SAMBHRAM
INSTITUTE OF TECHNOLOGY
Bengaluru - 560 097.

PARTICULARS OF THE EXPERIMENTS PERFORMED
CONTENTS

Sl. No.	DATE	EXPERIMENTS	PAGE NOS.	MARKS OBTAINED				SIGN. OF STAFF
				WRITEUP	CONDUCTION	VIVA	TOTAL	
01	10/2/18	TRIANGLE PROBLEM BASED ON BVA (4)	01	2	2	1	5	
02	16/2/18	COMMISSION PROBLEM BASED ON BVA (1)	04	2	2	1	5	
03	23/2/18	NEXT DATE PROBLEM BASED ON BVA	07	2	2	1	5	
04	2/3/18	TRIANGLE PROBLEM BASED ON EQUIVALENCE CLASS PARTITIONING (4)	10	2	2	1	5	
05	9/3/18	COMMISSION PROBLEM BASED ON EQUIVALENCE CLASS TESTING	13	2	2	1	5	
06	9/3/18	NEXT DATE PROBLEM BASED ON EQUIVALENCE CLASS TESTING. (2)	16	2	2	1	5	
07.	6/4/18	TRIANGLE PROBLEM BASED ON DECISION-TABLE (4)	19	2	2	1	5	
08	6/4/18	COMMISSION PROBLEM BASED ON DECISION-TABLE (3)	22	2	2	1	5	

SAMBHRAM
INSTITUTE OF TECHNOLOGY
Bengaluru - 560 097.

PARTICULARS OF THE EXPERIMENTS PERFORMED
CONTENTS

Sl. No.	DATE	EXPERIMENTS	PAGE NOS.	MARKS OBTAINED				SIGN. OF STAFF
				WRITEUP	CONDUCTION	VIVA	TOTAL	
09.	20/4/18	COMMISSION PROBLEM BASED ON DATA FLOW TESTING (3)	25	2	2	1	5	
10.	20/4/18	BINARY SEARCH BASED ON BASIS PATH TESTING (b)	29	2	2	1	5	
11.	27/4/18	QUICKSORT ALGORITHM BASED ON BASIS PATH TESTING.	33	2	2	1	5	
12.	27/4/18	ABSOLUTE LETTER GRADING PROCEDURE BASED ON BASIS PATH TESTING.	36	2	2	1	5	
		FINAL LAB INTERNAL TEST MARKS						
TOTAL LAB INTERNAL MARKS			MARKS ALLOTTED	20	MARKS OBTAINED	20		

25/4/18

ALGORITHM

Step1 : Input a, b, & c ie, three integer values which represents three sides of the triangle.

Step2 : If (a<(b+c)) and (b<(a+c)) and (c<(a+b))
then do step 3
else
print not a triangle do step 6

Step3 : if (a==b) and (b==c) then
printf triangle formed is equilateral do step 6

Step4 : If (a≠b) and (a≠c) and (b≠c) then
print triangle formed is Scalene do step6

Step 5 : Print triangle formed is isosceles

Step6 : Stop

4

01. 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.

REQUIREMENTS

R1: The system should accept 3 positive integers (a, b, c) which represents 3 sides of the triangle.

R2: Based on the input we 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 formed.

- Equilateral (ie, all the 3 sides are equal)
- Isosceles (ie, 2 sides are equal)
- Scalene (ie, All the 3 sides are unequal)

R4: Upper limit for the size of any side is 10.

DESIGN

From the given requirements, we can draw the following conditions:

C1: $a < b+c$?

C2: $b < a+c$?

C3: $c < a+b$?

TESTING

Technique used: Boundary Value analysis.

TEST CASES.

TC ID	Test Case Description	Inputs a	b	c	Expected Output	Actual Output	Status
1.	BVA	5	5	1	Isosceles triangle	Isosceles triangle	Success
2.	BVA	5	5	2	Isosceles triangle	Isosceles triangle	Success
3.	BVA	5	5	5	Equilateral triangle	Equilateral triangle	Success
4.	BVA	5	5	9	Isosceles triangle	Isosceles triangle	Success
5.	BVA	5	5	10	Not a triangle	Triangle not formed	Success
6.	BVA	5	1	5	Isoseles triangle	Isosceles triangle	Success
7.	BVA	5	2	5	Isosceles triangle	Isosceles triangle	Success
8.	BVA	5	9	5	Isosceles triangle	Isosceles triangle	Success
9.	BVA	5	10	5	Triangle not formed	triangle not formed	Success
10.	BVA	1	5	5	Isosceles triangle	Isosceles triangle	Success
11.	BVA	2	5	5	Isosceles triangle	Isosceles triangle	Success
12.	BVA	9	5	5	Isosceles triangle	Isosceles triangle	Success
13.	BVA	10	5	5	Triangle not formed	Triangle not formed	Success

c4 : a=b?
c5 : b=c?
c6 : c=a?

PROGRAM CODE

```
#include<stdio.h>
int main()
{
 int  a,b,c;
printf("enter the sides of triangle");
Scanf("%d%d%d", &a, &b, &c);
if((a<=0)||(a>10))
printf("a is out of range");
if((b<=0)||(b>10))
printf("b is out of range");
if((c<=0)||(c>10))
printf("c is out of range");
if((a<b+c)&&(b<a+c)&&(c<a+b))
{
if((a==b)&&(b==c)&&(c==a))
printf("equilateral triangle");
else if((a!=b)&&(b!=c)&&(a!=c))
printf("Scalene triangle");
else
{
printf("isosceles triangle");
return 0;
}
```

WORST CASES FOR BOUNDARY VALUE ANALYSIS

TC ID	Test case Description	Inputs a	b	c	Expected Output	Actual output	Status
1.	WCBVA	1	1	1	Equilateral	Equilateral triangle	Success
2.	WCBVA	1	1	2	Not a triangle	Not a triangle	Success
3.	WCBVA	1	1	5	Not a triangle	Not a triangle	Success
4.	WCBVA	1	1	9	Not a triangle	Not a triangle	Success
5.	WCBVA	1	1	10	Not a triangle	Not a triangle	Success
6.	WCBVA	1	2	1	Not a triangle	Not a triangle	Success
7.	WCBVA	1	2	2	Not a triangle	Not a triangle	Success
8.	WCBVA	1	2	5	Isosceles	Isosceles	Success
9.	WCBVA	1	2	9	Not a triangle	Not a triangle	Success
10.	WCBVA	1	2	10	Not a triangle	Not a triangle	Success
11.	WCBVA	1	5	1	Not a triangle	Not a triangle	Success
12.	WCBVA	1	5	2	Not a triangle	Not a triangle	Success
13.	WCBVA	1	5	5	Isosceles	Isosceles	Success
14.	WCBVA	1	5	9	Not a triangle	Not a triangle	Success
15.	WCBVA	1	5	10	Not a triangle	Not a triangle	Success
16.	WCBVA	1	9	1	Not a triangle	Not a triangle	Success
17.	WCBVA	1	9	2	Not a triangle	Not a triangle	Success
18.	WCBVA	1	9	5	Not a triangle	Not a triangle	Success
19.	WCBVA	1	9	9	Isosceles	Isosceles	Success
20.	WCBVA	1	9	10	Not a triangle	Not a triangle	Success
21.	WCBVA	1	10	1	Not a triangle	Not a triangle	Success
22.	WCBVA	1	10	2	Not a triangle	Not a triangle	Success
23.	WCBVA	1	10	5	Not a triangle	Not a triangle	Success
24	WCBVA	1	10	9	Not a triangle	Not a triangle	Success
25.	WCBVA	1	10	10	Isosceles	Isosceles	Success

```
}
else
printf(" Not a taiangle : " );
return 0;
}
```

TEST REPORT

No of test cases : 38
No of defects raised : 0
No of test cases passed : 38
No of test cases failed : 0

ALGORITHM

Step1 : Define lock price = 45.0, stock price = 30.0, Barrel price = 25.0

Step2 : Input locks

Step3 : while (locks != -1) input device uses = -1 to indicate the 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 step8 else goto step9.

