the locks and stocks less than or equal to 0 and other valid inputs	-2	-1	45	Value of Locks not in the range 1..70			
					Value of stocks not in the range 1..80			
SR5	Enter the locks and barrel less than or equal to 0 and other valid inputs	-2	40	-1	Value of Locks not in the range 1..70			
					Value of Barrels not in the range 1..90			
SR6	Enter the stocks and barrel less than or equal to 0 and other valid inputs	35	-1	-1	Value of stocks not in the range 1..80			
					Value of Barrels not in the range 1..90			
SR7	Enter the stocks and barrel less than or equal to 0 and other valid inputs	-2	-1	-1	Value of Locks not in the range 1..70			
					Value of stocks not in the range 1..80			
					Value of Barrels not in the range 1..90			

Equivalence Class Testing for Output range

We could define equivalence classes for output commission range as follows,
S1 = {<locks, stocks, barrels >: **sales ≤ 1000**}
S2 = {<locks, stocks, barrels >: **1000< sales ≤ 1800**}
S3 = {<locks, stocks, barrels >: **sales > 1800**}

Case Id	Description	Input Data			Expected Output		Actual output			
		Total Locks	Total Stocks	Total Barrels	Sales	Commission	Sales	Commission	Status	Comment
OR1	Enter the value for lock, stocks and barrels where 0 < Sales <= 1000	5	5	5	500	50				
OR2	Enter the value for lock, stocks and barrels where 1000 < Sales <=1800	15	15	15	1500	175				
OR3	Enter the value for lock, stocks and barrels where Sales > 1800	25	25	25	2500	360				

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO2	✓	✓	✓	✓						✓			✓	✓	✓

Test Case Name :Decision Table for Commission Problem

Experiment Number : 8

Test data : price for lock = 45.0 , stock = 30.0 and barrel = 25.0
sales = total locks * lock price + total stocks * stock price + total barrels * barrel price
commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any 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.

Input data decision Table

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
	A6: Calculate Sales	X								
	A7: proceed to commission decision table	X								

Commission calculation Decision Table (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 > 1000 AND sales ≤ 1800			T	F
	C4 : sales >1800				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 70,80 and 90 respectively

Commission Problem -Decision Table Test cases for input data

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		Locks	Stocks	Barrels				
1	Enter the value of Locks= -1	-1			Terminate the input loop check for sales if(sales=0) exit from program else calculate commission			
2	Enter the valid input for locks and stocks and invalid for barrels	20	30	-5	Total of locks, stocks is updated if it is within a precondition limit and Should display value of barrels is not in the range 1..90			
3	Enter the valid input for locks and barrrels and invalid for stocks	15	-2	45	Total of locks, barrels is updated if it is within a precondition limit and Should display value of stocks is not in the range 1..80			
4	Enter the valid input for stocks and barrrels and invalid for locks	-4	15	16	Total of stocks , barrels is updated if it is within a precondition limit and Should display value of locks is not in the range 1..70			
5	Enter the valid input for locks and invalid value for stocks and barrels	15	81	100	Total of locks is updated if it is within a precondition limit and **(i)**Should display value of stock is not in the range 1..80 **(ii)**Should display value of barrels is not in the range 1..90			
6	Enter the valid input for stocks and invalid value for locks and barrels	88	20	99	Total of stocks is updated if it is within a precondition limit and **(i)**Should display value of lock is not in the range 1..70 **(ii)**Should display value of barrels is not in the range 1..90			
7	Enter the valid input for barrels and invalid value for locks and stocks	100	200	25	Total of barrels is updated if it is within a precondition limit and **(i)**Should display value of lock is not in the range 1..70 **(ii)**Should display value of stocks is not in the range 1..80			
8	Enter the invalid input for lock , stocks and barrels	-5	400	-9	**(i)**Should display value of lock is not in the range 1..70 **(ii)**Should display value of stocks is not in the range 1..80 **(iii)**Should display value of barrel in not in the range 1..90			

9	Enter the valid input for lock, stocks and barrels	15	20	25	Total of locks,stocks and barrels is updated if it is within a precondition limit and calculate the sales and proceed to commission			

Commission Problem -Decision Table Test cases for commission calculation

Precondition : Locks = -1

Case Id	Description	Input Data	Expected Output		Actual Output	Status	Comments
		Sales	Commission	Values			
1	Check the value of sales	0	Terminate the program where commission is zero	0			
2	if sales value within these range(Sales >0 AND Sales ≤ 1000)	900	Then commission = 0.10*sales	90			
3	if sales value within these range(Sales > 1000 AND Sales ≤ 1800)	1400	Then commission = 0.10*1000 + 0.15*(sales - 1000)	160			
4	if sales value within these range(Sales > 1800	2500	Then commission = 0.10*1000 + 0.15*800 + 0.20 *(sales - 1800)	340			

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO3	✓	✓	✓	✓						✓			✓	✓	✓

Program 3 and 6 (Boundary Value Analysis and Equivalence Class Testing)

/* Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective of boundary value testing and equivalence class analysis. Derive different test cases, execute these test cases and discuss the test results. */

```
#include<stdio.h>
int check(int day, int month)
{
if((month==4||month==6||month==9 ||month==11) && day==31)
 return 1;
else
 return 0;
}

int isleap(int year)
{
if((year%4==0 && year%100!=0) || year%400==0)
 return 1;
else
 return 0;
}

int main()
{
int day, month, year, tomm_day, tomm_month, tomm_year;
char flag;
do
{
flag='y';
printf("\nenter the today's date in the form of dd mm yyyy\n");
scanf("%d%d%d", &day, &month, &year);
tomm_month=month;
tomm_year= year;
 if(day<1 || day>31)
 {
  printf("value of day, not in the range 1...31\n");
   flag='n';
 }
 if(month<1 || month>12)
 {
 printf("value of month, not in the range 1....12\n");
 flag='n';
 }
 else if(check(day, month))
         {
          printf("value of day, not in the range day<=30");
```

```
          flag='n';
          }

if(year<1812 || year>2019)
{
   printf("value of year, not in the range 1812.....2019\n");
   flag='n';
}

if(month==2)
  {
     if(isleap(year) && day>29)
      {
        printf("invalid date input for leap year");
        flag='n';
      }
        else if(!(isleap(year)) && day>28)
        {
         printf("invalid date input for not a leap year");
         flag='n';
        }
  }
} while(flag=='n');

switch (month)
{
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:if(day<31)
 tomm_day=day+1;
else
   {
    tomm_day=1;
     tomm_month=month+1;
   }
break;

case 4:
case 6:
case 9:
case 11: if(day<30)
 tomm_day=day+1;
 else
    {
```

```
    tomm_day=1;
    tomm_month=month+1;
    }
break;

case 12: if(day<31)
tomm_day=day+1;
else
   {
     tomm_day=1;
     tomm_month=1;
 if(year= =2019)
  {
    printf("the next day is out of boundary value of year\n");
   }

    else
       tomm_year=year+1;
  }
break;

case 2:
if(day<28)
   tomm_day=day+1;
else if(isleap(year) && day==28)
   tomm_day=day+1;
else if(day==28 || day==29)
   {
      tomm_day=1;
      tomm_month=3;
   }
break;
}
printf("next day is : %d  %d  %d", tomm_day, tomm_month, tomm_year);
return 0;
}
```

Test Case Name : Boundary Value Analysis test cases for NextDate program

Experiment Number :3

Test data : Enter the three integer value

Pre-condition : Month 1 to 12, Day 1 to 31 and Year 1812 to 2019

Brief Description :

	Min	Min +1	Normal	Max -1	Max
Month	1	2	6	11	12
Day	1	2	15	**29/30**	**30/31**
Year	1812	1813	1915	2018	2019

NextDate Boundary Value test cases (day=1 to 30)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
1	Enter day and month as nominal value and vary year from min to max	6	15	1812	6	16	1812					
2	Enter day and month as nominal value and vary year from min to max	6	15	1813	6	16	1813					
3	Enter day and month as nominal value and vary year from min to max	6	15	1915	6	16	1915					
4	Enter day and month as nominal value and vary year from min to max	6	15	2018	6	16	2018					
5	Enter year and month as nominal value and vary day from min to max	6	15	2019	6	16	2019					
6	Enter year and month as nominal value and vary day from min to max	6	1	1915	6	2	1915					

7	Enter year and month as nominal value and vary day from min to max	6	2	1915	6	3	1915					
8	Enter year and month as nominal value and vary day from min to max	6	15	1915	6	16	1915					
9	Enter year and month as nominal value and vary day from min to max	6	29	1915	6	30	1915					
10	Enter year and month as nominal value and vary day from min to max	6	30	1915	7	1	1915					
11	Enter year and day as nominal value and vary month from min to max	1	15	1915	1	16	1915					
12	Enter year and day as nominal value and vary month from min to max	2	15	1915	2	16	1915					
13	Enter year and day as nominal value and vary month from min to max	6	15	1915	6	16	1915					
14	Enter year and day as nominal value and vary month from min to max	11	15	1915	11	16	1915					
15	Enter year and day as nominal value and vary month from min to max	12	15	1915	12	16	1915					

NextDate Boundary Value test cases (day=1 to 31)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
1	Enter day and month as nominal value and vary year from min to max	7	15	1812	7	16	1812					
2	Enter day and month as nominal value and vary year from min to max	7	15	1813	7	16	1813					
3	Enter day and month as nominal value and vary year from min to max	7	15	1915	7	16	1915					
4	Enter day and month as nominal value and vary year from min to max	7	15	2018	7	16	2018					
5	Enter year and month as nominal value and vary day from min to max	7	15	2019	7	16	2019					

6	Enter year and month as nominal value and vary day from min to max	7	1	1915	7	2	1915					
7	Enter year and month as nominal value and vary day from min to max	7	2	1915	7	3	1915					
8	Enter year and month as nominal value and vary day from min to max	7	15	1915	7	16	1915					
9	Enter year and month as nominal value and vary day from min to max	7	30	1915	7	31	1915					
10	Enter year and month as nominal value and vary day from min to max	7	31	1915	8	1	1915					
11	Enter year and day as nominal value and vary month from min to max	1	15	1915	1	16	1915					
12	Enter year and day as nominal value and vary month from min to max	2	15	1915	2	16	1915					
13	Enter year and day as nominal value and vary month from min to max	7	15	1915	7	16	1915					
14	Enter year and day as nominal value and vary month from min to max	11	15	1915	11	16	1915					
15	Enter year and day as nominal value and vary month from min to max	12	15	1915	12	16	1915					

