EXPERIMENT No. 01: Triangle Problem-Boundary value Analysis

1.1 OBJECTIVE

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, execute the test cases and discuss the results.

1.2 REQUIREMENT SPECIFICATION

R1. The system should accept 3 positive integer numbers (a, b, c) which represents 3 sides of the triangle.
R2. Based on the input should determine if a triangle can be formed or not.
R3. If the requirement R2 is satisfied then the system should determine the type of the triangle, which can be

- Equilateral (i.e. all the three sides are equal)
- Isosceles (i.e Two sides are equal)
- Scalene (i.e All the three sides are unequal)

R4. Upper Limit for the size of any side is 10

1.3 DESIGN

ALGORITHM:

Step 1: Input a, b & c i.e three integer values which represent three sides of the triangle.

Step 2: if (a < (b + c)) and (b < (a + c)) and (c < (a + b) then do step 3

else

print not a triangle. do step 6

Technique used: Boundary value analysis

1. Test Case design

For BVA problem the test cases can be generation depends on the output and the constraints on the output. Here we least worried on the constraints on Input domain.

The Triangle problem takes 3 sides as input and checks it for validity, hence n = 3. Since BVA yields (4n + 1) test cases according to single fault assumption theory, hence we can say that the total number of test cases will be (4*3+1) =12+1=13.

The maximum limit of each sides a, b, and c of the triangle is 10 units according to requirement R4. So a, b and c lies between
$0 \leq a \leq 10$
$0 \leq b \leq 10$
$0 \leq c \leq 10$

Equivalence classes for a:
E1: Values less than 1.
E2: Values in the range.
E3: Values greater than 10.

Equivalence classes for b:
E4: Values less than 1.
E5: Values in the range.
E6: Values greater than 10.

Equivalence classes for c:
E7: Values less than 1.
E8: Values in the range.
E9: Values greater than 10.
From the above equivalence classes we can derive the following test cases using boundary value analysis approach.

1.4 PROGRAM CODE

```
#include<stdio.h>
#include<stdlib.h>
 int main()
{
    int a,b,c;
    printf("Enter the three sides of traingle");
    scanf("%d%d%d",&a,&b,&c);
    if((a<1||a>10)||(b<1||b>10)||(c<1||c>10))
    {
            printf("Out of range values");
            exit(0);
    }
    if((a<b+c)&&(b<a+c)&&(c<a+b))
    {
            if((a==b)&&(b==c))
            printf("Equilateral traingle");
            else if((a!=b)&&(b!=c)&&(a!=c))
            printf("Scalene traingle");
            else
            printf("Isosceles traingle");
    }
    else
    printf("Traingle cannot be formed");
    return 0;
}
```

1.5 TESTCASES

Table-1: Test case for Triangle Problem

Test Cases	Description	Inputs			Expected Output	Actual output	Status
		A	B	C			
BVA 1	a(nom),b(nom) and c(min)	5	5	1	Isosceles		
BVA 2	a(nom),b(nom) and c(min+)	5	5	2	Isosceles		
BVA 3	a(nom),b(nom) and c(nom)	5	5	5	Equilateral		
BVA 4	a(nom),b(nom) and c(max-)	5	5	9	Isosceles		
BVA 5	a(nom),b(nom) and c(max)	5	5	10	Triangle cannot be formed		
BVA 6	a(nom),b(min) and c(nom)	5	1	5	Isosceles		
BVA 7	a(nom),b(min+) and c(nom)	5	2	5	Isosceles		
BVA 8	a(nom),b(max-) and c(nom)	5	9	5	Isosceles		
BVA 9	a(nom),b(max) and c(nom)	5	10	5	Triangle cannot be formed		
BVA 10	a(min),b(nom) and c(nom)	1	5	5	Isosceles		
BVA 11	a(min+),b(nom) and c(nom)	2	5	5	Isosceles		
BVA 12	a(max-),b(nom) and c(nom)	9	5	5	Isosceles		
BVA 13	a(max),b(nom) and c(nom)	10	5	5	Triangle cannot be formed		

1.6 EXECUTION:

Execute the program and test the test cases in Table-1 against program and complete the table with for Actual output column and Status column

Test Report:

1 No. of test cases Executed:
2 No. of Defects Raised:
3 No of test cases Passed:
4 No of test cases failed:

1.7 SNAPSHOTS

```
user@user-Vostro-2520:~/ST$ ./a.out
Enter three sides of the triangle
5 5 10
a=5     ,b=5     ,c=10
triangle cannot be formed
user@user-Vostro-2520:~/ST$ ./a.out
Enter three sides of the triangle
10 11 12
a=10    ,b=11    ,c=12
Out of range
user@user-Vostro-2520:~/ST$ ./a.out
Enter three sides of the triangle
-1 2 3
a=-1    ,b=2     ,c=3
Out of range
user@user-Vostro-2520:~/ST$
```

1.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. What is an error?
2. What is the use of testing software?
3. What are the types of software testing?
4. What is boundary value analysis?
5. What is white box testing and list the types of white box testing?
6. What is black box testing? What are the different black box testing techniques?

EXPERIMENT No. 02: Commission Problem-Boundary value Analysis

2.1 Objective	2.5 Test cases
2.2 Requirement specification	2.6 Execution
2.3 Design	2.7 Snapshots
2.4 Program Code	2.8 Pre-experiment & Post-experiment questions

2.1 OBJECTIVE

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.

2.2 REQUIREMENT SPECIFICATION

Problem Definition: The Commission Problem includes a salesperson in the former Arizona Territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. Cost includes
Locks- $45
Stocks- $30
Barrels- $25
The salesperson had to sell at least one complete rifle per month and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 stocks and 90 barrels.
After each town visit, the sales person sent a telegram to the Missouri gunsmith with the number of locks, stocks and barrels sold in the town. At the end of the month, the salesperson sent a very short telegram showing --1 lock sold. The gunsmith then knew the sales for the month were complete and computed the salesperson‟s commission as follows:
On sales up to(and including) $1000= 10% On
the sales up to(and includes) $1800= 15% On
the sales in excess of $1800= 20%
The commission program produces a monthly sales report that gave the total number of locks, stocks and barrels sold, the salesperson's total dollar sales and finally the commission

2.3 DESIGN

ALGORITHM:

STEP 1: Define lockPrice=45.0, stockPrice=30.0, barrelPrice=25.0
STEP2: Input locks
STEP3: while(locks!=-1) „input device uses -1 to indicate end of data goto STEP 12
STEP4:input (stocks, barrels)
STEP5: compute lockSales, stockSales, barrelSales and sales
STEP6: output("Total sales:" sales)
STEP7: if (sales > 1800.0) goto STEP 8 else goto STEP 9
STEP8: commission=0.10*1000.0; commission=commission+0.15 * 800.0;
commission = commission + 0.20 * (sales-1800.0)