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 + 0.15 * (sales - 1000.0)

Step11 : Output ("Commission is $", Commission)

Step12 : Exit

TESTING:

Technique used: Boundary Value analysis

16/2/18 (1)

SAMBHRAM INSTITUTE OF TECHNOLOGY 04

02 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.

REQUIREMENTS

R1: The system should read the number of locks, stocks and barrels sold in a month
(i.e, $1 \leq$ locks ≤ 70)
(ie, $1 \leq$ stocks ≤ 80)
(ie, $1 \leq$ barrels ≤ 90)

R2: If R1 is satisfied the system should compute the salesperson and commission depending on the total number of stocks, locks and barrels sold. Else it should display error message. Following is the percentage of commission for the sales done.

10%. Sales upto (and including) \$1000

15%. Sales on next 800\$

20%. on any sales in excess of 1800\$

Also the system should compute the total dollar sales. The system should output salesperson's total dollar sales and his commissions.

DESIGN

C1: $1 \leq$ Locks ≤ 70? Locks = -1 (occurs if locks = -1 is used to control input iteration)

C2: $1 \leq$ stocks ≤ 80?

C3: $1 \leq$ barrels ≤ 90?

C4: Sales ≥ 1800?

TEST CASES USING BOUNDARY VALUE ANALYSIS OF WORST CASE

TC ID	TC Desciption	Input			Expected output		Actual output		Status
		Locks	stocks	barrels	sales	commission	sales	commission	
1.	WCBVA	1	1	1	100	10	100	10	Success
2.	WCBVA	1	1	2	125	12.5	125	12.5	Success
3.	WCBVA	1	2	1	130	13	130	13	Success
4.	WCBVA	2	1	1	145	14.5	145	14.5	Success
5.	WCBVA	5	5	5	500	50	500	50	Success
6.	WCBVA	10	10	9	975	97.5	975	97.5	Success
7.	WCBVA	10	9	10	970	97	970	97	Success
8.	WCBVA	9	10	10	955	95.5	955	95.5	Success
9.	WCBVA	10	10	10	1000	100	1000	100	Success
10.	WCBVA	10	10	11	1025	103.75	1025	103.75	Success
11.	WCBVA	10	11	10	1030	104.5	1030	104.5	Success
12.	WCBVA	11	10	10	1045	106.75	1045	106.75	Success
13.	WCBVA	14	14	14	1400	160	1400	160	Success
14.	WCBVA	18	18	17	1775	216.25	1775	216.25	Success
15.	WCBVA	18	17	18	1770	215.5	1770	215.5	Success
16.	WCBVA	17	18	18	1755	213.25	1755	213.25	Success
17.	WCBVA	18	18	18	1800	220	1800	220	Success
18.	WCBVA	18	18	19	1825	225	1825	225	Success
19.	WCBVA	18	19	18	1830	226	1830	226	Success
20	WCBVA	19	18	18	1845	229	1845	229	Success
21.	WCBVA	48	48	48	4800	820	4800	820	Success
22.	WCBVA	70	80	89	7775	1415	7775	1415	Success
23.	WCBVA	70	79	90	7770	1414	7770	1414	Success
24.	WCBVA	69	80	90	7755	1411	7755	1411	Success
25.	WCBVA	70	80	90	7800	1420	7800	1420	Success

C5 : Sales > 1000?
C6 : Sales < 1000?

PROGRAM CODE

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
 int locks, stocks, barrels, t-sales;
float commission;
Printf("enter the total number of locks");
Scanf("%d", &locks);
Printf("enter the total number of stocks");
Scanf("%d", &stocks);
Printf("enter the total number of barrels");
Scanf("%d", &barrels);
if((locks<=0)||(locks>70)||(stocks<=0)||(stocks>80)||
 (barrels<=0)||(barrels>90))
{
    if((locks<=0)||(locks>70))
  {
      Printf(" locks out of range\n");
  }
   if((stocks<=0)||(stocks>80))
  {
      Printf("stocks out of range");
  }
   if((barrels<=0)||(barrels>90))
  {
```

Test report

No of test cases = 25
No of defect raised = 0
No of test cases passed = 25
No of test cases failed = 0

```
Printf(" barrels out of range");
}
  exit(0);
}
t_sales = (locks * 45) + (stocks * 30) + ( barrels * 25);
    if( t_sales <= 1000)
  {
    Commission = 0.10 * t_sales;
  }
 elseif ( t_sales <1800)
{
   Commission = 0.10 * 1000;
   Commission = Commission +( 0.15 * ( t_sales -1000));
}
  else
{
   Commission = 0.10 *1000;
   Commission = Commission + (0.15 * 800);
   Commission = Commission +(0.20 * (t_sales -1800) );
}
Printf("the total sales is %d and the commission
     %f", t_sales, Commission );
 return;
}
```

ALGORITHM

Step1 : Input date is 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 : Tomorrow day = DD + 1 goto step18
Step5 : Tomorrow day = 1;
Tomorrow month = 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 : Tomorrow day = 1,
Tomorrow month = 1,
Tomorrow year = yyyy + 1;
goto step18
Step11 : If mm is 2
Step12 : If DD < 28 do step4 else step13
Step13 : If DD = 28 and YYYY is a leap do step14
else step15
Step14 : Tomorrow day = 29 goto step18
Step15 : Tomorrow day = 1,
Tomorrow month = 3, goto step18
Step16 : If DD = 29 then do step15 else step17
Step17 : Output ("Cannot have feb", DD), step19
step18 : Output (tomorrowday, tomorrow month,
tomorrow year);
Step19 : exit.

23/2/18

03. Design, develop, code and run the program in any suitable language to implement the next date function. Analyze it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results.

REQUIREMENT

R1: The System should accept 3 integers DD MM YYYY for day, month and year and return the next date as output.

R2: It reads Current date as input.

R3: The Constraints are as given in the design and each input can be for a normal year, leap year or a century year.

R4: Month can be having 30 days, 31 days, 29 or 28 days.

DESIGN

From the given requirements the following conditions can be obtained

C1 : $1 \leq$ month ≤ 12

C2 : $1 \leq$ days ≤ 31

C3 : $1812 \leq$ year ≤ 2012

BOUNDARY VALUE ANALYSIS FOR NEXT DATE FUNCTION.

TC ID	Test Case Description	Inputs			Expected output	Actual Output	Status
		d	m	Y			
01.	BVA	15	6	1812	16:6:1812	16:6:1812	Success
02.	BVA	15	6	1813	16:6:1813	16:6:1813	Success
03.	BVA	15	6	1912	16:6:1912	16:6:1912	Success
04.	BVA	15	6	2011	16:6:2011	16:6:2011	Success
05.	BVA	15	6	2012	16:6:2012	16:6:2012	Success
06.	BVA	15	1	1912	16:1:1912	16:1:1912	Success
07.	BVA	15	2	1912	16:2:1912	16:2:1912	Success
08.	BVA	15	11	1912	16:11:1912	16:11:1912	Success
09.	BVA	15	12	1912	16:12:1912	16:12:1912	Success
10.	BVA	1	6	1912	2:6:1912	2:6:1912	Success
11.	BVA	2	6	1912	3:6:1912	3:6:1912	Success
12.	BVA	29	6	1912	30:6:1912	30:6:1912	Success
13.	BVA	30	6	1912	1:7:1912	1:7:1912	Success.

PROGRAM CODE

