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.

Test Case Name :Decision Table for Commission Problem

Experiment Number : 3

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

Experiment Number : 3

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.

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			

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 a 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, equivalence class partitioning and decision-table approach and 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)
```

	satisfying precondition and a ≠b , b ≠ c and c ≠ a				Scalene triangle			

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

```
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	

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 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]);
```

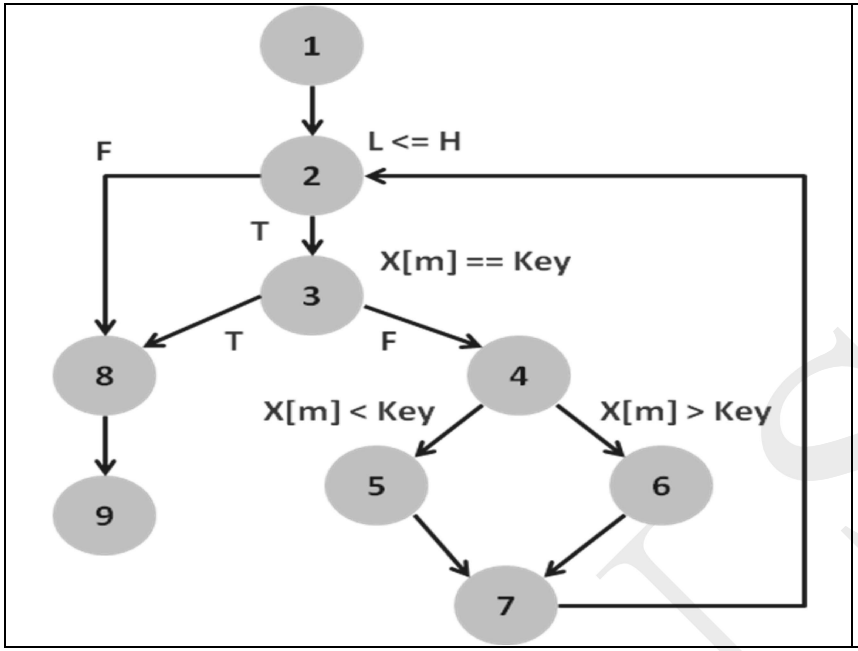
```
        printf("enter the key element to be searched\n");
            scanf("%d",&key);

 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;
}


int binsrc(int x[],int low, int high, int key)
{
        int mid;                                        1
        while(low<=high)                                2
                {
                mid=(low+high)/
                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 ∈ 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 ∉ X[]
P4: 1-2-8-9	Empty	Any Key	Failure	Empty List