STEP9: if (sales > 1000.0) goto STEP 10 else goto STEP 11
STEP10: commission=0.10* 1000.0; commission=commission + 0.15 * (sales-1000.0)
STEP11: Output ("Commission is $", commission)
STEP12: exit

2.4 PROGRAM CODE:

```
#include<stdio.h>
 int main()
{
   int locks,stocks,barrels,tlocks,tstocks,tbarrels;
   float lprice,sprice,bprice,sales=0,comm;
   int c1,c2,c3,temp;
   tlocks=0,tstocks=0,tbarrels=0;
   lprice=45.0,sprice=30.0,bprice=25.0;
   printf("Enter the number of locks and to exit the loop press -1\n");
   scanf("%d",&locks);
   while(locks!=-1)
   {
        c1=(locks<=0||locks>70);
        printf("Enter the number of stocks and barrels");
        scanf("%d%d",&stocks,&barrels);
        c2=(stocks<=0||stocks>80); c3=(barrels<=0||barrels>90);
        if(c1)
        printf("Value of locks not in range 1--70");
        else
        {
              temp=locks+tlocks; if(temp>70)
              printf("New value of locks = %d not in range 1--70",temp);
              else
        }
        tlocks=temp;
        printf("Total locks = %d\n",tlocks);
        if(c2)
        printf("Value of stocks not in range 1--80");
        else
        {
              temp=stocks+tstocks;
              if(temp>80)
              printf("New value of stocks = %d not in range 1--80",temp);
              else
        }
        tstocks=temp;
```

```
        printf("Total stocks = %d\n",tstocks);
        if(c3)
        printf("Value of barrels not in range 1--90");
        else
        {
                temp=barrels+tbarrels;
                if(temp>90)
                printf("New value of barrels = %d not in range 1--90",temp);
                else
        }
        tbarrels=temp;
        printf("Total barrels = %d\n",tbarrels);
        printf("Enter the number of locks and press -1 to exit the loop\n");
        scanf("%d",&locks);
    }
    if(tlocks>0&&tstocks>0&&tbarrels>0)
    {
        printf("Total locks = %d\nTotal stocks = %d\nTotal barrels =
                %d\n",tlocks,tstocks,tbarrels);
        sales=((tlocks*lprice)+(tstocks*sprice)+(tbarrels*bprice));
        printf("Total sales is %f\n",sales); if(sales>0)
        {
                if(sales>1800)
                {
                        comm=0.10*1000.0;
                        comm=comm+0.15*800.0;
                        comm=comm+0.20*(sales-1800);
                }
                else if(sales>1000)
                {
                        comm=0.10*1000.0;
                        comm=comm+0.15*(sales-1000);
                }
                else
                comm=0.10*sales; printf("Commission is %f \n",comm);
        }
    }
    else
    printf("There is no sales\n");
    return 0;
}
```

2.5 TESTCASES

Table-1: **BVA Test case for Commission Problem**

Test cases	Description	Inputs			Expected Output		Actual Output		Status
		Locks	Stocks	Barrels	Sales	Com	Sales	Com	
1	locks(nom), stocks(nom) and barrels(min)	35	40	1	2800	420			
2	locks(nom), stocks(nom) and barrels(min+)	35	40	2	2825	425			
3	locks(nom), stocks(nom) and barrels(nom)	35	40	45	3900	640			
4	locks(nom), stocks(nom) and barrels(max-)	35	40	89	5000	860			
5	locks(nom), stocks(nom) and barrels(max)	35	40	90	5025	865			
6	locks(nom), stocks(min) and barrels(nom)	35	1	45	2730	406			
7	locks(nom), stocks(min+) and barrels(nom)	35	2	45	2760	412			
8	locks(nom), stocks(max-) and barrels(nom)	35	79	45	5070	874			
9	locks(nom), stocks(max) and barrels(nom)	35	80	45	5100	880			
10	locks(min), stocks(nom) and barrels(nom)	1	40	45	2370	334			
11	locks(min+), stocks(nom) and barrels(nom)	2	40	45	2415	343			
12	locks(max-), stocks(nom) and barrels(nom)	69	40	45	5430	946			
13	locks(max), stocks(nom) and barrels(nom)	70	40	45	5475	955			

This is how we can apply BVA technique to create test cases for our Commission Problem.

2.6 EXECUTIONS

Execute the program and test the test cases in Table-1 against program and complete the table with for Actual output column and Status column**.**

TEST REPORT:

1. No of TC‟s Executed:
2. No of Defects Raised:
3. No of TC‟s Pass:
4. No of TC‟s Failed:

2.7 SNAPSHOTS:

2.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. What is Testing?
2. What is fault? Name types of faults
3. What is commission problem?
4. What is boundary value analysis?
5. What is debugging?

EXPERIMENT No. 03: Next Date Function-Boundary value Analysis

3.1 Objective	3.5 Test cases
3.2 Requirement specification	3.6 Execution
3.3 Design	3.7 Snapshots
3.4 Program Code	3.8 Pre-experiment & Post-experiment questions

3.1 OBJECTIVE

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, derive different test cases, execute these test cases and discuss the test results.

3.2 REQUIREMENT SPECIFICATION

Problem Definition: "Next Date" is a function consisting of three variables like: month, date and year. It returns the date of next day as output. It reads current date as input date.

The constraints are

C1: $1 \leq \text{month} \leq 12$

C2: $1 \leq \text{day} \leq 31$

C3: $1812 \leq \text{year} \leq 2012$.

If any one condition out of C1, C2 or C3 fails, then this function produces an output "value of month not in the range 1...12".

Since many combinations of dates can exist, hence we can simply displays one message for this function: "Invalid Input Date".

A very common and popular problem occurs if the year is a leap year. We have taken into consideration that there are 31 days in a month. But what happens if a month has 30 days or even 29 or 28 days?

A year is called as a leap year if it is divisible by 4, unless it is a century year. Century years are leap years only if they are multiples of 400. So, 1992, 1996 and 2000 are leap years while 1900 is not a leap year.