```
#include <stdio.h>
main()
{
int month[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
                 31};
int d, m, y, nd, nm, ny, ndays;
Printf("enter the month, day, year \n");
Scanf("%d %d %d", &m, &d, &y);
ndays = month[m-1];
if(m==2)
{
   if (y%100==0)
{
   if (y%400==0)
   ndays = 29;
}
else
   if (y%4==0)
   ndays = 29;
}
   if((y<1812)||(y>2012))
{
   printf("invalid year");
   return 0;
}
   if((m<1)||(m>12))
{
   printf("invalid month");
```

WORST CASE ANALYSIS TEST CASES FOR NEXT DATE FUNCTION

TC ID	TC Description	INPUTS			Expected output	Actual output	Status
		DD	MM	YYYY			
01.	WCA	1	1	1812	2:1:1812	2:1:1812	Success
02,	WCA	1	1	1813	2:1:1813	2:1:1813	Success
03.	WCA	1	1	1912	2:1:1912	2:1:1912	Success
04.	WCA	1	1	2011	2:1:2011	2:1:2011	Success
05.	WCA	1	1	2012	2:1:2012	2:1:2012	Success
06.	WCA	1	2	1812	2:2:1812	2:2:1812	Success
07.	WCA	1	2	1813	2:2:1813	2:2:1813	Success
08.	WCA	1	2	1912	2:2:1912	2:2:1912	Success
09.	WCA	1	2	2011	2:2:2011	2:2:2011	Success
10.	WCA	1	2	2012	2:2:2012	2:2:2012	Success
11.	WCA	1	6	1812	2:6:1812	2:6:1812	Success
12.	WCA	1	6	1813	2:6:1813	2:6:1813	Success
13.	WCA	1	6	1912	2:6:1912	2:6:1912	Success
14.	WCA	1	6	2011	2:6:2011	2:6:2011	Success
15.	WCA	1	6	2012	2:6:2012	2:6:2012	Success
16.	WCA	1	11	1812	2:11:1812	2:11:1812	Success
17.	WCA	1	11	1813	2:11:1813	2:11:1813	Success
18.	WCA	1	11	1912	2:11:1912	2:11:1912	Success
19.	WCA	1	11	2011	2:11:2011	2:11:2011	Success
20.	WCA	1	11	2012	2:11:2012	2:11:2012	Success
21.	WCA	1	12	1812	2:12:1812	2:12:1812	Success
22.	WCA	1	12	1813	2:12:1813	2:12:1813	Success
23.	WCA	1	12	1912	2:12:1912	2:12:1912	Success
24,	WCA	1	12	2011	2:12:2011	2:12:2011	Success
25.	WCA	1	12	2012	2:12:2012	2:12:2012	Success.

```
    return 0;
}
    if((d<=0)||(d>ndays))
{
    Printf("Invalid day");
    return 0;
}
    nd = d+1;
    nm = m;
    ny = y;
    if (nd > ndays)
{
    nd = 1;
    nm++;
}
    if (nm>12)
{
    nm = 1;
    ny++;
}
Printf("\n given date is %d:%d:%d \n", m,d,y);
Printf(" next day date is %d:%d:%d ~~\n~~", nm,nd,ny
}
```

TEST REPORT

No of test cases : 38
No of defect raised : 0
No of testcases passed : 38
No of testcases failed : 0

ALGORITHM

Step1 : Input a, b and c ie, three integer values which represent three sides of the triangle.

Step2 : If (a<(b+c)) and (b<(a+c)) and (c<(a+b)) then do step3 else
print not a triangle do step 6

Step3 : If (a=b) and (b=c) then
print triangle formed is equilateral do step 6.

Step4 : If (a≠b) and (a≠c) and (b≠c) then
print triangle formed is scalene do step 6

Step5 : Print triangle formed Isosceles.

Step6 : Stop.

2/3/18

04. 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.

REQUIREMENTS

R1: The system should accept 3 numbers (a,b,c) which represents 3 sides of the triangle.
R2: Based on 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 triangle :
1) Equilateral (ie, All 3 sides are equal)
2) Isosceles (ie, 2 sides are equal)
3) Scalene (ie, All 3 sides are unequal)
R4: Upper limit for the size of any side is 10.

DESIGN

From the given requirements we can draw the following conditions :
c1 : a<b+c ?
c2 : b<a+c ?
c3 : c<a+b ?
c4 : a=b ?

WEAK NORMAL EQUIVALENCE CLASS TEST CASES FOR TRIANGLE PROBLEM

TC ID	Test Case Description	Input a	b	c	Expected output	Actual output	Status
01.	WN1	5	5	5	Equilateral	Equilateral	Success
02.	WN2	2	2	3	Isosceles	Isosceles	Success
03.	WN3	3	4	5	Scalene	Scalene	Success
04.	WN4	4	1	2	Triangle cannot be formed	Triangle Cann ot be formed	Success

WEAK ROBUST EQUIVALENCE CLASS TEST CASES FOR TRIANGLE PROBLEM

TC ID	Test Case Description	Input a	b	c	Expected output	Actual output	Status
01.	WR1	-1	5	5	a is out of range	a is out of range	Success
02.	WR2	5	-1	5	b is out of range	b is out of range	Success
03.	WR3	5	5	-1	c is out of range	c is out of range	Success
04.	WR4	11	5	5	a is out of range	a is out of range	Success
05.	WR5	5	11	5	b is out of range	b is out of range	Success
06.	WR6	5	5	11	c is out of range	c is out of range	Success

c5 : b=c ?

c6 : a=c ?

PROGRAM CODE

```
#include <stdio.h>
int main()
{
  int a,b,c;
  Paint("entre the sides of triangle.");
  Scanf("%d%d%d",&a,&b,&c);
  if((a<=0)||(a>10))
  Printf("a is out of range");
  if((b<=0)||(b>10))
  Printf("b is out of range");
  if((c<=0)||(c>10))
  Printf("c is out of range");
  if((a<b+c)&&(b<a+c)&&(c<a+b))
  {
   if((a==b)&&(b==c)&&(c==a))
   Printf("equilateral triangle");
  elseif((a!=b)&&(b!=c)&&(a!=c))
  Printf("Scalene triangle");
  else
  {
   Printf("isosceles triangle");
   return 0;
  }
  }
```

STRONG ROBUST EQUIVALENCE CLASS TEST CASES FOR TRIANGLE PROBLEM.

TC ID	Test Case Description	Input a	b	c	Expected output	Actual Output	status
01.	SR1	-1	5	5	a is out of range	a is out of range	Success
02.	SR2	5	-1	5	b is out of range	b is out of range	Success
03.	SR3	5	5	-1	c is out of range	c is out of range	Success
04.	SR4	-1	-1	5	a,b is out of range	a,b is out of range	Success
05.	SR5	5	-1	-1	b,c is out of range	b,c is out of range	Success
06.	SR6	-1	5	-1	a,c is out of range	a,c is out of range	Success
07.	SR7	-1	-1	-1	a,b,c is out of range	a,b,c is out of range.	Success

TEST REPORT

No of test cases = 17
No of defect raised = 0
No of test cases passed = 17
No of test cases failed = 0

