JYOTHY INSTITUTE OF TECHNOLOGY
AFFILIATED TO VTU, BELAGAVI
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
ACCREDITED BY NBA, NEW DELHI

LAB MANUAL
FOR
VI SEMESTER

SOFTWARE TESTING LABORATORY
(18ISL66)

SOFTWARE TESTING LABORATORY

Subject Code: 18ISL66 CIE Marks: 40
Hours/Week: 03 Exam Hours: 03
Total Hours: 36 SEE Marks: 60

Programs List:

1. 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.

2. 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.

3. 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.

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

5. 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.

6. 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.

7. 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.

8. 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.

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

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.

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.

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

TRIANGLE PROBLEM

```
#include<stdio.h>
void main(){
int a,b,c;
printf("enter the three values");
scanf("%d %d %d",&a,&b,&c);
if(a>=0 && b>=0 && c>=0)
{
if(a==b&&b==c&&c==a)
{
printf("it is equilateral");
}
else if(a==b || b==c || c==a)
{
printf("it is isosceles");
}
else
{
printf("it is scalenE");
}
}
else{
if(a<0 && b>0 && c>0)
{
printf("value of a is not in range");
}

else if(b<0 && c>0 && a>0)
{
printf("value of b is not in range");
}
else if(c<0 && a>0 && b>0)
{
printf("value of c is not in range");
}
else if(a<0 && b<0 && c>0)
{
printf("value of a and b is not in range");
}
else if(a<0 && b>0 && c<0)
{
printf("value of a and c is not in range");
}
else if(a>0 && b<0 && c<0)
{
printf("value of b and c is not in range");
}
```

```
else
{
printf("value of a,b and c is not in range");
}
}
}
```

1. 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.

<u>Test Case Name :Boundary Value Analysis</u>

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

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

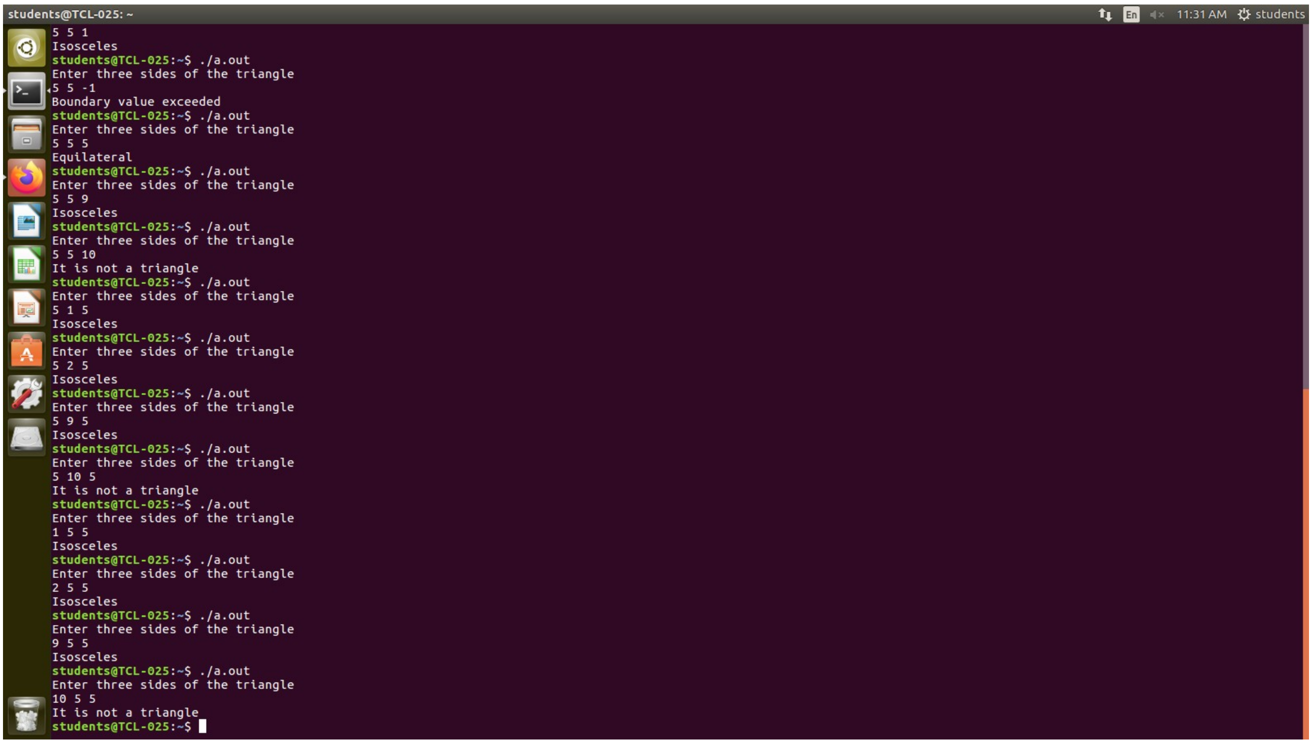
Brief Description : Check whether given value for an Equilateral, Isosceles , Scalene triangle or can't form triangle