3.3 DESIGN

Algorithm

STEP 1: Input date in format DD.MM.YYYY

STEP2: if MM is 01, 03, 05,07,08,10 do STEP3 else STEP6

STEP3:if DD < 31 then do STEP4 else if DD=31 do STEP5 else output(Invalid Date);

STEP4: tomorrowday=DD+1 goto STEP18

STEP5: tomorrowday=1; tomorrowmonth=month + 1 goto STEP18

STEP6: if MM is 04, 06, 09, 11 do STEP7

STEP7: if DD<30 then do STEP4 else if DD=30 do STEP5 else output(Invalid Date);

STEP8: if MM is 12

STEP9: if DD<31 then STEP4 else STEP10

STEP10: tomorrowday=1, tommorowmonth=1, tommorowyear=YYYY+1; goto STEP18

STEP11: if MM is 2

STEP12: if DD<28 do STEP4 else do STEP13

STEP13: if DD=28 & YYYY is a leap do STEP14 else STEP15

STEP14: tommorowday=29 goto STEP18

STEP15: tommorowday=1, tomorrowmonth=3, goto STEP18;

STEP16: if DD=29 then do STEP15 else STEP17

STEP17: output("Cannot have feb", DD); STEP19

STEP18: output(tomorrowday, tomorrowmonth, tomorrowyear);

STEP19: exit

3.4 PROGRAM CODE:

```
#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,tday,tmonth,tyear;
        char flag;
```

```
do{
        flag='y';
        printf("Enter todays date in ddmmyy format\n");
        scanf("%d%d%d",&day,&month,&year);
        tmonth=month;
        tyear=year;
        if(day<1||day>31)
        {
                printf("Value of day not in range 1--31\n");
                flag='n';
        }
        if(month<1||month>12)
        {
                printf("Value of month not in range 1--12\n");
                flag='n';
        }
        else if(check(day,month))
        {
                printf("Value of day not in range <=30");
                flag='n';
        }
        if(year<1812||year>2018)
        {
                printf("Value of year not in range 1812--2018");
                flag='n';
        }
        if(month==2)
        {
                if(isleap(year)&&day>29)
                {
                        printf("Invalid input for leap year");
                        flag='n';
                }
                if(!isleap(year)&&day>28)
                {
                        printf("Invalid input date 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)
        tday=day+1;
        else
```

```
                {
                        tday=1;
                        tmonth=month+1;
                }
                break;
                case 4:
                case 6:
                case 9:
                case 11: if(day<30)
                tday=day+1;
                else
                {
                        tday=1;
                        tmonth=month+1;
                }
                break;
                case 12:if(day<31)
                tday=day+1;
                else
                {
                        tday=1;
                        tmonth=1;
                        if(year==2018)
                        {
                                printf("Next year is out of boundary value of year");
                        }
                        else
                        tyear=year+1;
                }
                break;
                case 2: if(day<28)
                tday=day+1;
                else if(isleap(year)&&day==28)
                tday=day+1;
                else if(day==28||day==29)
                {
                        tday=1;
                        tmonth=3;
                }
                break;
        }
        printf("Next day is %d/%d/%d\n",tday,tmonth,tyear);
        return 0;
}
```

3.5 TESTCASES

Technique used: Boundary value analysis

Table-1: Test case for Next Date Problem

Test cases	Description	Inputs			Output	Status
		DD	MM	YY		
1	day(nom),month(nom) and year(min)	15	6	1812	16/6/1812	
2	day(nom),month(nom) and year(min+)	15	6	1813	16/6/1813	
3	day(nom),month(min) and year(nom)	15	6	1912	16/6/1912	
4	day(nom),month(nom) and year(max-)	15	6	2011	16/6/2011	
5	day(nom),month(nom) and year(max)	15	6	2012	16/6/2012	
6	day(nom),month(min) and year(nom)	15	1	1912	16/1/1912	
7	day(nom),month(min+) and year(nom)	15	2	1912	16/2/1912	
8	day(nom),month(max-) and year(nom)	15	11	1912	16/11/1912	
9	day(nom),month(max) and year(nom)	15	12	1912	16/12/1912	
10	day(min),month(nom) and year(nom)	1	6	1912	2/6/1912	
11	day(min+),month(nom) and year(nom)	2	6	1912	3/6/1912	
12	day(max-),month(nom) and year(nom)	30	6	1912	1/7/1912	
13	day(max),month(nom) and year(nom)	31	6	1912	Day out of range for the month	

This is how we can apply BA technique to create test cases for our Next Date Problem.

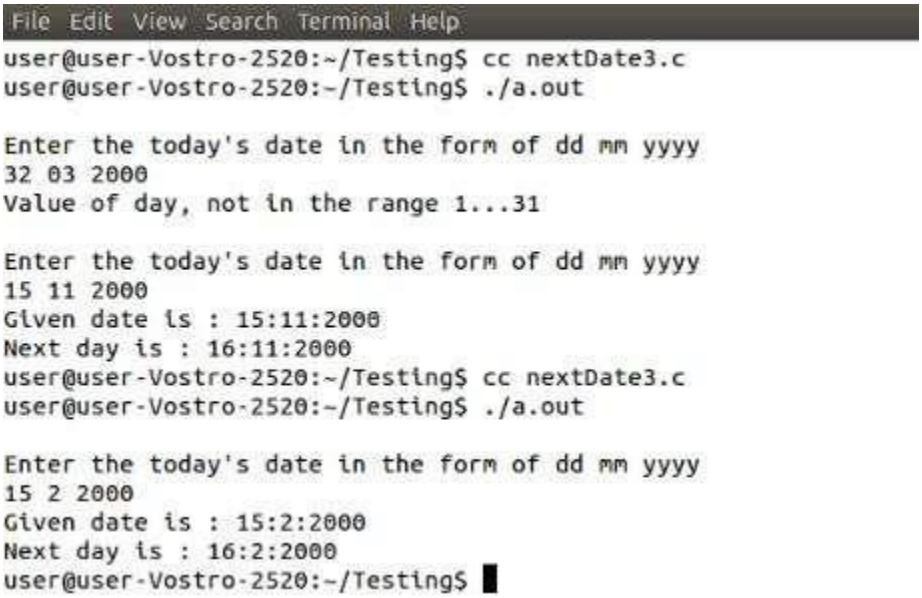
3.6 EXECUTIONS

Execute the program and test the test cases in Table-1 against program and complete the table with for Actual output column and Status column

Test Report:

1. No of TC"s Executed:
2. No of Defects Raised:
3. No of TC"s Pass:
4. No of TC"s Failed:

3.7 SNAPSHOTS:



3.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. Differentiate between Functional Testing and Structural Testing.
2. What are the conditions for next date problem?
3. What is the test case?
4. What is glass box testing?
5. What is the procedure for manual testing?

EXPERIMENT No. 04: Triangle Problem-Equivalence Class Partitioning

4.1 Objective:

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 equivalence class partitioning, execute the test cases and discuss the results.

4.2 Requirement Specification:

R1. The system should accept 3 positive integer numbers (a, b, c) which represents 3 sides of the triangle.
R2. Based on the input should determine if a triangle can be formed or not.
R3. If the requirement R2 is satisfied then the system should determine the type of the triangle, which can be

- Equilateral (i.e. all the three sides are equal)
- Isosceles (i.e. two sides are equal)
- Scalene (i.e. All the three sides are unequal)

R4. Upper Limit for the size of any side is 10

4.3 DESIGN

Form the given requirements we can draw the following conditions:
C1: a<b+c?
C2: b<a+c?
C3: c<a+b?
C4: a=b?
C5: a=c?
C6: b=c?
According to the property of the triangle, if any one of the three conditions C1, C2 and C3 are not satisfied then triangle cannot be constructed. So only when C1, C2 and C3 are true the triangle can be formed, then depending on conditions C4, C5 and C6 we can decide what type of triangle will be formed. (i.e requirement R3).