```
else
printf("triangle cannot be formed");
return 0;
}
```

ALGORITHM

STEP 1: Define dock price = 45.0, stock price = 30.0, barrel price = 25
STEP 2: Input locks
STEP 3: while (locks != 1) input devices uses -1 to indicate end of data goto step 12.
STEP 4: Input (stocks, barrels)
STEP 5: Compute locksales, stocksales, barrelsales and sales.
STEP 6: Output("Total Sales", sales)
STEP 7: If (sales > 1800.0) goto step8 else goto step9
STEP 8: Commission = 0.10 * 1000 ;
Commission = Commission + 0.15 * 800.0 ;
Commission = Commission + 0.20 * (sales - 1800.0)
STEP 9: If (sales > 1000.0) goto step10 else goto step11
STEP 10: Commission = 0.10 * 1000.0 ;
Commission = Commission + 0.15 * (sales - 1000.0)
STEP 11: Output("Commission is $", Commission)
STEP 12: Exit

TESTING:

Technique used: Equivalence class testing.

9/3/18

05. 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.

REQUIREMENT

R1: The System should read the number of locks, stocks and barrels in a month (i.e, $1 \leq locks \leq 70$)
(i.e, $1 \leq stocks \leq 80$)
(i.e, $1 \leq barrels \leq 90$)

R2: If R1 is satisfied the System should compute salesperson and commission depending on the total number of stocks and barrels sold else it should display error message. Following is the percentage of commission for the sales done.
10% on sales upto (and including) $1000
15% on next 800
20% on any sales in excess on $1800
Also the System should compute the total dollar sales. The System should output salesperson total dollar sales and his commission.

DESIGN

From the given requirements we can draw the following conditions C1: $1 \leq locks \leq 70$?
C2: $1 \leq stocks \leq 80$?
C3: $1 \leq barrels \leq 90$?
C4: Sales > 1000 ?
C5: Sales ?

WEAK NORMAL EQUIVALENCE CLASS TEST CASES FOR COMMISSION PROBLEM

TC ID	Test Case Description	INPUTS			EXPECTED OUTPUT		ACTUAL OUTPUT		STATUS
		Locks	Stocks	Barrels	SALES	COMMISSION	SALES	COMMISSION	
01.	WN1	10	8	9	915	91.5	915	91.5	Success
02.	WN2	10	11	19	1255	138.25	1255	138.25	Success
03.	WN3	18	19	17	1805	221	1805	221	Success

WEAK ROBUST EQUIVALENCE CLASS TEST CASES FOR COMMISSION PROBLEM.

TC ID	Test Case Description	INPUT			EXPECTED OUTPUT		ACTUAL OUTPUT		STATUS
		LOCKS	STOCKS	BARREL	SALES	COMMISSION	SALES	COMMISSION	
01.	WR1	-1	40	45	Locks out of range.		Locks out of range.		Success
02.	WR2	35	-1	45	stocks out of range		stocks out of range.		Success
03.	WR3	35	40	-1	barrels out of range.		barrels out of range.		Success
04.	WR4	71	40	45	Locks out of range.		Locks out of range.		Success
05.	WR5	35	81	45	stocks out of range.		stocks out of range.		Success
06.	WR6	35	40	91	Barrels out of range.		Barrels out of range		Success

PROGRAM CODE

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int locks, stocks, barrels, t_sales;
float commission;
Printf("enter the total number of locks");
Scanf("%d", &locks);
Printf("enter the total number of stocks");
Scanf("%d", &stocks);
Printf("enter the total number of barrels");
Scanf("%d", &barrels);
if((locks<=0)||(locks>70)||(stocks<=0)||(stocks>80)||
  (barrels<=0)||(barrels>90))
{
  if((locks<=0)||(locks>70))
{
    Printf("locks out of range\n");
}
  if((stocks<=0)||(stocks>80))
{
    Printf("stocks out of range");
}
  if((barrels<=0)||(barrels>90))
{
    Printf("barrels out of range");
}
  exit(0);
```

STRONG ROBUST EQUIVALENCE CLASS TEST CASES FOR COMMISSION PROBLEM.

TC ID	Test Case Description	INPUTS			EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
		LOCKS	STOCKS	BARREL			
01.	SR1	-1	40	45	Locks out of range.	Locks out of range.	Success
02.	SR2	35	-1	45	Stocks out of range.	Stocks out of range.	Success
03.	SR3	35	40	-1	Barrels out of range.	Barrels out of range.	Success
04.	SR4	-1	-1	45	Locks, Stocks out of range.	Locks, Stocks out of range.	Success
05.	SR5	35	-1	-1	Stocks, barrels out of range.	Stocks, barrels out of range.	Success
06.	SR6	-1	40	-1	Locks, barrels out of range.	Locks, barrels out of range.	Success
07.	SR7	-1	-1	-1	Locks, Stocks, barrels out of range.	Locks, Stocks, barrels out of range.	Success

TEST REPORT:

No of test cases : 16
No of defect raised : 0
No of test cases passed : 16
No of test cases failed : 0

```
}
t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);
  if( t_sales <= 1000)
{
   Commission = 0.10 * t_sales;
}
elseif ( t-sales <1800)
{
   Commission = 0.10 * 1000;
  Commission = Commission + (0.15* (t_sales -1000));
}
else
{
  Commission = 0.10 * 1000;
  Commission = Commission + (0.15 * 800);
  Commission = Commission + (0.20 * (t_Sales -1800));
}
Printf("the total sales is %d in the Commission is
    %f", t_sales, Commission);
  return ;
}
```

ALGORITHM

STEP 1: Input date in format MM DD YYYY
STEP 2: If MM is 01, 03, 05, 08, 10 do step3 else step6
STEP 3: If DD<31 then do step4 else if DD=31 do step5 else output (invalid date)
STEP 4: Tomorrow day: DD+1 goto step18
STEP 5: Tomorrow day: 1;
Tomorrow month = month+1 goto step18
STEP 6: If mm is 04, 06, 09, 11 do step 7
STEP 7: If DD<30 then do step 4 else if DD=30 do step 5 else output (Invalid date)
STEP 8: If mm is 12
STEP 9: If DD<31 then step4 else step10
STEP 10: Tomorrow day=1,
Tomorrow month=1,
Tomorrow year = YYYY+1;
goto step 18
STEP 11: If mm is 2
STEP 12: If DD<28 do step 4 else step 13
STEP 13: If DD=28 and YYYY is a leap do step14 else step15
STEP 14: Tomorrow day=29 goto step 18
STEP 15: Tomorrow day=1,
Tomorrow month=3, goto step 18
STEP 16: If DD=29 then do step 4 else step 17
STEP 17: Output ("Invalid date", DD), step 19
STEP 18: Output (Tomorrow day, Tomorrow month, Tomorrow year);
STEP 19: END

9/8/18

SAMBHRAM INSTITUTE OF TECHNOLOGY

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

REQUIREMENTS

R1: The system must accept 3 integers DD MM YYYY for day, month and year and return the next date as output.

R2: It reads current date as input
C1: $1 \leq$ month ≤ 12
C2: $1 \leq$ Day ≤ 31
C3: $1812 \leq$ year ≤ 2012

R3: Each input can be for a normal year, a leap year & century year.

R4: Month can be having 30, 31, 28 & 29 days.

DESIGN

From a given requirement the following conditions can be obtained C1: $1 \leq$ month ≤ 12 ?
C2: $1 \leq$ Day ≤ 31 ?
C3: $1812 \leq$ year ≤ 2012 ?

WEAK NORMAL EQUIVALENCE CLASS TEST CASES FOR NEXT DATE PROBLEM.

TC ID	Test Case Description	INPUTS			EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
		mm	DD	YYYY			
01.	WN1	06	15	1912	6:16:1912	6:16:1912	Success

WEAK ROBUST EQUIVALENCE CLASS TEST CASES FOR NEXT DATE PROBLEM.

TC ID	Test Case Description	INPUTS			Expected output	Actual output	STATUS
		mm	DD	YYYY			
01.	WR1	06	15	1912	06:16:1912	06:16:1912	Success
02.	WR2	-1	15	1912	Invalid month	Invalid month	Success
03.	WR3	06	-1	1912	Invalid day	Invalid day	Success
04.	WR4	06	15	1811	Invalid year	Invalid year	Success
05.	WR5	13	15	1912	Invalid month	Invalid month	Success
06.	WR6	06	32	1912	Invalid day	Invalid day	Success
07.	WR7	06	15	2013	Invalid year	Invalid year	Success

PROGRAM CODE

```
#include <stdio.h>
main()
{
int month[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
int d,m,y,nd,nm,ny,ndays;
Printf("enter the month, day, year \n");
Scanf("%d %d %d", &m, &d, &y);
ndays = month[m-1];
if (m==2)
{
 if(y%100==0)
{
  if (y%400==0)
  ndays = 29;
}
else
  if(y%4==0)
  ndays = 29;
}
if((y<1812)||(y>2012))
{
  Printf("invalid year");
  return 0;
}
  if((m<1)||(m>12))
{
  Printf("invalid month");
  return 0;
```

STRONG ROBUST EQUIVALENCE CLASS TEST CASES FOR NEXT DATE PROBLEM.