Triangle Problem -Boundary value Test cases for input data

Case Id	Description	Input Data			Expected Output	Actual Output	Status
		a	b	c			
1		5	5	1	Isosceles		
2		5	5	-1	Boundary value exceeded		
3		5	5	5	Equilateral		
4		5	5	9	Isosceles		
5		5	5	10	Not a triangle		
6		5	1	5	Isosceles		
7		5	2	5	Isosceles		
8		5	9	5	Isosceles		
9		5	10	5	Isosceles		
10		1	5	5	Isosceles		
11		2	5	5	Isosceles		
12		9	5	5	Isosceles		

13		10	5	5	Not a triangle		

Output Snapshots:



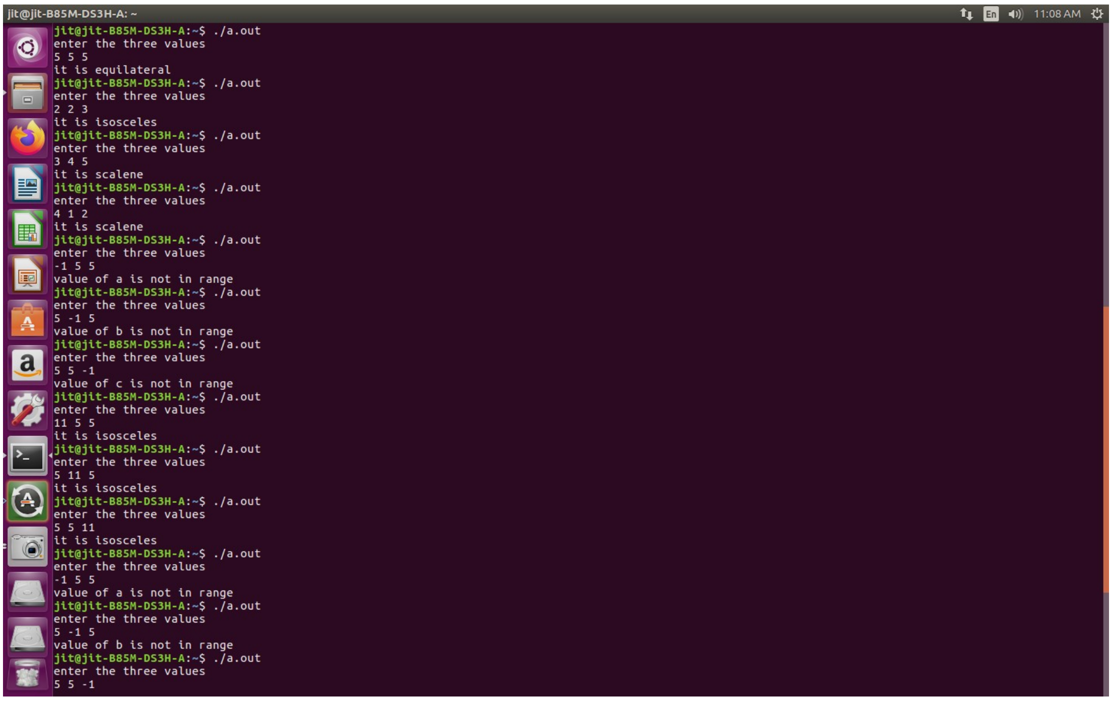
4. Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of a triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they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