ALGORITHM:

Step 1: Input a, b & c i.e three integer values which represent three sides of the triangle.
Step 2: if (a < (b + c)) and (b < (a + c)) and (c < (a + b) **then**
do step 3
else

print not a triangle. **do** step 6.

Step 3: if (a=b) and (b=c) **then**

Print triangle formed is equilateral. **do** step 6.

Step 4: if (a ≠ b) and (a ≠ c) and (b ≠ c) **then**

Print triangle formed is scalene**. do** step 6.

Step 5: Print triangle formed is Isosceles.

Step 6: stop

4.4 PROGRAM CODE

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int a,b,c;
        printf("Enter the three sides of traingle");
        scanf("%d%d%d",&a,&b,&c);
        if((a<1||a>10)||(b<1||b>10)||(c<1||c>10))
        {
                printf("Out of range values");
                exit(0);
        }
        if((a<b+c)&&(b<a+c)&&(c<a+b))
        {
                if((a==b)&&(b==c))
                printf("Equilateral traingle");
                else
                if((a!=b)&&(b!=c)&&(a!=c))
                printf("Scalene traingle");
                else
                printf("Isosceles traingle");
        }
        else
        printf("Traingle cannot be formed");
        return 0;
}
```

4.5 TESTCASES

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		a	b	C			
1	Three sides are different	3	4	5	Scalene	Scalene	Pass
2	All Values are same	5	5	5	Equilateral	Equilateral	Pass
3	Two sides are same, one side is different	2	5	5	Isosceles	Isosceles	Pass
4	Value of a=10, b=5, c=5	10	5	5	Triangle cannot be formed	Triangle cannot be formed	Pass

Table-1: Weak Normal and Strong normal equivalence Test case for Triangle Problem

Second attempt

The strong normal equivalence class test cases can be generated by using following possibilities:

D1 = {<a, b, c>: a=b=c}
D2 = {<a, b, c>: a=b, a≠ c}
D3= {<a, b, c>: a=c, a≠ b}
D4 = {<a, b, c>: b=c, a≠ b}
D5 = {<a, b, c>: a≠ b, a≠ c, b≠ c}
D6 = {<a, b, c>: a≥ b+ c}
D7 = {<a, b, c>: b≥ a+ c}
D8 = {<a, b, c>: c≥ a+ b}

Table-2 Weak Robust Test case for Triangle Problem

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		a	b	C			
1	Since a>10 the values of one variable is greater than the range	11	3	2	Out of range	Out of range	Pass
2	Since b>10 the values of one variable is greater than the range	3	11	2	Out of range	Out of range	Pass
3	Since c>10 the values of one variable is greater than 10	3	2	11	Out of range	Out of range	Pass
4	Since a<1the value of one variable is lesser than the range	0	4	5	Out of range	Out of range	Pass
5	Since b<1the value of one variable is lesser than the range	4	0	5	Out of range	Out of range	Pass
6	Since c<1the value of one variable is lesser than the range	4	5	0	Out of range	Out of range	Pass

Table-3: Strong Robust Test case for Triangle Problem

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		a	b	C			
1	Since a, b>10 the values of two variables is greater than the range	11	11	5	Out of range	Out of range	Pass
2	Since b, c>10 the values of two variables is greater than the range	5	11	11	Out of range	Out of range	Pass

3	Since a, c>10 the values of two variables is greater than range	11	5	11	Out of range	Out of range	Pass
4	Since a, b<0 the value of two variables is lesser than the range	0	0	5	Out of range	Out of range	Pass
5	Since b, c<0the value of two variables is lesser than the range	5	0	0	Out of range	Out of range	Pass
6	Since a, c<0 the value of two variables is lesser than the range	0	5	0	Out of range	Out of range	Pass
7	Since all the values greater than 10 i.e greater than the range	11	11	11	Out of range	Out of range	Pass
8	Since all the values lesser than 0 i.e lesser than the range	0	0	0	Out of range	Out of range	Pass

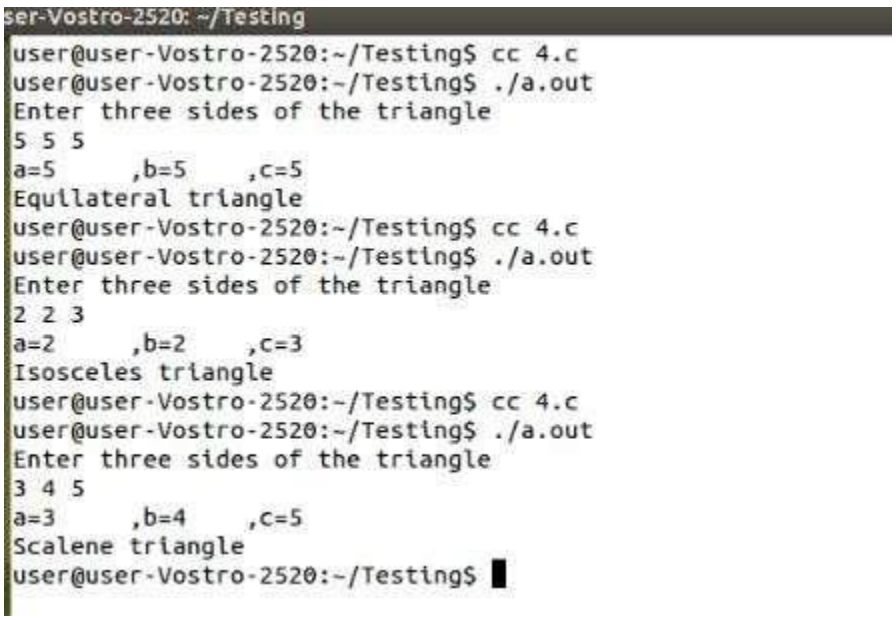
4.6 EXECUTION:

Execute the program and test the test cases in Table-1, Table-2 and Table-3 against program and complete the table with for Actual output column and Status column

Test Report

1. No of TC"s Executed:
2. No of Defects Raised:
3. No of TC"s Pass:
4. No of TC"s Failed:

4.7 SNAPSHOTS:



4.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. What is equivalence partitioning?
2. What is black box testing?
3. What are the different black box testing techniques?
4. What is Non-Functional Testing?
5. What is Functional Testing?

EXPERIMENT No. 05: Commission Problem-Equivalence Class Partitioning

5.1 Objective	5.5 Test cases
5.2 Requirement specification	5.6 Execution
5.3 Design	5.7 Snapshots
5.4 Program Code	5.8 Pre-experiment & Post-experiment questions

5.1 OBJECTIVES:

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

5.2 REQUIREMENT SPECIFICATION

Problem Definition: The Commission Problem includes a salesperson in the former Arizona Territory sold rifle locks, stocks and barrels made by a gunsmith in Missouri. Cost includes

Locks- $45
Stocks- $30
Barrels- $25

The salesperson had to sell at least one complete rifle per month and production limits were such that the most the salesperson could sell in a month was 70 locks, 80 stocks and 90 barrels.

After each town visit, the sales person sent a telegram to the Missouri gunsmith with the number of locks, stocks and barrels sold in the town. At the end of the month, the salesperson sent a very short telegram showing --1 lock sold. The gunsmith then knew the sales for the month were complete and computed the sales person's commission as follows:

On sales up to(and including) $1000= 10% On
the sales up to(and includes) $1800= 15% On
the sales in excess of $1800= 20%

The commission program produces a monthly sales report that gave the total number of locks, stocks and barrels sold, the sales person's total dollar sales and finally the commission

5.3 DESIGN

Algorithm

STEP 1: Define lockPrice=45.0, stockPrice=30.0, barrelPrice=25.0

STEP2: Input locks

STEP3: while(locks!=-1) „input device uses -1 to indicate end of data goto STEP12

STEP4:input (stocks, barrels)

STEP5: compute lockSales, stockSales, barrelSales and sales

STEP6: output("Total sales:" sales)

STEP7: if (sales > 1800.0) goto STEP 8 else goto STEP 9

STEP8:commission=0.10*1000.0;commission=commission+0.15*800.0;**commissio**=

commission + 0.20 * (sales-1800.0)

STEP9: if (sales > 1000.0) goto STEP 10 else goto STEP 11

STEP10: commission=0.10* 1000.0; commission=commission + 0.15 * (sales-

1000.0)

STEP11: Output("Commission is $", commission)

STEP12: exit

5.4 PROGRAM CODE:

```
#include<stdio.h>
int main()
{
        int locks,stocks,barrels,tlocks,tstocks,tbarrels;
        float lprice,sprice,bprice,sales=0,comm;
        int c1,c2,c3,temp;
        tlocks=0,tstocks=0,tbarrels=0;
        lprice=45.0,sprice=30.0,bprice=25.0;
        printf("Enter the number of locks and to exit the loop press -1\n");
        scanf("%d",&locks);
        while(locks!=-1)
        {
                c1=(locks<=0||locks>70);
                printf("Enter the number of stocks and barrels");
                scanf("%d%d",&stocks,&barrels);
                c2=(stocks<=0||stocks>80);
                c3=(barrels<=0||barrels>90);
                if(c1)
                printf("Value of locks not in range 1--70");
                else
                {
                        temp=locks+tlocks;
                        if(temp>70)
                        printf("New value of locks = %d not in range 1--70",temp);
                        else
                        tlocks=temp;
                }
                printf("Total locks = %d\n",tlocks);
                if(c2)
                printf("Value of stocks not in range 1--80");
                else
                {
                        temp=stocks+tstocks;
                        if(temp>80)
                        printf("New value of stocks = %d not in range 1--80",temp);
                        else
                        tstocks=temp;
```

```
                }
                printf("Total stocks = %d\n",tstocks);
                if(c3)
                printf("Value of barrels not in range 1--90");
                else
                {
                        temp=barrels+tbarrels;
                        if(temp>90)
                        printf("New value of barrels = %d not in range 1--90",temp);
                        else
                        tbarrels=temp;
                }
                printf("Total barrels = %d\n",tbarrels);
                printf("Enter the number of locks and press -1 to exit the loop\n");
                scanf("%d",&locks);
        }
        if(tlocks>0&&tstocks>0&&tbarrels>0)
        {
                printf("Total locks = %d\nTotal stocks = %d\nTotal barrels =
                %d\n",tlocks,tstocks,tbarrels);
                sales=((tlocks*lprice)+(tstocks*sprice)+(tbarrels*bprice));
                printf("Total sales is %f\n",sales);
                if(sales>0)
                {
                        if(sales>1800)
                        {
                                comm=0.10*1000.0;
                                comm=comm+0.15*800.0;
                                comm=comm+0.20*(sales-1800);
                        }
                        else if(sales>1000)
                        {
                                comm=0.10*1000.0;
                                comm=comm+0.15*(sales-1000);
                        }
                        else
                        comm=0.10*sales;
                        printf("Commission is %f \n",comm);
                }
        }
        else
        printf("There is no sales\n");

        return 0;
}
```

5.5 TESTCASES:

First attempt

We will have eight weak robust test cases.

TC Id	Test Case Description	Input Data			Expected Output		Actual Output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1	When lock=-1	-1	40	45	Terminates out of loop				
2	When all values are within range	30	40	45	3675	595			
3	Locks out ofrange	-2	40	45	No sales	No commission			
4	Stocks out ofrange	35	-2	45	No sales	No commission			
5	Barrels out ofrange	35	40	-2	No sales	No commission			
6	Locks out ofrange	71	40	45	No sales	No commission			
7	Stocks out ofrange	35	81	45	No sales	No commission			
8	Barrels out ofrange	35	40	91	No sales	No commission			

Second attempt:

Strong robust equivalence class test cases:

TC Id	Test Case Description	Input Data			Expected Output		Actual Output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1	Locks & stocks out of range	-2	-2	45	No sales	No commission			
2	Stocks & barrels out of range	35	-2	-2	No sales	No commission			
3	Locks & Barrels out of range	-2	40	-2	No sales	No commission			
4	Locks & Barrels out of range	71	40	91	No sales	No commission			
5	Stocks & barrels out of range	35	81	91	No sales	No commission			
6	Locks & Stocks out of range	71	81	45	No sales	No commission			
7	Locks , Stocks & barrels out of range	-2	-2	-2	No sales	No commission			
8	Locks , Stocks & barrels out of range	71	81	91	No sales	No commission			

Weak Normal & Strong Normal

TC Id	Test Case Description	Input Data			Expected Output		Actual Output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1	All normal values for locks, stocks &barrels	35	40	45	3900	640			

5.6 EXECUTIONS

Execute the program and test the test cases in Table-1 against program and complete the table with for Actual output column and Status column

Test Report:

1. No of TC‟s Executed:
2. No of Defects Raised:
3. No of TC‟s Pass:
4. No of TC‟s Failed:

5.7 SNASHOTS