TC ID	Test Case Description	INPUTS MM	DD	YYYY	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
01.	SR1	-1	15	1912	Invalid month	Invalid month	Success
02.	SR2	6	-1	1912	Invalid Day	Invalid Day	Success
03.	SR3	6	15	1811	Invalid year	Invalid year	Success
04.	SR4	-1	-1	1912	Invalid month, Invalid Day.	Invalid month Invalid Day	Success Success
05.	SR5	-1	15	1811	Invalid month Invalid year	Invalid month Invalid year	Success Success
06.	SR6	6	-1	1811	Invalid Day Invalid year	Invalid Day Invalid year	Success
07.	SR7	-1	-1	1811	Invalid month Invalid Day Invalid year	Invalid month Invalid Day Invalid year	Success

TEST REPORT

No of test Cases : 15
No of defect raised : 0
No of test Cases passed : 15
No of test Cases failed : 0


```
}
if((d<=0)||(d>ndays))
{
  Printf("invalid day");
  return 0;
}
nd = d+1;
nm = m;
ny = y;
if(nd > ndays)
{
   nd=1;
   nm++;
}
   if(nm>12)
{
   nm=1;
   ny++;
}
Printf(" given date is %d:%d:%d \n", m,d,y);
Printf(" next date is %d:%d:%d \n", nm,nd,ny);
}
```

ALGORITHM.

Step1: Input a, b, & c i,e three integer values which represents three sides of the triangle.

S1

Step2: if (a<(b+c)) and (b<(a+c)) and (c<(a+b))
then do step3
else
print not a triangle do step6.

Step3: If (a==b) and (b==c) then
print triangle formed is equilateral do step6

Step4: If (a≠b) and (a≠c) and (b≠c) then
print triangle formed is Scalene do step6.

Step5: Print triangle formed is isosceles.

Step6: Stop.

Coding

Technique used: Decision-table based approach.

6/4/18

Q7. 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.

REQUIREMENTS

R1: The System should 3 positive integers (a, b, c) which represents 3 sides of the triangle.

R2: Based on the input we 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 formed.

- Equilateral (i.e, all the 3 sides are equal)
- Isosceles (i.e, 2 sides are equal)
- Scalene (i.e, All the 3 sides are unequal)

R4: Upper limit for the size of any side is 10.

DESIGN

From the given requirements, we can draw the following conditions:

C1: a<b+c ?

C2: b<a+c ?

C3: c<a+b ?

Decision table

Conditions	CONDITION ENTRIES (RULES)										
	R_1	R_2	R_3	R_4	R_5	R_6	R7	R8	R9	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
ACTION	ACTION ENTRIES										
a1: Not a Δle	X	X	X								
a2: Scalene										X	
a3: Isosceles							X	X	X		
a4: Equilateral											X
a5: Impossible				X	X	X					

C4: a=b?
C5: b=c?
C6: c=a?

PROGRAM CODE

```
#include <stdio.h>
int main()
{
  int a,b,c;
  Printf("enter the sides of triangle");
  Scanf("%d %d %d", &a, &b, &c);
  if ((a<=0)||(a>10))
  printf(" a is out of range");
  if((b<=0)||(b>10))
  printf("b is out of range");
  if((c<=0)||(c>10))
  printf(" C is out of range.");
  if((a<b+c)&&(b<a+c)&&(c<a+b))
  {
  if((a==b)&&(b==c)&&(c==a))
  printf(" equilateral triangle");
  else if((a!=b)&&(b!=c)&&(a!=c))
  printf("Scalene triangle");
  else
  {
  printf("isosceles triangle");
  return 0;
  }
  }
```

Test case ID	Test Case Description	Inputs a	b	c	Expected output	Actual output	Status
01.	Testing for req R1	4	1	2	Not a triangle	Not a triangle	Success
02.	Testing for req R1	1	4	2	Not a triangle	Not a triangle	Success
03.	Testing for req R1	1	2	4	Not a triangle	Not a triangle	Success
04.	Testing for req R2	5	5	5	Equilateral	Equilateral	Success
05.	Testing for req R2	2	2	3	Isosceles	Isosceles	Success
06.	Testing for req R2	2	3	2	Isosceles	Isosceles	Success
07.	Testing for req R2	3	2	2	Isosceles	Isosceles	Success
08.	Testing for req R2	3	4	5	Scalene	Scalene.	Success

```
else
printf(" not a triangle :");
return 0;
}
```

Test report
Number of test case's executed : 8
Number of defects raised : 0
Number of test cases passed : 8
Number of test cases failed : 0

4 1 2
1 4 2
1 2 4
5 5 5
2 2 3
2 3 2
3 2 2
3 4 5

ALGORITHM

Step1: Define lock price = 45.0, stock price = 30.0, barrel price = 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 Step8 else goto Step9
Step8: Commission = 0.10 * 1000;
Commission = Commission + 0.15 * 800.0
Commission = Commission + 0.20 * (Sales - 1800.0)
Step9: If (Sales > 1000.0) goto Step10 else goto Step11
Step10: Commission = 0.10 * 1000.0;
Commission = Commission + 0.15 * (Sales - 1000)
Step11: Output ("Commission is $", Commission)
Step12: Exit.

Testing:
Technique used: Decision-table based testing

6/4/18

08. 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.

REQUIREMENTS

R1: The System should read the number of locks, stocks and barrels sold in a month.
(i.e, 1 ≤ locks ≤ 70)
(i.e, 1 ≤ stocks ≤ 80)
(i.e, 1 ≤ barrels ≤ 90)

R2: If R1 is satisfied the System should Compute the salesperson and Commission depending on the total number of stocks and barrels sold.
Else it should display error message. Following is the percentage of Commission for the sales done.
10% Sales upto (and including) $1000
15% Sales on next 800$
20% on any sales in excess of 1800
Also the System should Compute the total dollar sales. The System should output salesperson's total dollar sales and his commissions.

DESIGN.

C1: 1 ≤ Locks ≤ 70 ? Locks = -1 (Occurs if locks = -1 is used to control input iteration).
C2: 1 ≤ Stocks ≤ 80 ?
C3: 1 ≤ barrels ≤ 90 ?
C4: Sales ≥ 1800

Decision table.

CONDITIONS	CONDITION ENTRIES (RULES) Rules1	Rules2	Rules3	Rules 4	Rule 5	Rule 6
C1: $1 \le$ locks ≤ 70 ?	F	T	T	T	T	T
C2: $1 \le$ stocks ≤ 80?	-	F	T	T	T	T
C3: $1 \le$ barrels ≤ 90?	-	-	F	T	T	T
C4: Sales ≥ 1800 ?	-	-	-	T	F	F
C5: Sales < 1800 ?	-	-	-	F	T	F
C6: Sales ≤ 1000 ?	-	-	-	F	F	T
ACTIONS	ACTION ENTRIES					
a1: com1 = 0.10 * sales						X
a2: com2 = com1 + 0.15 * (sales − 1000)					X	
a3: com3 = com2 + 0.20 * (sales − 1800)				X		
a4: out of range	X	X	X			

C5: Sales > 1000 ?

C6: Sales < 1000 ?