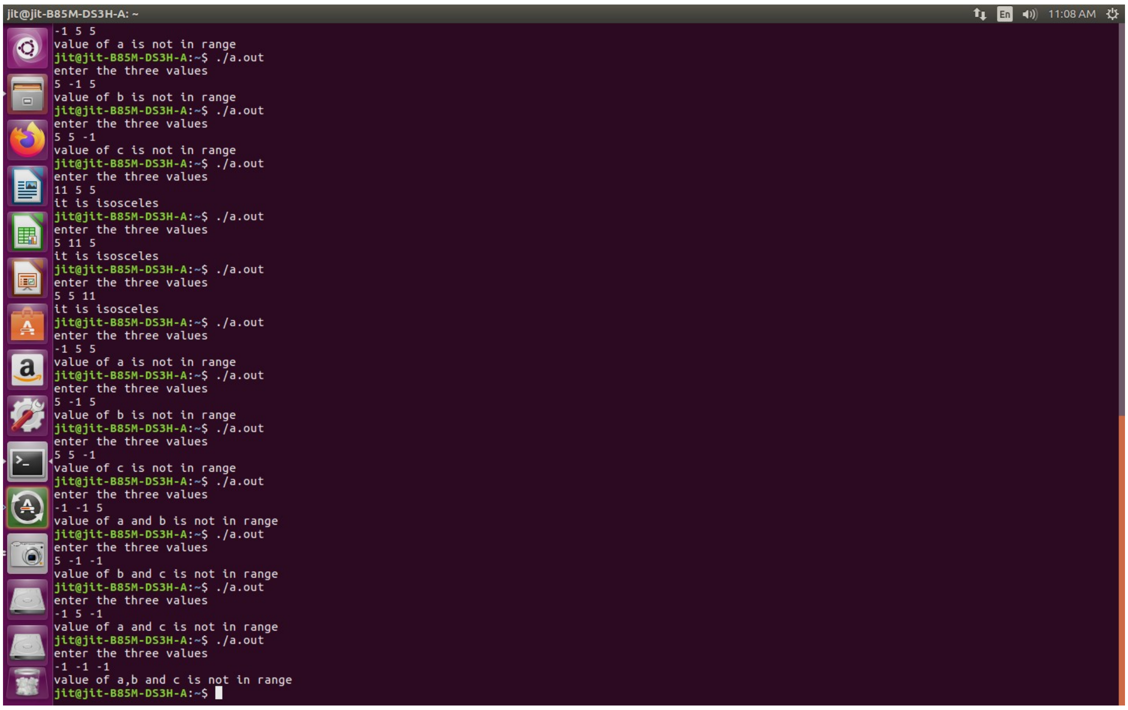
<u>Test Case Name : Equivalence class partitioning</u>

Case Id	Description	Input Data			Expected Output	Actual Output	Status
		a	b	c			
1	WN1	5	5	5	Equilateral		
2	WN2	2	2	3	Isosceles		
3	WN3	3	4	5	Scalene		
4	WN4	4	1	2	Scalene		
5	WR1	-1	5	5	Value of a not in range		
6	WR2	5	-1	5	Value of b not in range		
7	WR3	5	5	-1	Value of c not in range		
8	WR4	11	5	5	Isosceles		
9	WR5	5	11	5	Isosceles		
10	WR6	5	5	11	Isosceles		

Case Id	Description	Input Data			Expected Output	Actual Output	Status
1	SR1	-1	5	5	Value of a not in range		
2	SR2	5	-1	5	Value of b not in range		
3	SR3	5	5	-1	Value of c not in range		
4	SR4	-1	-1	5	Value of a and b not in range		
5	SR5	-1	5	-1	Value of b and c not in range		
6	SR6	5	-1	-1	Value of a and c not in range		
7	SR7	-1	-1	-1	Value of a and b and c not in range		

Output Snapshots:





7. 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.