```
Enter the total number of locks
-1
values of locks not in the range of 1..70
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
1
Enter the total number of stocks
-1
values of stocks not in the range of 1..80
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
1
Enter the total number of stocks
1
Enter the total number of barrelss
-1
values of stocks not in the range of 1..80
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
80
values of locks not in the range of 1..70
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
70
Enter the total number of stocks
80
Enter the total number of barrelss
90
The total sales is 7800
The commission is 1420.000000
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
80
values of locks not in the range of 1..70
user@user-Vostro-2520:~/ST$ ./a.out
Enter the total number of locks
70
Enter the total number of stocks
90
values of stocks not in the range of 1..80
user@user-Vostro-2520:~/ST$
```

5.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. What is an error?
2. What is a failure?
3. What is weak robust and strong robust ?
4. What is the difference between weak normal and strong normal?
5. How we perform equivalence class partitioning?

EXPERIMENT No. 06: NextDate Function-Equivalence Class Value

6.1 Objective	6.5 Test cases
6.2 Requirement specification	6.6 Execution
6.3 Design	6.7 Snapshots
6.4 Program Code	6.8 Pre-experiment & Post-experiment questions

6.1 OBJECTIVES:

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.

6.2 REQUIREMENT SPECIFICATION

Problem Definition: "Next Date" is a function consisting of three variables like: month, date and year. It returns the date of next day as output. It reads current date as input date.
The constraints are
C1:1≤month≤12
C2:1≤day≤31
C3: 1812 ≤ year ≤ 2012.
If any one condition out of C1, C2 or C3 fails, then this function produces an output "value of month not in the range 1...12".

Since many combinations of dates can exist, hence we can simply displays one message for this function: "Invalid Input Date".

A very common and popular problem occurs if the year is a leap year. We have taken intoconsideration that there are 31 days in a month. But what happens if a month has 30 days or even 29 or 28 days ?

A year is called as a leap year if it is divisible by 4, unless it is a century year. Century years are leap years only if they are multiples of 400. So, 1992, 1996 and 2000 are leap years while 1900 is not a leap year.

Furthermore, in this Next Date problem we find examples of Zipf's law also, which states that**"80% of the activity occurs in 20% of the space".** Thus in this case also, much of the source code of Next Date function is devoted to the leap year considerations.

6.3 DESIGN

Algorithm

STEP 1: Input date in format DD.MM.YYYY
STEP2: if MM is 01, 03, 05,07,08,10 do STEP3 else STEP6
STEP3:if DD < 31 then do STEP4 else if DD=31 do STEP5 else output(Invalid Date);
STEP4: tomorrowday=DD+1 goto STEP18
STEP5: tomorrowday=1; tomorrowmonth=month + 1 goto STEP18
STEP6: if MM is 04, 06, 09, 11 do STEP7

STEP7: if DD<30 then do STEP4 else if DD=30 do STEP5 else output(Invalid Date);
STEP8: if MM is 12
STEP9: if DD<31 then STEP4 else STEP10
STEP10: tomorrowday=1, tommorowmonth=1, tommorowyear=YYYY+1; goto STEP18
STEP11: if MM is 2
STEP12: if DD<28 do STEP4 else do STEP13
STEP13: if DD=28 & YYYY is a leap do STEP14 else STEP15
STEP14: tommorowday=29 goto STEP18
STEP15: tommorowday=1, tomorrowmonth=3, goto STEP18;
STEP16: if DD=29 then do STEP15 else STEP17
STEP17: output("Cannot have feb", DD); STEP19
STEP18: output(tomorrowday, tomorrowmonth, tomorrowyear);
STEP19: exit

6.4 PROGRAM CODE:

```
#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,tday,tmonth,tyear;
        char flag;
        do{
                flag='y';
                printf("Enter todays date in ddmmyy format\n");
                scanf("%d%d%d",&day,&month,&year);
                tmonth=month;
                tyear=year;
                if(day<1||day>31)
                {
                        printf("Value of day not in range 1--31\n");
                        flag='n';
```

```
        }
        if(month<1||month>12)
        {
                printf("Value of month not in range 1--12\n");
                flag='n';
        }
        else if(check(day,month))

        {
                printf("Value of day not in range <=30");
                flag='n';
        }
        if(year<1812||year>2018)
        {
                printf("Value of year not in range 1812--2018");

                flag='n';
        }
        if(month==2)
        {
                if(isleap(year)&&day>29)
                {
                        printf("Invalid input for leap year");
                        flag='n';
                }
                if(!isleap(year)&&day>28)
                {
                        printf("Invalid input date 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)
        tday=day+1;
        else
        {
```

```
                tday=1;
                tmonth=month+1;
            }
            break;
            case 4:
            case 6:
            case 9:
            case 11: if(day<30)
            tday=day+1;
            else
            {
                tday=1;
                tmonth=month+1;
            }
            break;
            case 12:if(day<31)
            tday=day+1;
            else
            {
                tday=1;