PROGRAM CODE

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int locks, stocks, barrels, t_sales;
  float commission;
  Printf("enter the total number of locks");
  Scanf("%d", &locks);
  Printf("enter the total number of stocks");
  Scanf("%d", &stocks);
  Printf("enter the total number of barrels");
  Scanf("%d", &barrels);
  if((locks<=0)||(locks>70)||(stocks<=0)||(stocks>80)||
     (barrels<=0)||(barrels>90))
  {
     if((locks<=0)||(locks>70))
     {
        printf("locks out of range\n");
     }
     if((stocks<=0)||(stocks>80))
     {
        printf("stocks out of range\n");
     }
     if((barrels<=0)||(barrels>90))
     {
```

Test cases.

Test case ID	Test case Description.	Locks	stocks	barrels	Expected output	Actual output	Status
01.	Test for req R1 (based C1)	-2	40	45	out of range	Out of range	Success
02.	Test for req R1 (based C1)	90	45	45	Out of range	Out of range	Success
03.	Test for req R1 (based C2)	35	-3	45	Out of range	Out of range	Success
04.	Test for req R1 (based C2)	35	100	45	Out of range	Out of range	Success
05.	Test for req R1 (based C3)	35	40	-10	Out of range	Out of range	Success
06.	Test for req R1 (based C3)	35	40	150	Out of range	Out of range	Success
07.	Test for req R2 (based C4)	25	25	25	500 a1:50	500 a1:50	Success
08.	Test for req R2 (based C5)	15	15	15	1500 a2:175	1500 a2:175	Success
09.	Test for req R2 (based C6)	25	25	25	2500 a3:360	2500 a3:360	Success


```
Printf("barrels out of range");
}
exit(0);
}
t_sales = (locks * 45) + (stocks * 30) + (barrels * 25);
if( t_sales <=1000)
{
   Commission = 0.10 * t_sales;
}
else if (t_sales <1800)
{
   Commission = 0.10 *1000;
   Commission = Commission + (0.15 *( t_sales -1000));
}
else
{
  Commission = 0.10 *1000;
  Commission = Commission +(0.15 *800);
  Commission = Commission + (0.20 *( t_sales - 1800 ));
}
printf("the total sales is %d in the commission is %f",
      t_sales, Commission);
return;
}
```

Test report:

No of test case's executed : 9
No of defects raised : 0
No of test cases passed : 9
No of test cases failed : 0

Algorithm

Step1: Define lock price = 45.0, stock price = 30.0, barrel price = 25.0

Step2: Input locks.

Step3: while(locks != -1) input devices 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 step8 else goto step 9

Step8: Commission = 0.10 * 1000;
Commission = Commission + 0.15 * 800.0;
Commission = Commission + 0.20 * (sales - 1800.0)

Step9: If (sales > 1000.0) goto step10 else goto step11

Step10: Commission = 0.10 * 1000.0;
Commission = Commission + 0.15 * (sales - 1000.0)

Step11: Output ("Commission is $ ", Commission)

Step12: Exit

Testing:

Technique used: Data flow testing.

2019-18

SAMBHRAM INSTITUTE OF TECHNOLOGY

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

Requirements.

R1: The System should read the number of locks, stocks, and barrels sold in a month
(i.e, 1 ≤ locks ≤ 70)
(i.e, 1 ≤ stocks ≤ 80)
(i.e, 1 ≤ barrels ≤ 90)

R2: If R1 is satisfied, the System should Compute the sales person. and commission depending on the total number of stocks and barrels sold.
Else it should display error message. Following is the percentage of commission for the sales done.
10% Sales upto (and including) $1000
15% Sales on next 800$
20% on any Sales in excess of 1800
Also the System should compute the total dollar sales.
The System Should output Salesperson's total dollar sales and his commissions.

DESIGN:

C1: 1 ≤ locks ≤ 70? Locks = -1 (occurs if locks = -1 is used to control input iteration).

C2: 1 ≤ stocks ≤ 80?

C3: 1 ≤ barrels ≤ 90?

C4: Sales ≥ 1800?

Test Case Name : Data Flow Testing for Commission program

Precondition : Enter -1 for locks to exit from input loop.

Brief Description : Enter the locks, stocks and barrels > 0.

Define/Use nodes for Variables in the Commission problem.

Variable name	Defined at node	Used at Node.
lprice	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, 40

C5: Sales > 1000?

C6: Sales < 1000?

Program Code.

```
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;
13     printf("enter the numbers of locks and to exit the loop
           enter -1 for locks\n");
       Scanf("%d", &locks);
14     while( locks != -1) {
15     printf("enter the numbers of stocks and barrels \n");
       Scanf("%d %d", &stocks, &barrels);
16     tlocks = tlocks + locks;
17     tstocks = tstocks + stocks;
18     tbarrels = tbarrels + barrels;
19     printf("\n enter the numbers of locks and to exit the loop
           enter -1 for locks\n");
       Scanf("%d", &locks);
20 }
```

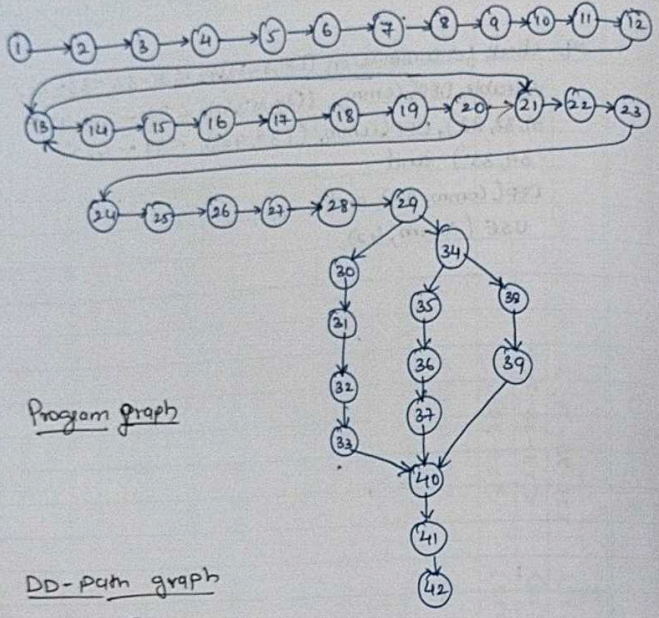
Selected Define / Use Paths for Commission problem.

Test case Id.	Description	Variables path (Beginning, End nodes)	Du paths.	Different classes?
01.	Check for lock price Variable DEF(lprice, 7) and USE(lprice, 24)	(7, 24)	<7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24>	Yes
02.	Check for stock price Variable DEF(sprice, 8) and USE(sprice, 25)	(8, 25)	<8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25>	Yes
03.	Check for barrel price Variable DEF(bprice, 9) and USE(bprice, 26)	(9, 26)	<9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26>	Yes.
04.	Check for total locks Variables DEF(tlocks, 10) and DEF(tlocks, 16) & usage node(USE(tlocks, 16); USE 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