NextDate Worst case Test Cases

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		Month	day	year	Month	day	year	Month	day	year		
1	Enter the min value month, day and year	1	1	1812	1	2	1812					
2	Enter the min+1 value for year and min for month and day	1	1	1813	1	2	1813					
3	Enter the normal value for year and min for month and day	1	1	1915	1	2	1915					
4	Enter the max -1 value for year and min for month and day	1	1	2018	1	2	2018					

5	Enter the max value for year and min for month and day	1	1	2019	1	2	2019					
6	Enter the min+1 value of day and min for month and year	1	2	1812	1	3	1812					
7	Enter the min+1 value for day and year and min for month	1	2	1813	1	3	1813					
8	Enter the min+1 value for day , normal value for year and min value for month	1	2	1915	1	3	1915					
9	Enter the min+1 value for day , max -1 value for year and min value for month	1	2	2018	1	3	2018					
10	Enter the min+1 value for day , max value for year and min value for month	1	2	2019	1	3	2019					
11	Enter the normal value of day and min for year and month	1	15	1812	1	16	1812					
12	Enter the normal value for day and min+1 for year and min for month	1	15	1813	1	16	1813					
13	Enter the normal value for day , normal value for year and min value for month	1	15	1915	1	16	1915					
14	Enter the normal value for day , max -1 value for year and min value for month	1	15	2018	1	16	2018					
15	Enter the normal value for day , max value for year and min value for month	1	15	2019	1	16	2019					
16	Enter the max - 1 value of day and min for day and year	1	30	1812	1	31	1812					
17	Enter the max -1 value for day and min for month and min+1 for year	1	30	1813	1	31	1813					
18	Enter the max - 1 value for day , normal value for year and min value for month	1	30	1915	1	31	1915					
19	Enter the max - 1 value for day , max -1 value for year and min value for month	1	30	2018	1	31	2018					

20	Enter the max -1 value for day , max value for year and min value for month	1	30	2019	1	31	2019					
21	Enter the max value of day and min for year and month	1	31	1812	2	1	1812					
22	Enter the max value for day and min for month and min + 1 for year	1	31	1813	2	1	1813					
23	Enter the max value for day , normal value for year and min value for month	1	31	1915	2	1	1915					
24	Enter the max value for day , max -1 value for year and min value for month	1	31	2018	2	1	2018					
25	Enter the max value for day , max value for year and min value for month	1	31	2019	2	1	2019					

NextDate Special value test cases

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
1	Enter the valid value for month, day and year	12	31	1811	Should display the message value of the year not in range 1812..2019							
2	Enter the valid value for month, day and year	12	31	2018	1	1	2019					
3	Enter the valid value for month, day and year	12	31	2019	Should display the message Next year is out of boundary 2019							
4	Enter the valid value for month, day and year	2	28	1900	3	01	1900					
5	Enter the valid value for month, day and year	2	28	2014	3	01	2014					

6	Enter the valid value for month, day and year	2	29	2012	3 01 2012				
7	Enter the valid value for month, day and year	2	29	2020	Should display the message value of the year not in range 1812..2019				
8	Enter the valid value for month, day and year	2	28	2012	2	29	2012		
9	Enter the valid value for month, day and year	2	28	2000	2	29	2000		
10	Enter the valid value for month, day and year	2	29	2000	3	01	2000		

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	✓	✓	✓	✓						✓			✓	✓	✓

Test Case Name : Equivalence class test cases for NextDate

Experiment Number :6

Test data : Enter the three integer value

Pre-condition : Month 1 to 12 , DAY 1 TO 31 & YEAR 1812 TO 2019

Valid Classes

M1 = { month ; 1 ≤ month ≤ 12 }

D1 = { day : 1 ≤ day ≤ 31 }

Y1 = { year : 1812 ≤ year ≤ 2019 }

Invalid Classes

M2 = {month : month < 1}

M3 = {month : month > 12}

D2 = {day : day < 1}

D3 = {day : day > 31}

Y2 = {year : year < 1812}

Y3 = {year : year > 2019}

NextDate Equivalence Class Testing

(Weak and Strong Normal Equivalence Class)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
WN1, SN1	Enter the valid value for month, day and year	6	15	1915	6	16	1915					

(Weak Robust Equivalence Class)

Case Id	Description	Input Data			Expected Output			Actual output			Status	Comment
		month	day	year	month	day	year	month	day	year		
WR1	Enter the valid value for month, day and year	6	15	1915	6	16	1915					
WR2	Enter the invalid value for month and valid value for day and year	-1	15	1915	Should display the message value of the month not in the range 1..12							
WR3	Enter the invalid value for month and valid value for day and year	13	15	1915	Should display the message value of the month not in the range 1..12							
WR4	Enter the invalid value for day and valid value for month and year	6	-1	1915	Should display the message value of the day not in the range 1..31							
WR5	Enter the invalid value for day and valid value for month and year	6	32	1915	Should display the message value of the day not in the range 1..31							
WR6	Enter the invalid value for year and valid value for month and day	6	15	1811	Should display the message value of the year not in the range 1812..2017							
WR7	Enter the invalid value for year and valid value for month and day	6	15	2020	Should display the message value of the year not in the range 1812..2019							

(Strong Robust Equivalence Class)