<u>Test Case Name : Decision Table Testing</u>

C1: a<b+c ; C2: b<a+c ; C3: c<b+a

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

a1: Not a triangle ; a2: Scalene ; a3: Isosceles

a4: Equilateral ; a5: Impossible

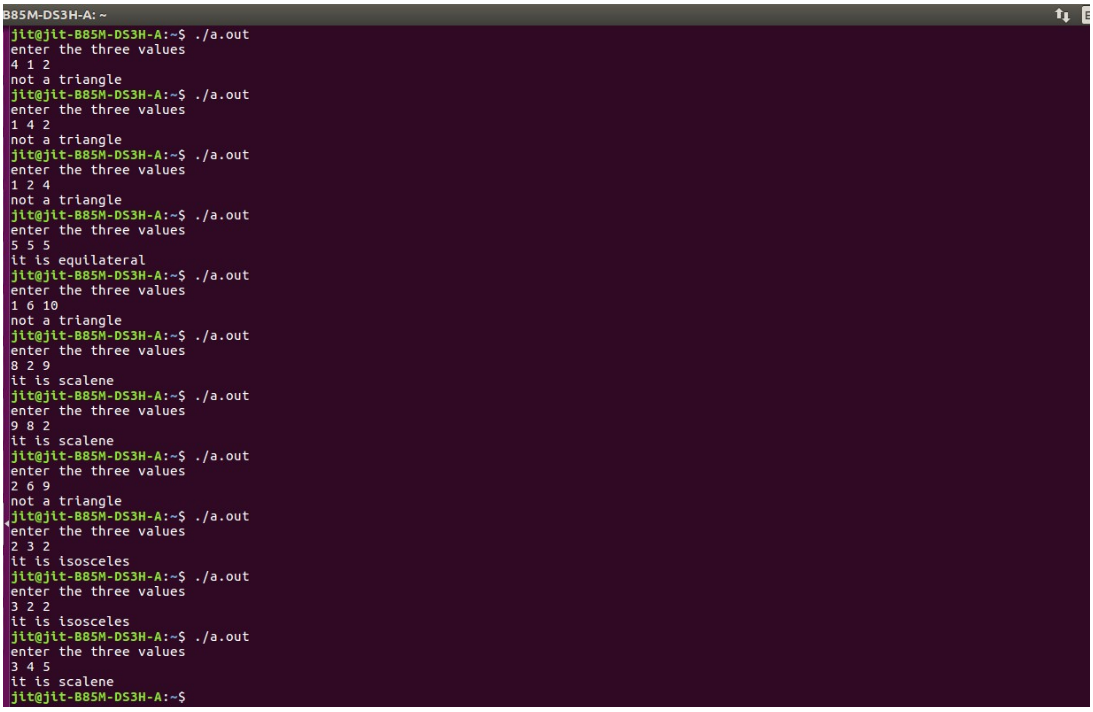
	RULES →	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
CONDITIONS	C1: a<b+c	F	T	T	T	T	T	T	T	T	T	T
	C2: b<a+c	-	F	T	T	T	T	T	T	T	T	T
	C3: c<b+a	-	-	F	T	T	T	T	T	T	T	T
	C4: a=b?	-	-	-	T	T	T	F	T	F	F	F
	C5: a=c?	-	-	-	T	T	F	T	F	T	F	F
	C6: c=b?	-	-	-	T	F	T	T	F	F	T	F
ACTIONS	a1: Not a triangle	Y	Y	Y								
	a2: Scalene											Y
	a3: Isosceles								Y	Y	Y	
	a4: Equilateral				Y							
	a5: Impossible					Y	Y	Y				

DECISION TABLE:

Case Id	Description	Input Data			Expected Output	Actual Output	Status
1	DT1	4	1	2	Not a triangle		
2	DT2	1	4	2	Not a triangle		
3	DT3	1	2	4	Not a triangle		

Case Id	Description	Input Data			Expected Output	Actual Output	Status
4	DT4	5	5	5	Equilateral		
5	DT5	2	?	?	Impossible		
6	DT6	?	2	?	Impossible		
7	DT7	?	?	2	Impossible		
8	DT8	3	2	2	Isosceles		
9	DT9	2	3	2	Isosceles		
10	DT10	2	2	3	Isosceles		
11	DT11	3	4	5	Scalene		

Output Snapshot:



```
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
4 1 2
not a triangle
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
1 4 2
not a triangle
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
1 2 4
not a triangle
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
5 5 5
it is equilateral
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
1 6 10
not a triangle
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
8 2 9
it is scalene
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
9 8 2
it is scalene
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
2 6 9
not a triangle
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
2 3 2
it is isosceles
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
3 2 2
it is isosceles
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the three values
3 4 5
it is scalene
jit@jit-B85M-DS3H-A:~$
```

COMMISSION PROBLEM

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

Program Code

```
#include<stdio.h>
int flag=0;
void main(){
int l,s,b,lock,stock,barrel,total;
float commission;
printf("Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:\n");
scanf("%d %d %d",&l,&s,&b);
if(l<1 || l>70)
{
printf("value of lock not in range\n");
flag=1;
}
if(s<1 || s>80)
{
printf("value of stock not in range\n");
flag=1;
}
```

```
if(b<1 || b>90)
{
printf("value of barrel not in range\n");
flag=1;
}
else {
lock=45*l;
stock=30*s;
barrel=25*b;
total=lock+stock+barrel;
if (flag==0)
{
printf("total is:%d\n",total);

if(total>=1 && total<=1000)
{
commission=total*0.1;
printf("commission is:%f\n",commission);
}
else if (total>=1001 && total<=1800)
{
commission=1000*0.1+(total-
printf("commission is:%f\n",commission);
}
else if (total>1800)
{
commission=1000*0.1+(800)*0.
printf("commission is:%f\n",commission);
}
}
}
}
```

2. 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.