```
21  printf("\ntotal locks = %d\n", 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  printf("the commission is = %f\n", comm);
41  return 0;
42  }
```

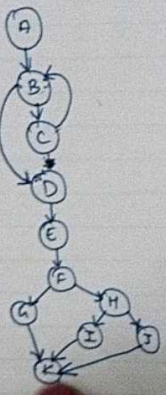
05.	check for total stocks Variable DEF(tstocks, 11) and DEF(tstocks, 17)) and 3 usage node(USE(tstocks, 17), USE(tstocks, 22), USE(tstocks, 25)	(11, 17)	<11-12-13-14-15-16-17>	Yes
		(11, 22)	<11-12-13-14-15-16-17-18-19-20-21-14-21>	NO
		(11, 25)	<11-12-13-14-15-16-17-18-19-20-21-14-21-23-24-25>	NO
		(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
06.	check for locks Variable (DEF(locks, 13), DEF(locks, 19) and USE(locks, 14), USE(locks, 16)	(13, 14)	<13-14>	Yes
		(13, 16)	<13-14-15-16>	Yes
		(19, 14)	<19-20-14>	Yes
		(19, 16)	<19-20-14-15-16>	Yes
07.	check for stocks Variables (DEF(stocks, 15) and USE(stocks, 17)	(15, 17)	<15-16-17>	Yes
08.	check for sales 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

09.	check for commission Variable DEF(comm, 31, 32, 33), DEF(comm, 34, 35) and DEF(comm, 39, 42) and USE(comm, 39) and USE(comm, 42)	(31, 32, 33), (42)	<31-32-33-42>	Yes
		((34, 35), (42)	<34-35-42>	Yes
		(39, 42))	<39-42>	Yes



Program graph

DD-path graph



Algorithm,

Step1 : Input value of 'n', Enter 'n' integer numbers in array int mid ;
Step2 : Initialize low=0, high=n-1
Step3 : until (low <= high) do
mid = (low + high)/2
if (a[mid] == key)
then do step5
else if (a[mid] > key)
then do
high = mid-1
else
low = mid+1
Step4 : Print unsuccessful Search do step6
Step5 : Print Succesful Search. Element found at position mid+1
Step6 : Stop

Testing :
Technique used: Basis path testing.

20/4/18
SAMBHRAM

10. 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.

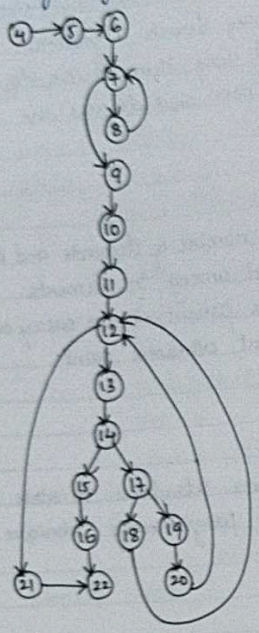
Requirement
R1: The System should accept 'n' number of elements and key element that is to be Searched among 'n' elements.
R2: Check if the key element is present in the array and display the position if present otherwise print unsuccessful Search.

Design:
We use integer array as a data structure to store 'n' number of elements. Iterative programming technique is used.

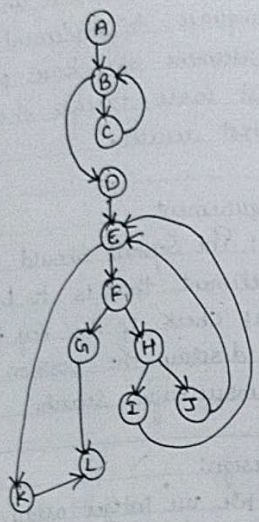
Program Code:

```
#include<stdio.h>
int main()
{
int a[30], i, n, element, low=0, high, mid;
printf("enter the value of n");
Scanf("%d", &n);
printf("enter array elements");
for(i=0; i<n; i++)
Scanf("%d", &a[i]);
```

Program graph.



DD-graph.

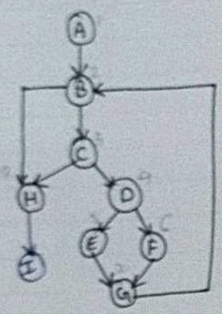


$V(G) = e - n + 2p$
$= 15 - 12 + 2(1)$
$= 3 + 2$
$V(G) = 5$

Test case ID	Feasible time	value of n	array elements	Expected output	Actual output	status.
1	P1: A-B-D-E-K-L	0	-	no element	no element	True
2	P2: A-B-C-B-D-E-F-G-L	4	{10, 20, 30, 40}	20, key found	key found	True
3	P3: A-B-C-B-D-E-F-H-I-E-F-G-L	3	{30, 40, 50}	50 found	key found	True
4	P4: A-B-C-B-D-E-F-H-I-E-F-K-L	5	{10, 20, 30, 40, 50}	60 element not found	element not found	True
5				element not found	element	True.

```
10  printf("enter the element to be searched");
11  scanf("%d", &element);
12  while (low<=high)
    {
13    mid = (low+high)/2;
14    if (element == a[mid])
      {
15    printf("key found");
16    return 0;
      }
17    else if (element > a[mid])
18    low = mid+1;
19    else
20    high = mid-1; }
21    printf(" element not found");
22.   return 0;
    }
```

DD-path graph.



Cyclomatic complexity:

$V(G) = e - n + 2p$

$= 11 - 9 + 2(1)$

$= 4$

Binary search using basis path testing

Paths	Inputs n	Keys	Expected output	Actual output	status
P1: A-B-C-H-I	{10,20,30,40}	10	element found in position = 1	element found in position = 1	True
P2: A-B-C-D-E-G-B	{10,20,30,40}	30	element found in position = 3	element found in position = 3	True
P3: A-B-C-D-F-G-B	{10,20,30,40}	40	element found in position = 4	element found in position = 4	True.
P4: A-B-H-I	{10,20,30,40}	60	element not found	element not found.	True.
	{0}	50	number of element should be greater than zero.	number of element should be greater than zero	True.

Page No.
Date

```
#include<stdio.h>
int binsrc(int x[], int low, int high, int key)
{
1   int mid;
2   while(low<=high)
    {
    mid=(low+high)/2;
3   if(x[mid]==key)
    return mid;
4   if(x[mid]<key)
5   low=mid+1;
6   else
7   high=mid-1;
8   }
9   return -1;
    }
    int main()
    {
    int a[20], key, i, n, succ;
1   printf("enter a value");
2   scanf("%d", &n);
3   if(n>0)
    {
4   printf("enter the elements in ascending order\n");
5   for(i=0; i<n; i++)
6   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 elements should be greater than zero");
return 0;
}
```

Algorithm

Step1 : Start.

Step2 : Input value of n (array size) and array elements a[i]

Step3 : quicksort

Algorithm.

Step1: choose the highest index value has pivot

Step2: Take two variable to point left and right of list.

Step3: left points to the lower index.

Step4: right points to the higher index.

Step5: while value is left is less than pivot move right

Step6: while value at right is greater than pivot move left.

Step7: If both step5 and step6 does not match swap left and right.

Step8: If left ≥ right then ~~pivot~~ point where they met is pivot.

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

Requirements:

R1: The System should accept 'n' number of elements and key element that is to be searched among 'n' elements.

R2: check if the key element is present in the array and display the position if present otherwise print unsuccessful search.

Design:

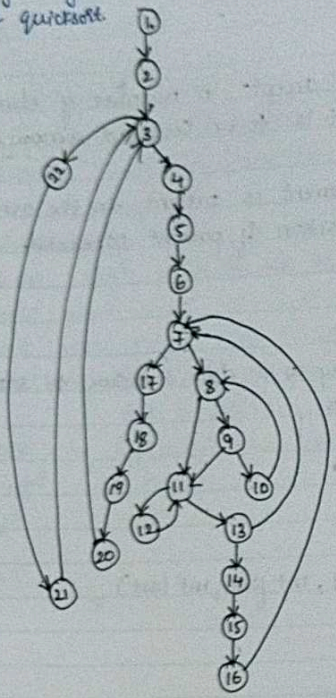
We use integer array as a data structure to store 'n' number of elements.

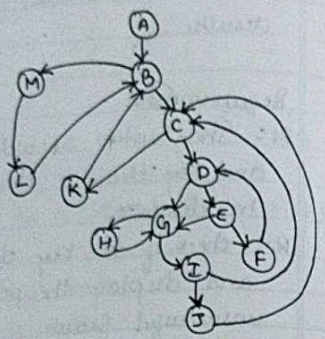
Program Code.

Pgm 11 Pg 1

```
#include<stdio.h>
1  void quicksort(int x[10], int first, int last)
2  {
     int temp, pivot, i, j;
3    if(first<last)
4    {
       pivot = first;
5      i = first;
6      j = last;
7      while(i<j)
```

Program graph for quicksort.





DD-path for quicksort.

Cyclomatic Complexity,

$V(G) = e - n + 2p$

$= 19 - 13 + 2$

$= 8$