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comment
		month	day	year				
SR1	Enter the invalid value for month and valid value for day and year	-1	15	1915	Should display the message value of the month not in the range 1..12			
SR2	Enter the invalid value for day and valid value for month and year	6	-1	1915	Should display the message value of the day not in the range 1..31			
SR3	Enter the invalid value for year and valid value for month and day	6	15	1811	Should display the message value of the year not in the range 1812..2019			
SR4	Enter the invalid value for month and day and valid value for year	-1	-1	1915	(i)Should display the message value of the month not in range 1..12			
					(ii) Should display the message value of the day not in range 1..31			
SR5	Enter the invalid value for day and year and valid value for month	6	-1	1811	**(i)** Should display the message value of the day not in range 1..31			
					(ii) Should display the message value of the year not in range 1812..2019			
SR6	Enter the invalid value for year and month and valid value for day	-1	15	1811	(i)Should display the message value of the month not in range 1..12			
					(ii) Should display the message value of the year not in range 1812..2019			
SR7	Enter the invalid value for month, day and year	-1	-1	1811	(i)Should display the message value of the month not in range 1..12			
					(ii) Should display the message value of the day not in range 1..31			
					(iii) Should display the message value of the year not in range 1812..2019			

Some addition Equivalence Class Testcases

Case Id	Description	Input Data			Expected Output			Actual Output			Status	Comment
		day	month	year	day	month	year	day	month	year		
1	Enter the invalid value for year valid value for day and month	31	12	1811	Should display the message value of the year not in range 1812..2019							
2	Enter the valid value for month, day and year	31	12	2016	1	1	2017					
3	Enter the valid value for month, day and year	28	2	2000	29	2	2000					
4	Enter the valid value for month, day and year	28	2	1996	29	2	1996					
5	Enter the valid value for month, day and year	29	2	2000	1	3	2000					
6	Enter the valid value for month, day and year	29	2	1996	1	3	1996					
7	Enter the valid value for month, day and year	28	2	2002	1	3	2002					
8	Enter the valid value for month, day and year	29	2	2002	Invalid I/P Date							
9	Enter the invalid value for year, valid value for day and month	31	12	2020								

Program 7: Decision Table Approach for Solving Triangle Problem

/* 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 triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results */

```
#include<stdio.h>
int 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");
return 0;
}
```

Test Case Name :Decision table for triangle problem

Experiment Number : 7

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

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

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

Input data decision Table

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 < a + b	-	-	F	T	T	T	T	T	T	T	T
	C4 : a = b	-	-	-	T	T	T	T	F	F	F	F
	C5 : a = c	-	-	-	T	T	F	F	T	T	F	F
	C6 : b = c	-	-	-	T	F	T	F	T	F	T	F
Actions	a1 : Not a triangle	X	X	X								
	a2 : Scalene triangle											X
	a3 : Isosceles triangle							X		X	X	
	a4 : Equilateral triangle				X							
	a5 : Impossible					X	X		X			

Triangle Problem -Decision Table Test cases for input data

Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
		a	b	c				
1	Enter the value of a, b and c Such that a is not less than sum of two sides	20	5	5	Message should be displayed can't form a triangle			
2	Enter the value of a, b and c Such that b is not less than sum of two sides and a is less than sum of other two sides	3	15	11	Message should be displayed can't 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 less than sum of other two sides	4	5	20	Message should be displayed can't form a triangle			
4	Enter the value a, b and c satisfying precondition and a=b, b=c and c=a	5	5	5	Should display the message Equilateral triangle			
5	Enter the value a ,b and c satisfying precondition and a=b and b ≠ c	10	10	9	Should display the message Isosceles triangle			
6	Enter the value a, b and c satisfying precondition and a ≠b , b ≠ c and c ≠ a	5	6	7	Should display the message Scalene triangle			

```
1  //Program 9:(Dataflow Testing  for commission calculation)
2  #include<stdio.h>
3  int main()
4  {
5     int locks, stocks, barrels, tlocks, tstocks, tbarrels;
6                 float lprice, sprice, bprice, lsales, ssales, bsales, sales, comm;
7              lprice =45.0;
8                 sprice=30.0;
9              bprice=25.0;
10             tlocks=0;
11             tstocks=0;
12             tbarrels=0;
               printf("\nenter the number of locks and to exit the loop enter -1 for locks\n");
13             scanf("%d",&locks);
14             while(locks!= -1){
               printf("enter the number of stocks and barrels\n");
15             scanf("%d%d",&stocks, &barrels);
16                    tlocks = tlocks + locks;
17             tstocks = tstocks + stocks;
18                    tbarrels = btarrels + barrels;
               printf("\n enter the number of locks and to exit the loop enter -1 for locks\n");
19             scanf("%d", &locks);
20    }
21       printf("\n total locks = %d\", tlocks);
22       printf("total stocks =%d\n", tstocks);
23       printf("total barrels =%d\n", tbarrels);

24       lsales = lprice*tlocks;
25       ssales = sprice*tstocks;
26       bsales = bprice*tbarrels;
27       sales = lsales + ssales + bsales;
28       printf("\n the total sales=%f\n", sales);
29       if(sales > 1800.0)
```

```
30      {
31              comm=0.10*1000.0;
32              comm=comm+0.15*800;
33              comm=comm+0.20*(sales-1800.0);
           }
34      else if(sales > 1000)
35      {
36              comm =0.10*1000;
37              comm=comm+0.15*(sales-1000);
         }
38      else
39              { comm=0.10*sales;
40  }
41      printf" \n value of commission is\n");
42  printf("the commission is=%f\n", comm);
43      return 0;  }
```