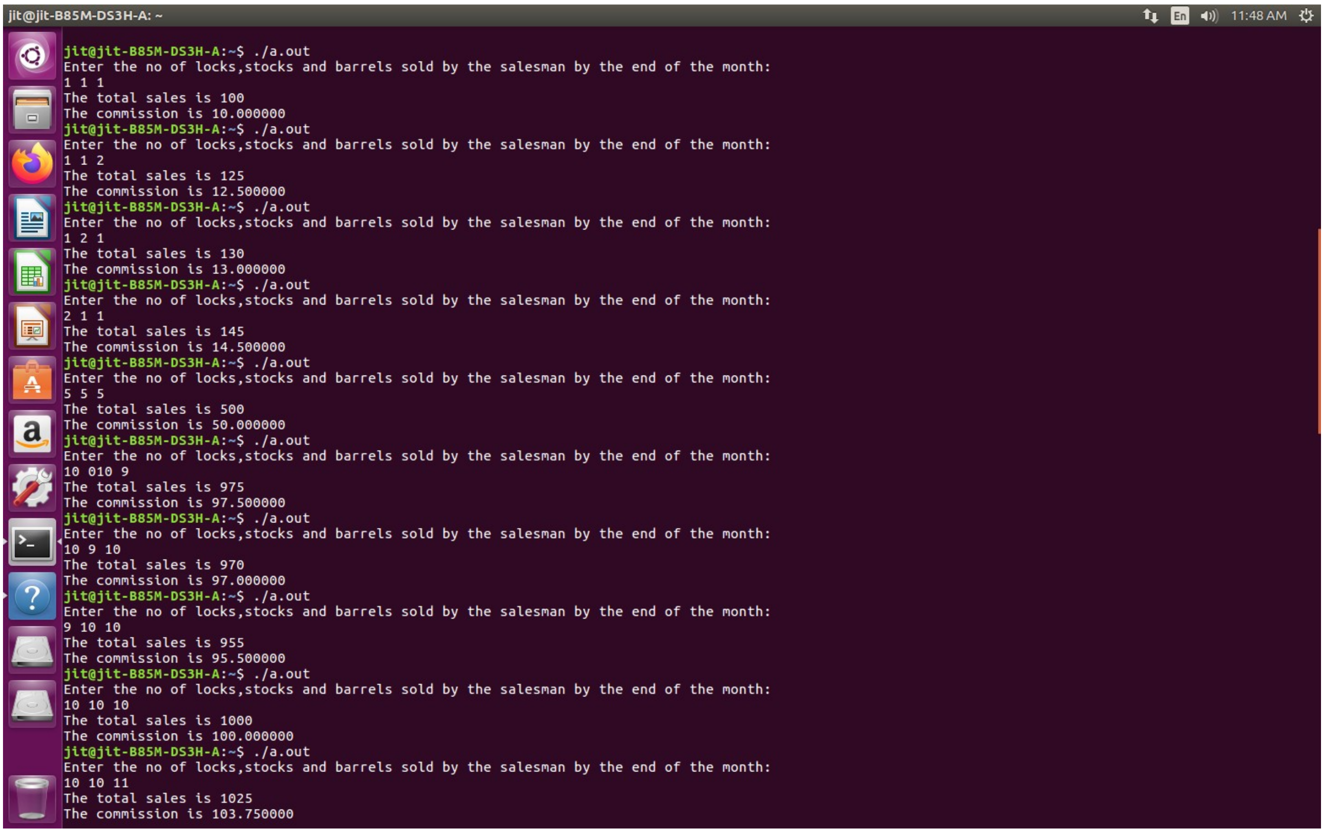
Test Case Name :Boundary Value Analysis

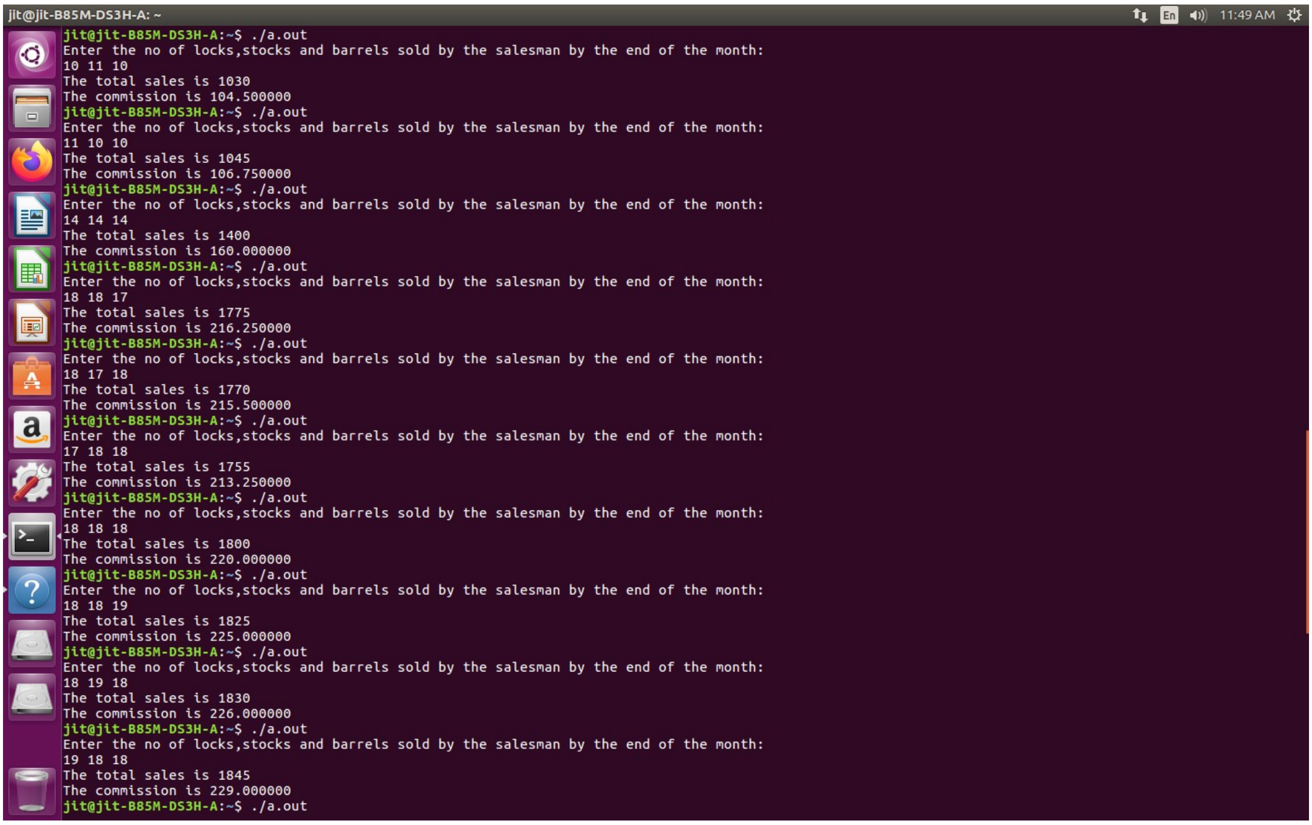
TEST CASES:

Case id	Description	Input data			Expected output		Actual output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1.	Enter the min values for locks, stocks and barrels	1	1	1	100	10			
2	Min for 2 items and +1 for any 1	1	1	2	125	12.5			
3		1	2	1	130	13			
4		2	1	1	145	14.5			
5	Enter the values such that value of sales is mid value between 100 and 1000	5	5	5	500	50			
6	Enter values such that value of sales is nearly less than 1000	10	10	9	975	97.5			
7		10	9	10	970	97			
8		9	10	10	955	95.5			
9.	Enter values such that value of sales is exactly 1000	10	10	10	1000	100			
10	Enter values such that value of sales is nearly greater than 1000	10	10	11	1025	103.75			
11		10	11	10	1030	104.5			
12		11	10	10	1045	106.73			
13	Enter the values such that value of sales is mid value between 1000 and 1800	14	14	14	1400	160			
14	Enter values such that value of sales is nearly less than 1800	18	18	17	1775	216.25			
15		18	17	18	1770	215.5			
16		17	18	18	1755	213.25			