                tmonth=1;
                if(year==2018)
                {
                    printf("Next year is out of boundary value of year");
                }
                else
                tyear=year+1;
            }
            break;
            case 2: if(day<28)
            tday=day+1;
            else if(isleap(year)&&day==28)
            tday=day+1;
            else if(day==28||day==29)
            {
                tday=1;
                tmonth=3;
            }
            break;
        }
        printf("Next day is %d/%d/%d\n",tday,tmonth,tyear);
        return 0;
}
```

6.5 TESTING

Test Case design

The NextDate function is a function which will take in a date as input and produces as output the next date in the Georgian calendar. It uses three variables (month, day and year) which each have valid and invalid intervals.

Table 1: Valid Case

TC Id	Test Case Description	Input Data			ExpectedOutput	Actual Output	Status
		DD	MM	YYYY			
1	Valid case	15	6	1900	16/6/1900		

Weak Robust:

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		DD	MM	YYYY			
1	Valid Range	15	6	1912	16/6/1912		
2	Invalid date	-1	6	1912	Day not in range		
3	Invalid month	15	-1	1912	Month not in range		
4	Invalid year	15	6	1911	year not in range		
5	Invalid date	32	6	1912	Day not in range		
6	Invalid month	15	13	1912	Month not in range		
7	Invalid year	15	6	2013	year not in range		

Strong Robust:

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		DD	MM	YYYY			
1	Invalid date & month	-1	-1	1912	Not in range		
2	Invalid date & year	-1	6	-1	Not in range		
3	Invalid month & year	15	-1	-1	Not in range		
4	Invalid date & month	32	13	1912	Not in range		
5	Invalid date & year	32	6	2013	Not in range		

6	Invalid month & year	15	13	2013	Not in range		
7	Invalid date, month & year	-1	-1	-1	Not in range		
8	Invalid date, month & year	32	13	2013	Not in range		

Strong Normal:

TC ID	Test Case Description	Input Data			Expected Output	Actual Output	Status
		DD	MM	YYYY			
1	Valid Condition	14	6	2000	15/6/2000		
2	Valid Input	14	6	1996	15/6/1996		
3	Valid Input	14	6	2002	15/6/2002		
4	Valid Input	29	6	2000	30/6/2000		
5	Valid Input	29	6	1996	30/6/1996		
6	Valid Input	29	6	2002	30/6/2002		
7	Valid Input	30	6	2000	1/7/2000		
8	Valid Input	30	6	1996	1/7/1996		
9	Valid Input	30	6	2002	1/7/2002		
10	Invalid Range	31	6	2000	Not in Range		
11	Invalid Range	31	6	1996	Not in Range		
12	Invalid Range	31	6	2002	Not in Range		
13	Valid Range	14	7	2000	15/7/2000		
14	Valid Range	14	7	1996	15/7/1996		
15	Valid Range	14	7	2002	15/7/2002		
16	Valid Input	29	7	1996	30/7/1996		
17	Valid Input	29	7	2000	30/7/2000		
18	Valid Input	29	7	2002	30/7/2002		
19	Valid Input	30	7	2000	31/7/2000		
20	Valid Input	30	7	1996	31/7/1996		

21	Valid Input	30	7	2002	31/7/2002		
22	Valid Input	31	7	1996	1/8/1996		
23	Valid Input	31	7	2000	1/8/2000		
24	Valid Input	31	7	2002	1/8/2002		
25	Valid Input	29	2	1996	1/3/1996		
26	Invalid Input	29	2	2000	Not in range		
27	Invalid Input	29	2	2002	Not in range		
28	Invalid Input	30	2	1996	Not in range		
29	Invalid Input	30	2	2000	Not in range		
30	Invalid Input	30	2	2002	Not in range		
31	Invalid Input	31	2	1996	Not in range		
32	Invalid Input	31	2	2000	Not in range		
33	Invalid Input	31	2	2002	Not in range		
34	Valid Input	14	2	1996	15/2/1996		
35	Valid Input	14	2	2000	15/2/2000		
36	Valid Input	14	2	2002	15/2/2002		

Weak Normal:

TC Id	Test Case Description	Input Data			Expected Output	Actual Output	Status
		DD	MM	YYYY			
1	Valid Condition	14	6	2000	15/6/2000		
2	Valid Input	29	7	1996	30/7/1996		
3	Invalid Date	30	2	2002	not in range		
4	Invalid date	31	6	2000	not in range		

6.6 EXECUTIONS

Execute the program and test the test cases in Table-1 against program and complete the table with for Actual output column and Status column

Test Report:

1. No of TC‟s Executed:
2. No of Defects Raised:
3. No of TC‟s Pass:
4. No of TC‟s Failed:

6.7 SNAPSHOTS:

```
File  Edit  View  Search  Terminal  Help
user@user-Vostro-2520:~/Testing$ cc nextDate6.c
user@user-Vostro-2520:~/Testing$ ./a.out

Enter the today's date in the form of dd mm yyyy
14 06 2000
Given date is : 14:6:2000
Next day is : 15:6:2000
user@user-Vostro-2520:~/Testing$ cc nextDate6.c
user@user-Vostro-2520:~/Testing$ ./a.out

Enter the today's date in the form of dd mm yyyy
29 07 1996
Given date is : 29:7:1996
Next day is : 30:7:1996
user@user-Vostro-2520:~/Testing$ cc nextDate6.c
user@user-Vostro-2520:~/Testing$ ./a.out

Enter the today's date in the form of dd mm yyyy
31 06 2000
Value of day, not in the range day<=30
Enter the today's date in the form of dd mm yyyy
29 07 2002
Given date is : 29:7:2002
Next day is : 30:7:2002
user@user-Vostro-2520:~/Testing$
```

EXPERIMENT No. 07: Triangle Problem-Decision Table Approach

7.1 OBJECTIVES:

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. Derive test cases for your program based on decision-table approach, execute the test cases and discuss the results.

7.2 REQUIREMENT SPECIFICATION:

R1. The system should accept 3 positive integer numbers (a, b, c) which represents 3 sides of
the triangle. Based on the input it should determine if a triangle can be formed or not.
R2. If the requirement R1 is satisfied then the system should determine the type of the triangle, which can be

- Equilateral (i.e. all the three sides are equal)
- Isosceles (i.e Two sides are equal)
- Scalene (i.e All the three sides are unequal)

else suitable error message should be displayed. Here we assume that user gives three positive integer numbers as input.

7.3 DESIGN:

Form the given requirements we can draw the following conditions:
C1: a<b+c?
C2: b<a+c?
C3: c<a+b?
C4: a=b?
C5: a=c?
C6: b=c?
According to the property of the triangle, if any one of the three conditions C1, C2 and C3 are not satisfied then triangle cannot be constructed. So only when C1, C2 and C3 are true the triangle can be formed, then depending on conditions C4, C5 and C6 we can decide what type of triangle will be formed. (i.e requirement R2).

ALGORITHM:

Step 1: Input a, b & c i.e three integer values which represent three sides of the triangle.
Step 2: if (a < (b + c)) and (b < (a + c)) and (c < (a + b) then do
step 3
else
print not a triangle. do step 6.

Step 3: if (a=b) and (b=c) then
Print triangle formed is equilateral. do step 6.
Step 4: if (a ≠ b) and (a ≠ c) and (b ≠ c) then
Print triangle formed is scalene. do step 6.
Step 5: Print triangle formed is Isosceles.
Step 6: stop

7.4 PROGRAM CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
        int a,b,c;
        printf("Enter the three sides of traingle");
        scanf("%d%d%d",&a,&b,&c);
        if((a<1||a>10)||(b<1||b>10)||(c<1||c>10))
        {
                printf("Out of range values");
                exit(0);
        }
        if((a<b+c)&&(b<a+c)&&(c<a+b))
        {
                if((a==b)&&(b==c))
                printf("Equilateral traingle");
                else if((a!=b)&&(b!=c)&&(a!=c))
                printf("Scalene traingle");
                else
                printf("Isosceles traingle");
        }
        else
        printf("Traingle cannot be formed");
        return 0;
}
```

7.5 TESTCASES:

Technique Used: Decision Table Approach

Decision Table-Based Testing has been around since the early 1960‟s; it is used to depict complex logical relationships between input data. A Decision Table is the method used to build a complete set of test cases without using the internal structure of the program in question. In order to create test cases we use a table to contain the input and output values of a program.