Define /Use nodes for variables in the commission problem

Variable name	Defined at node	Used at Node
lprice	7	24
sprice	8	25
bprice	9	26
tlocks	10,16	16, 21, 24
tstocks	11,17	17, 22, 25
tbarrels	12,18	18, 23, 26
locks	13,19	14,16
stocks	15	17
barrels	15	18
lsales	24	27
ssales	25	27
bsales	26	27
sales	27	28, 29, 33, 34, 37, 39
comm	31, 32, 33, 36, 37, 39	32, 33, 37, 42

Selected Define/Use Paths for Commission problem

Test case id	Description	Variables Path(Beginning, End nodes)	Du Paths	Definition clear?	Comments
1	Check for lock price variable **DEF(lprice,7)** and **USE(lprice,24)**	(7 , 24)	<7-8-9-10-11-12-13-14-15-16-17-18-19-20-14-21-22-23-24>	Yes	
2	Check for Stock price variable **DEF(sprice,8)** and **USE(sprice,25)**	(8 , 25)	<8-9-10-11-12-13-14-15-16-17-18-19-20-14-21-22-23-24-25>	Yes	
3	Check for barrel price variable **DEF(bprice,9)** and **USE(bprice,26)**	(9 , 26)	<9-10-11-12-13-14-15-16-17-18-19-20-14-21-22-23-24-25-26>	Yes	
4	Check for total locks variable **DEF(tlocks,10) and DEF(tlocks,16)** and **3 usage nodes USE(tlocks,16), USE(tlocks,21), USE(tlocks,24)**	(10 , 16)	<10-11-12-13-14-15-16>	Yes	
		(10 , 21)	<10-11-12-13-14-15-16-17-18-19-20-14-21>	No	
		(10 , 24)	<10-11-12-13-14-15-16-17-18-19-20-14-21-22-23-24>	No	
		(16 , 16)	<16-16>	Yes	
		(16 , 21)	<16-17-18-19-14-21>	No	
		(16 , 24)	<16-17-18-19-20-14-21-22-23-24>	No	
5	Check for total stocks **variable DEF(tstocks,11) and DEF(tstocks,17)** and **3 usage nodes (USE(tstocks,17), USE(tstocks,22), USE(tstocks,25)**	(11 , 17)	<11-12-13-14-15-16-17>	Yes	
		(11 , 22)	<11-12-13-14-15-16-17-18-19-20-14-21-22>	No	
		(11, 25)	<11-12-13-14-15-16-17-18-19-20-14-21-22-23-24-25>	No	
		(17 , 17)	<17-17>	Yes	
		(17 , 22)	<17-18-19-20-14-21-22>	No	
		(17 , 25)	<17-18-19-20-14-21-22-23-24-25>	No	

6	check for locks variable **DEF(locks,13), DEF(locks,19) and USE(locks,14), USE(locks,16)**	(13 , 14)	<13-14>	Yes	Begin the loop
		(13 , 16)	<13-14-15-16>	Yes	
		(19 , 14)	<19-20-14>	Yes	
		(19 , 16)	<19-20-14-15-16>	Yes	Repeat the loop
7	Check for stocks variable **(DEF(stocks,15)** and **USE(stocks,17)**	(15 , 17)	<15-16-17>	Yes	
8	Check for sales variable **DEF(sales, 27)** and **USE(Sales, 28), USE(Sales , 29), USE(Sales,33) , USE(Sales , 34) , USE(Sales,37) , USE(Sales , 39)**	(27 ,28)	<27-28>	Yes	
		(27 , 29)	<27-28-29>	Yes	
		(27 , 33)	<27-28-29-30-31-32-33>	Yes	
		(27 , 34)	<27-28-29-34>	Yes	
		(27 , 37)	<27-28-29-34-35-36-37>	Yes	
		(27 , 39)	<27-28-29-34-38-39>	Yes	
9	Check for Commission variable **DEF(comm, 31,32,33) , DEF(comm,36,37) and DEF(comm,39)** and **USE(comm,42)**	((31,32,33),42)	<31-32-33-42>	Yes	
		((36 , 37) , 42)	<36-37-42>	Yes	
		(39 , 42)	<39 - 42>	Yes	

Program 10 (Binary Search- Path Testing)