```
8   {
    while (x[i] <= x[pivot] && i < last)
10  i++;
11  while (x[j] > x[pivot])
12  j--;
13  if (i<j)
14  {
15  tmp = x[i];
16  x[i] = x[j];
17  x[j] = tmp; }
18  tmp = x[pivot];
19  x[pivot] = x[j];
20  x[j] = tmp;
21  quicksort (x, first, j-1);
22  quicksort (x, j+1, last);
    }
    }
    int main()
    {
     int a[20], i, keys, n;
     printf ("enter the size of the array");
     scanf ("%d", &n);
     if (n>0)
     {
      printf ("enter the element 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");
      for (i=0; i<n; i++)
```

Basis paths.

P1 : A-B-M
P2 : A-B-C-K-B-M
P3 : A-B-C-K-B-M-L-B-M
P4 : A-B-C-D-G-I-C-K-B-M-L-B-M
P5 : A-B-C-D-G-I-J-C-K-B-M-L-B-M
P6 : A-B-C-D-E-F-D-G-I-C-K-B-M-L-B-M
P7 : A-B-C-D-G-H-G-I-C-K-B-M-L-B-M
P8 : A-B-C-D-E-G-H-G-I-C-K-B-M-L-B-M

Test cases:

Test case ID	Paths	Inputs	first, last	Expected output	Actual output	Status.
01.	P1	5	1,1	Sorted (5)	Sorted (5)	Success
02.	P2	5,4	1,2	Sorted (4,5)	Sorted (4,5)	Success
03.	P3	1,2,3	1,3	Sorted(1,2,3)	Sorted (1,2,3)	Success
04.	P4	1,2,3,4,5	1,5	Sorted(1,2,3,4,5)	Sorted (1,2,3,4,5)	Success
05.	P5	5,4,3,2,1	1,5	Sorted(1,2,3,4,5)	Sorted (1,2,3,4,5)	Success
06.	P6	1,4,3,2,5	1,5	Sorted(1,2,3,4,5)	Sorted(1,2,3,4,5)	Success
07.	P7	5,2,3,1,4	1,5	Sorted(1,2,3,4,5)	Sorted (1,2,3,4,5)	Success
08.	P8	5,2,3,1,4	1,5	Sorted(1,2,3,4,5)	Sorted(1,2,3,4,5)	Success

```
printf("%d", a[i]);
}
else
{
printf("size of array is invalid");
}
return 0;
}
```

Test report.
no of test cases executed : 8
no of test cases pass = 8
no of defects exist = 0
no of test cases fail = 0

5
5 4
1 2 3
1 2 3 4 5

Algorithm

Step1: start
Step2: Input the student percentage P.
Step3: if (P≥90) grade=A goto step 4.1
else if (P≥80) && (P<90) grade = B goto step 4.2
else if ((P≥70) && (P<80) grade = C goto step 4.3
else if ((P≥60) && (P<70)) grade = D goto step 4.4
else
grade = E goto step 4.5
Step4: Print the grade message with respect to the grade
Step4.1: Case A: print grade, excellent goto step5
Step4.2: Case B: print grade; Very good goto step5
Step4.3: Case C: print grade; good goto step5
Step4.4: Case D: print grade, satisfactory goto step5
Step4.5: Case E: print grade, fail goto step5.

Testing:
Technique used: Basis path testing.

12. 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 drive different test cases, execute these test cases and discuss the test results.

REQUIREMENT.
R1: The system should accept the percentage of the student.
R2: Based on the R1 the grade of the student is determined.

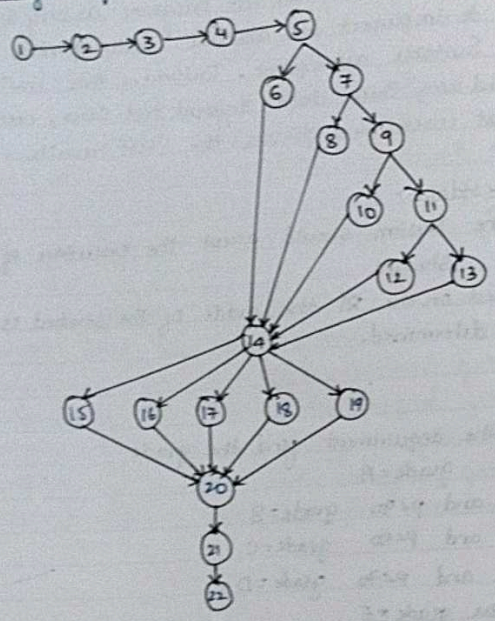
Design:
From the requirement find the grade
P ≥ 90 grade = A
P ≥ 80 and P < 90 grade = B
P ≥ 70 and P < 80 grade = C.
P ≥ 60 and P < 70 grade = D
otherwise grade = E

Program code:

Prgm 12
Pg 1

```
#include<stdio.h>
void main()
{
1 char grade, A,B, C, D, E;
2 float P;
3 printf("Enter the student percentage.");
4 scanf("%f", &P);
```

Program graph:



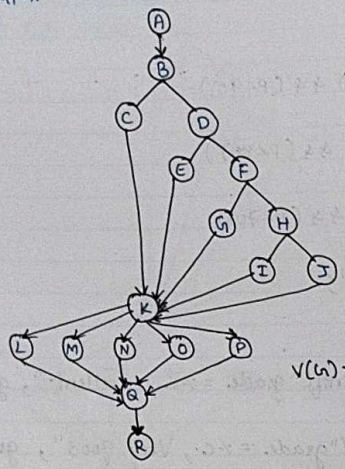
SAMBHRAM INSTITUTE OF TECHNOLOGY

```
5   if (P≥90)
6   grade = 'A';
7   else if ( (P≥80) && (P<90))
8   grade = 'B';
9   else if((P≥70) && (P<80))
10. grade = 'C';
11  else if ((P≥60) && (P<70))
12  grade = 'D';
13  else grade = 'E';
14  Switch (grade)
    {
15    Case 'A' : printf("grade = %c, excellent", grade);
        break;
16    Case 'B': printf"grade = %c, Very good", grade);
        break;
17    Case 'C' : printf("grade = %c, good", grade);
        break;
18    Case 'D': printf("grade = %c, Satisfactory", grade);
        break;
19    Case 'E': printf("grade = %c, fail", grade);
20  }
21  getche();
22  }
```

Test report.

no of test cases executed = 9
no of defects cases = 0
no of test cases pass = 9
no of test cases failed = 0.

DD-graph.



V(G) = e-n+2p
(25-18)+2
9

Predicate+1
8+1
9

P1: A-B-D-F-H-J-K-P-Q-R
P2: A-B-D-F-H-I-K-O-Q-R
P3: A-B-D-F-G-K-N-Q-R.
P4: A-B-D-E-K-M-Q-R
P5: A-B-C-K-L-Q-R
P6: A-B-D-F-H-J-K-L-Q-R
P7: A-B-D-F-H-J-K-M-Q-R
P8: A-B-D-F-H-J-K-N-Q-R
P9: A-B-D-F-H-J-K-O-Q-R.

Test report.
No of testcases executed = 9
No of defects cases = 0
No of Test cases pass = 9
No of Test cases fail = 0

Test case ID.	Description	Input	Expected output	Actual output	Status
01.	P1	41	grade=E, Fail	grade=E; Fail	Succ
02.	P2	62	grade=D, Satisfactory	grade=D, Satisfactory	Succ
03.	P3	79	grade=C, Good.	grade=C, Good	Succ
04.	P4	80	grade=B, Very good	grade=B, Very good	Succ
05.	P5	90	grade=A, Excellent	grade=A, Excellent	Succ
06.	P6	91	grade=A, Excellent	grade=A, Excellent	Succ
07.	P7	81	grade=B, Very good	grade=B, Very good	Succ
08.	P8	71	grade=C, good	grade=C, good	Succ
09.	P9	61	grade=D, Satisfactory	grade=D, Satisfactory.	Succ