The decision table is as given below:

Conditions	**Condition Entries (Rules)**										
	R1	R2	R3	R4	R5	R 6	R 7	R 8	R 9	R10	R11
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?	--	--	--	F	T	T	T	F	F	F	T
C5: a=c?	--	--	--	T	F	T	F	T	F	F	T
C6: b=c?	--	--	--	T	T	F	F	F	T	F	T
Actions	**Action Entries**										
a1:Not a Triangle	X	X	X								
a2: Scalene										X	
a3: Isosceles							X	X	X		
a4: Equilateral											X
a5: Impossible				X	X	X					

The "--" symbol in the table indicates don't care values. The table shows the six conditions and 5 actions. All the conditions in the decision table are binary; hence, it is called as "Limited Entry decision table". Each column of the decision table represents a test case. That is,

The table is read as follows:

- Action: Not a Triangle

  1. When condition C1 is false we can say that with the given a, b and c values, it's Not a triangle.

  2. Similarly condition C2 and C3, if any one of them are false, we can say that with the given a, b and c values it's Not a triangle.

- Action: Impossible

  3. When conditions C1, C2, C3 are true and two conditions among C4, C5, C6 is true, there is no chance of one conditions among C4, C5, C6 failing. So we can neglect these rules.

  Example: if condition C4: a=b is true and C5: a=c is true

  Then it is impossible, that condition C6: b=c will fail, so the action is Impossible.

- Action: Isosceles

  4. When conditions C1, C2, C3 are true and any one condition among C4, C5 and C6 is true with remaining two conditions false then action is Isosceles triangle.

  Example: If condition C4: a=b is true and C5: a=c and C6: b=c are false, it means two sides are equal. So the action will be Isosceles triangle.

- Action: Equilateral

  5. When conditions C1, C2, C3 are true and also conditions C4, C5 and C6 are true then, the action is Equilateral triangle.

- Action: Scalene

  6. When conditions C1, C2, C3 are true and conditions C4, C5 and C6 are false i.e sides a, b and c are different, then action is Scalene triangle.

  Number of Test Cases = Number of Rules.

Using the decision table we obtain 11 functional test cases: 3 impossible cases, 3 ways of failing the triangle property, 1 way to get an equilateral triangle, 1 way to get a scalene triangle, and 3 ways to get an isosceles triangle.

Deriving test cases using Decision Table Approach:
Test Cases:

TC ID	Test Case Description	A	B	C	Expected Output	Actual Output	Status
1	Testing for Rule 1	6	4	1	Traingle cannot be formed		
2	Testing for Rule 2	1	5	3	Traingle cannot be formed		
3	Testing for Rule 3	1	2	4	Traingle cannot be formed		
4	Testing for Rule 7	2	2	3	Isosceles		
5	Testing for Rule 8	2	3	2	Isosceles		
6	Testing for Rule 9	3	2	2	Isosceles		
7	Testing for Rule 10	3	4	5	Scalene		
8	Testing for Rule 11	5	5	5	Equilateral		

7.6 EXECUTION

Execute the program against the designed test cases and complete the table for Actual output column and status column.

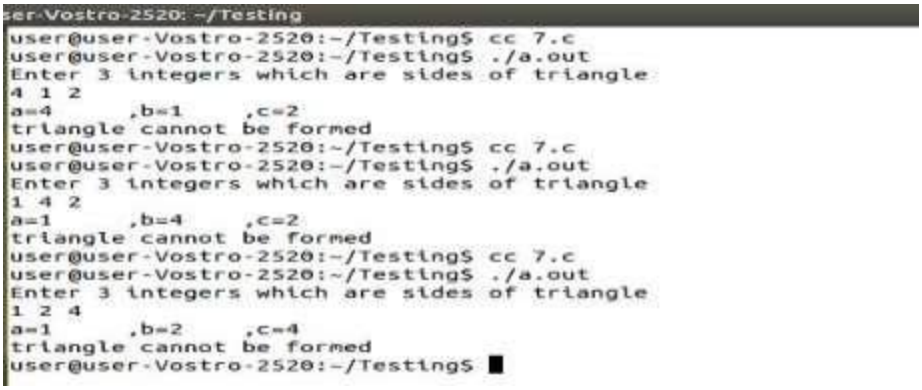
Test Report:

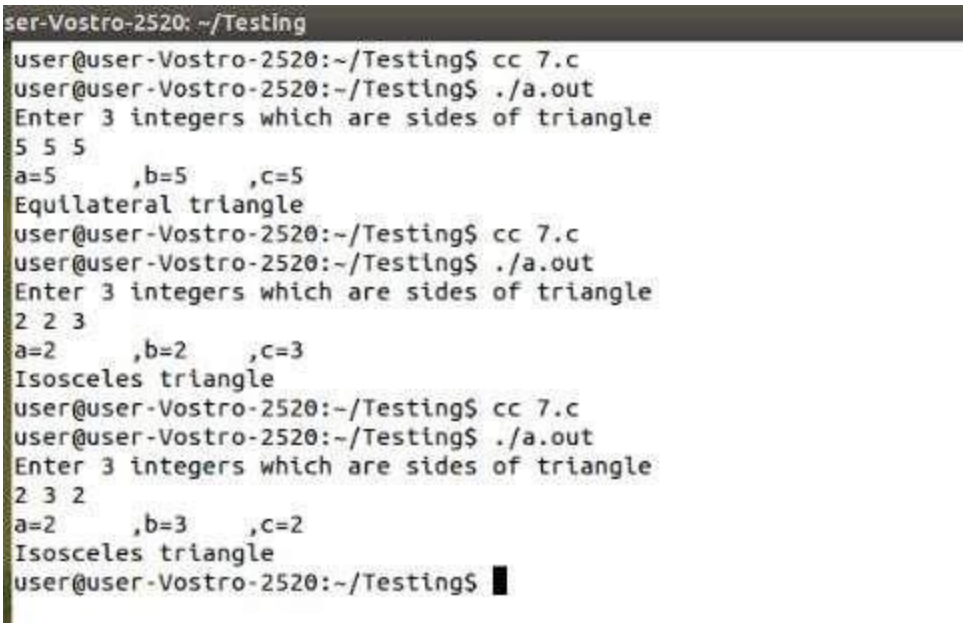
7.2.1 No of TC‟s Executed: 08
7.2.2 No of Defects Raised:
7.2.3 No of TC‟s Pass:
7.2.4 No of TC‟s Failed:

The decision table technique is indicated for applications characterised by any of the following:
Prominent if-then-else logic
Logical relationships among input variables
Calculations involving subsets of the input variables
Cause-and-effect relationship between inputs and outputs
The decision table-based testing works well for triangle problem because a lot of decision making i.e if-then-else logic takes place.

7.7 SNAPSHOTS:

1. Output screen of Triangle cannot be formed





7.8 PRE-EXPERIMENT & POST-EXPERIMENT QUESTIONS

1. Explain the decision table approach
2. What you mean by don't care values?
3. How many test cases in the decision table approach?
4. What are the differences between boundary value analysis and decision table approach?