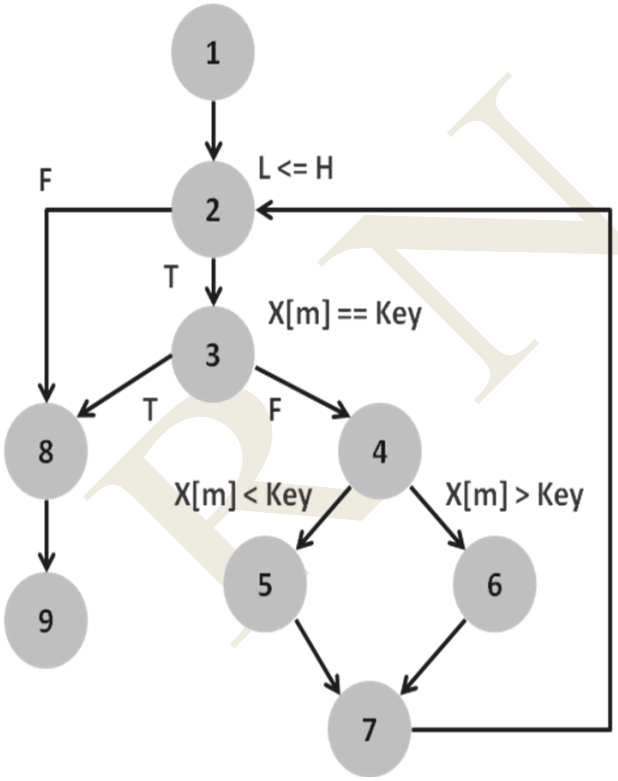
/* Design,develop a code and run the program in any suitable language to implement the binary search algorithm. Determine the basis paths and using them derive different test cases execute these test cases and discuss the test results */

```
#include<stdio.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");
 }
else
    printf("Number of element should be greater than zero\n");
   return 0;
}
```

Binary Search function with line number

```
int binsrc(int x[],int low, int high, int key)
{
int mid;                                  1
while(low<=high)                          2
{
mid=(low+high)/ 2;
if(x[mid]==key)                           3
return mid;                               8
if(x[mid]<key)                            4
low=mid+1;                                5
else
high=mid-1;                               6
}                                         7
return -1;                                8
}                                         9
```

Program Graph – for Binary Search



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/Issues:

Array has Elements in Ascending order	T/F
Key element is in the Array	T/F
Array has ODD number of Elements	T/F

Test Cases – Binary Search

Paths	Inputs		Expected Output	Remarks
	X[]	Key		
P1: 1-2-3-8-9	{10,20,30,40,50}	30	Success	Key $\in$ X[] and 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 05	Repeat and Failure	Key $\notin$ X[]
P4: 1-2-8-9	Empty	Any Key	Failure	Empty List

Program 11 (Quick Sort-Path Testing)

/*Design, develop, code and run the program in any suitable language to implement the Quick-Sort Algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.*/

```
#include<stdio.h>
void quicksort (int x[10], int first, int last)
{
int temp, pivot, i, j;
if(first<last)
{
pivot=first;
i=first;
j=last;
while(i<j)
{
while(x[i]<=x[pivot] && i<last)
i++;
        while(x[j]>x[pivot])
j--;
if(i<j)
 {
temp=x[i];
x[i]=x[j];
x[j]=temp;
}
  }
   temp=x[pivot];
   x[pivot]=x[j];
   x[j]=temp;
quicksort(x,first,j-1);
quicksort(x,j+1,last);
}
}
```

// main program

```
int main()
{
int a[20], i, key, n;
printf("enter the size of the array");
scanf("%d", &n);
if(n>0)
{
printf("enter the elements of the array");
for(i=0; i<n; i++)
scanf("%d", &a[i]);

quicksort (a,0,n-1);
printf("the elements in the sorted array is:\n");
        for(i=0; i<n; i++)
 printf("%d\t", a[i]);
}
else
{
printf("size of array is invalid\n");
}
```