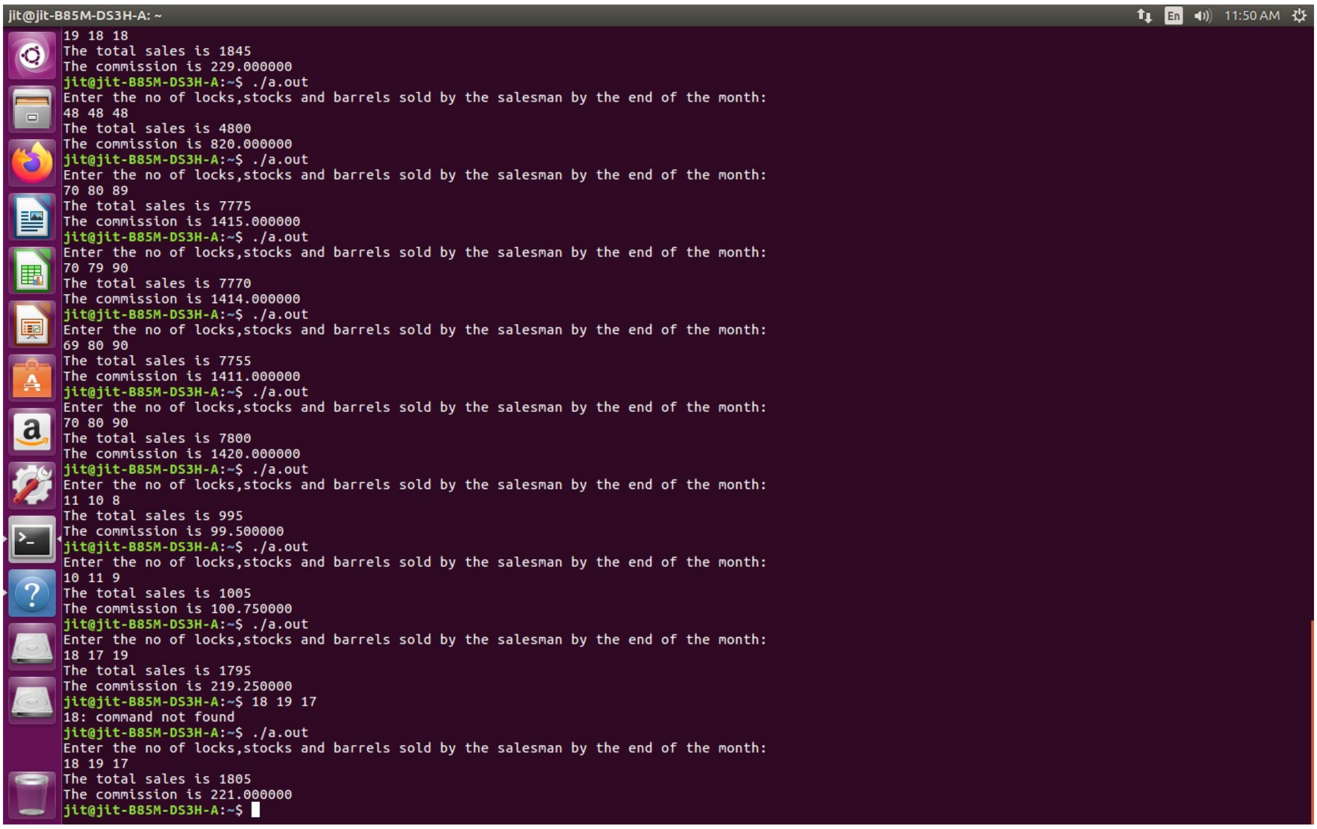
17	Enter values such that value of sales is exactly 1800	18	18	18	1800	220			
18	Enter values such that value of sales is nearly greater than 1800	18	18	19	1825	225			
19		18	19	18	1830	226			
20		19	18	18	1845	229			
21	Enter normal values for locks, stocks and barrels	48	48	48	4800	820			
22	Enter the max value for 2 items and max-1 for any one item	70	80	89	7775	1415			
23		70	79	90	7770	1414			
24		69	80	90	7755	1411			
25	Enter the max values for locks, stocks and barrels	70	80	90	7800	1420			

Output Snapshots:





```
jit@jit-B85M-DS3H-A: ~
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
10 11 10
The total sales is 1030
The commission is 104.500000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
11 10 10
The total sales is 1045
The commission is 106.750000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
14 14 14
The total sales is 1400
The commission is 160.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 18 17
The total sales is 1775
The commission is 216.250000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 17 18
The total sales is 1770
The commission is 215.500000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
17 18 18
The total sales is 1755
The commission is 213.250000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 18 18
The total sales is 1800
The commission is 220.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 18 19
The total sales is 1825
The commission is 225.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 19 18
The total sales is 1830
The commission is 226.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
19 18 18
The total sales is 1845
The commission is 229.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
```



```
jit@jit-B85M-DS3H-A: ~
19 18 18
The total sales is 1845
The commission is 229.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
48 48 48
The total sales is 4800
The commission is 820.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
70 80 89
The total sales is 7775
The commission is 1415.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
70 79 90
The total sales is 7770
The commission is 1414.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
69 80 90
The total sales is 7755
The commission is 1411.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
70 80 90
The total sales is 7800
The commission is 1420.000000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
11 10 8
The total sales is 995
The commission is 99.500000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
10 11 9
The total sales is 1005
The commission is 100.750000
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 17 19
The total sales is 1795
The commission is 219.250000
jit@jit-B85M-DS3H-A:~$ 18 19 17
18: command not found
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
18 19 17
The total sales is 1805
The commission is 221.000000
jit@jit-B85M-DS3H-A:~$
```

5. 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.