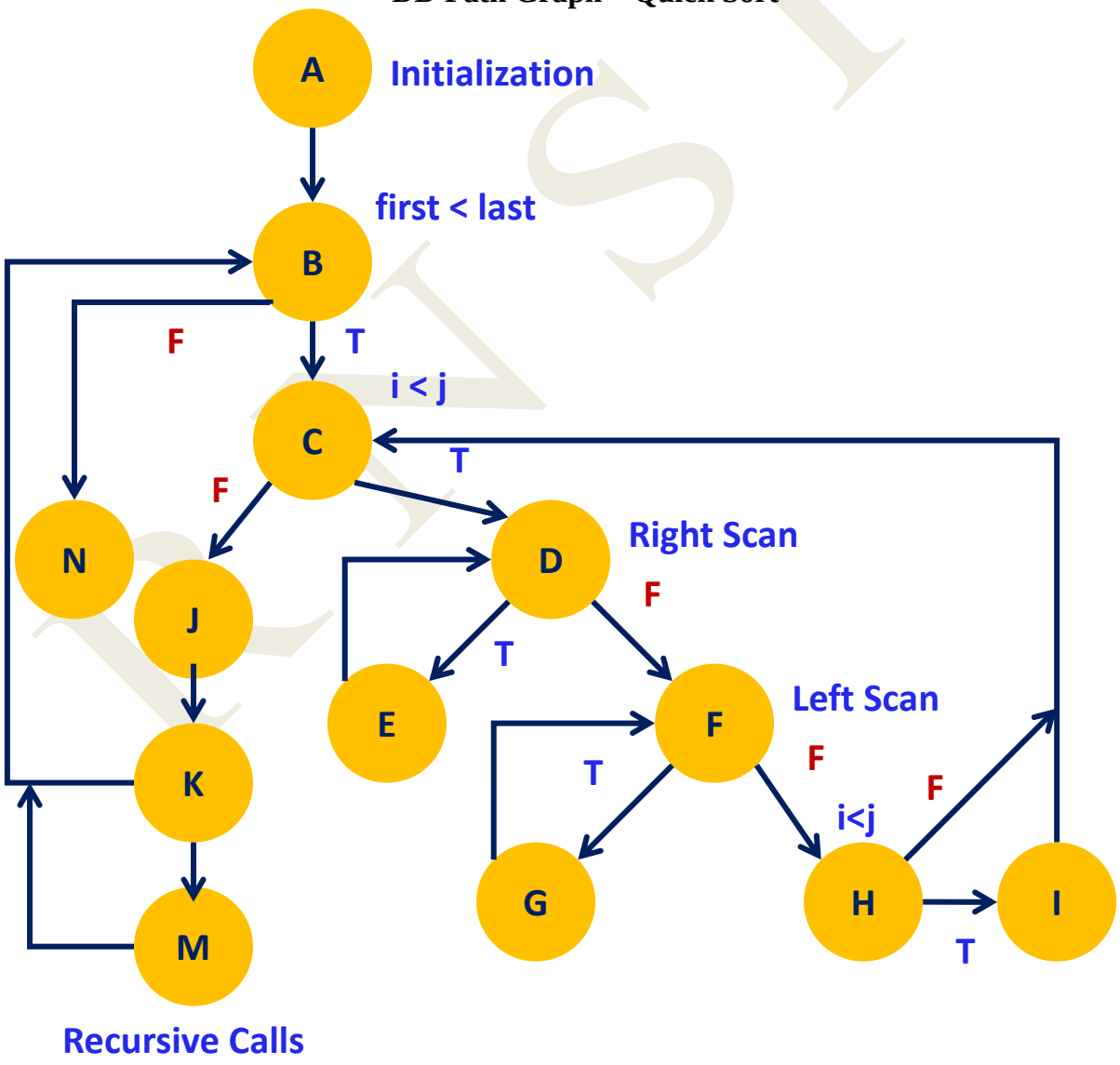
Quick sort function with line number

```
void quicksort (int x[10],int first,int last)
{
int temp, pivot, i, j;  1          A
if(first<last)  2                  B
{
 pivot=first;  3
 i=first;  4
 j=last;  5
 while(i<j)             6C
   {
while(x[i]<=x[pivot] && i<last) 7D
i++;  8E
while(x[j]>x[pivot])  9F
j--;  10 G
 if(i<j)   11 H
{
  temp=x[i];  12
  x[i]=x[j];  13 I
  x[j]=temp;  14
}
  }
temp=x[pivot];  15
x[pivot]=x[j];  16J
x[j]=temp;  17
quicksort (x,first,j-1);  18K
quicksort (x,j+1,last);  19 M
}
}   20N
```

Relation between program nodes and DD Path name

Node No.	DD Path name
1	A
2	B
6	C
7	D
8	E
9	F
10	G
11	H
12,13,14	I
15,16,17	J
18	K
19	M
20	N

DD Path Graph – Quick Sort



Independent Paths– Quick Sort

P1: A-B-N

P2: A-B-C-J-K-B

P3: A-B-C-J-K-M-B

P4: A-B-C-D-F-H-C

P5: A-B-C-D-F-H-I-C

P6: A-B-C-D-E-D-F-H

P7: A-B-C-D-F-G-F-H

Independent Paths:

No. of Edges=18, No. of Nodes=13,
No. of P=1

V (G) = E-N+2P = 18-13+2 = **7**

Pre-Conditions/Issues:

Array has only one Element, Two Elements, and Three Elements (6 Possibilities)

Array has Elements in ASC/DSC/Arbitrary (Any of the Permutations)

EX: 3 elements: 123, 132, 213, 231, 312, 321

Test Cases – Quick Sort

Paths	Inputs		Expected Output	Remarks
	X[]	First , Last		
P1: A-B-N	5	1,1	Sorted	Only one Elem
P2: A-B-C-J-K-B	5,4	1,2	Repeat & Sorted	Two Elements
P3: A-B-C-J-K-M-B	1,2,3 OR 3,1,2	1,3	Repeat & Sorted	Three Elements
P4: A-B-C-D-F-H-C	1,2,3,4,5	1,5	Repeat & Sorted	ASC Sequence
P5: A-B-C-D-F-H-I-C	5,4,3,2,1	1,5	Repeat & Sorted	DSC Sequence
P6: A-B-C-D-E-D-F-H	1,4,3,2,5 OR 2,2,2,2,2	1,5	Repeat & Sorted	Pivot is MIN
P7: A-B-C-D-F-G-F-H	5,2,3,1,4	1,5	Repeat & Sorted	Pivot is MAX

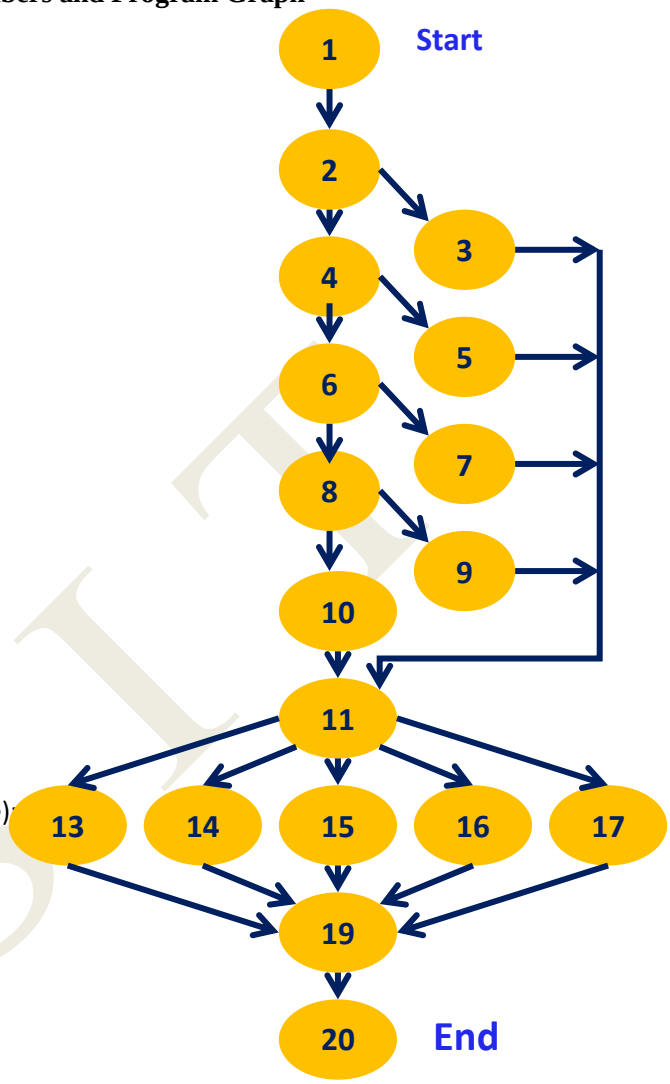
Program 12 (Absolute Letter Grading Path Testing)

/* Design, develop, code and run the program in any suitable language to implement an absolute letter grading procedure, making suitable assumptions. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results */

```
#include<stdio.h>
int main()
{
float per;
char grade;
      scanf("%f",&per);
      if(per>=90)
      grade= 'A';
      else if(per>=80 && per<90)
      grade ='B';
      else if(per>=70 && per<80)
      grade ='C';
      else if(per>=60 && per<70)
      grade='D';
      else grade='E';
              switch(grade)
      {
      case 'A': printf("\nEXCELLENT");   break;
      case 'B':printf("\nVery Good"); break;
      case 'C' : printf("\nGood"); break;
      case 'D': printf("\nAbove Average"); break;
      case 'E': printf("\n Satisfactory"); break;
      }
      printf("\t The percentage = %f and grade is %c ",per,grade);
       return 0;
 }
```

Absolute Grading Program with Line Numbers and Program Graph

```
int main()
{
   float per;
   char grade;
1.   scanf("%f",&per);
2.   if(per>=90)
3.   grade= 'A';
4.   else if(per>=80 && per<90)
5.   grade ='B';
6.   else if(per>=70 && per<80)
7.   grade ='C';
8.   else if(per>=60 && per<70)
9.   grade='D';
10.  else grade='E';
11.   switch(grade)
12.   {
13.  case 'A': printf("\nEXCELLENT");   break;
14.  case 'B':printf("\nVery Good"); break;
15.  case 'C' : printf("\nGood"); break;
16.  case 'D': printf("\nAbove Average"); break;
17.  case 'E': printf("\n Satisfactory"); break;
18.   }
19.  printf("\t The percentage = %f and grade is %c ",per,grade);
20.   return 0;
 }
```



Independent Paths:

#Edges=26, #Nodes=18, #P=1
V(G) = E-N+2P = 26-18+2 = **10**

P1: 1-2-3-11 A Grade
P2: 1-2-4-5-11 B Grade
P3: 1-2-4-6-7-11 C Grade
P4: 1-2-4-6-8-9-11 D Grade
P5: 1-2-4-6-8-10-11 E Grade

Case Construct

P6: 11-13-19-20 A Grade
P7: 11-14-19-20 B Grade
P8: 11-15-19-20 C Grade
P9: 11-16-19-20 D Grade
P10:11-17-19-20 E Grade

Feasible paths

P11: 1-2-3-11-13-19-20 A Grade
P12: 1-2-4-5-11-14-19-20 B Grade
P13: 1-2-4-6-7-11-15-19-20 C Grade
P14: 1-2-4-6-8-9-11-16-19-20 D Grade
P15: 1-2-4-6-8-10-11-17-19-20 E Grade

Infeasible paths
P16: 1-2-4-6-8-10-11-13-19-20
P17: 1-2-4-6-8-10-11-14-19-20
P18: 1-2-4-6-8-10-11-15-19-20
P19: 1-2-4-6-8-10-11-16-19-20
P20: 1-2-4-6-8-10-11-17-19-20

Pre-Conditions/Issues: Percentage **per** is a positive Float Number

Test Cases – Absolute Grading

Paths	Input	Expected Output	Remarks
P1: 1-2-3-11-13-19-20	**>=90**	**A Grade, Excellent**	**PASS**
P2: 1-2-4-5-11-14-19-20	**80-89**	**B Grade, Very Good**	**PASS**
P3: 1-2-4-6-7-11-15-19-20	**70-79**	**C Grade, Good**	**PASS**
P4: 1-2-4-6-8-9-11-16-19-20	**60-69**	**D Grade, Above Average**	**PASS**
P5: 1-2-4-6-8-10-11-17-19-20	**<60**	**E Grade, Satisfactory**	**PASS**