<u>Test Case Name : Equivalence class partitioning</u>

WEAK AND STRONG NORMAL EQUIVALENCE CLASS

Case id	Description	Input data			Expected output		Actual output		Status
		Locks	Stocks	Barrels	Sales	Commission	Sales	Commission	
1.	Enter the values within the rangefor locks, stocks and barrels	35	40	45	3900	640			

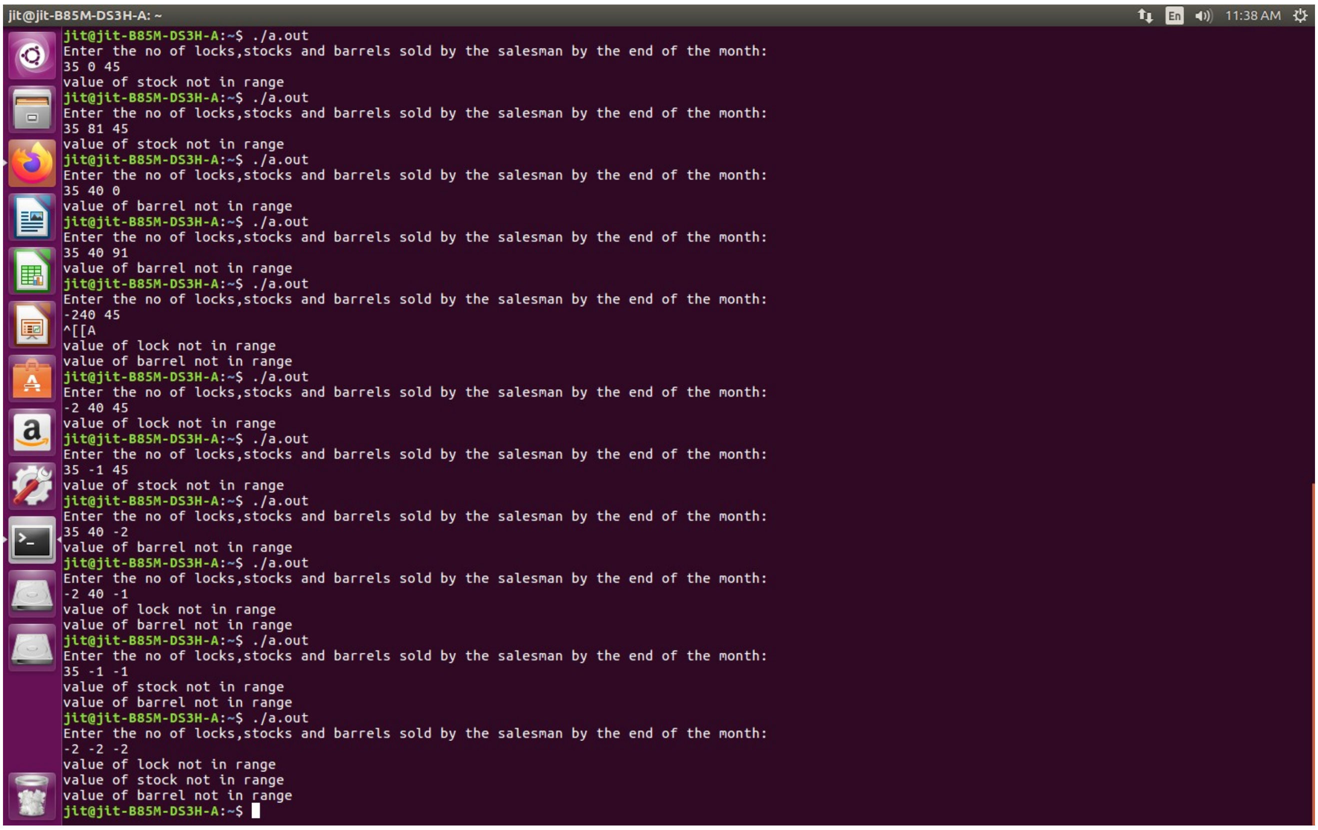
WEAK ROBUST EQUIVALENCE CLASS

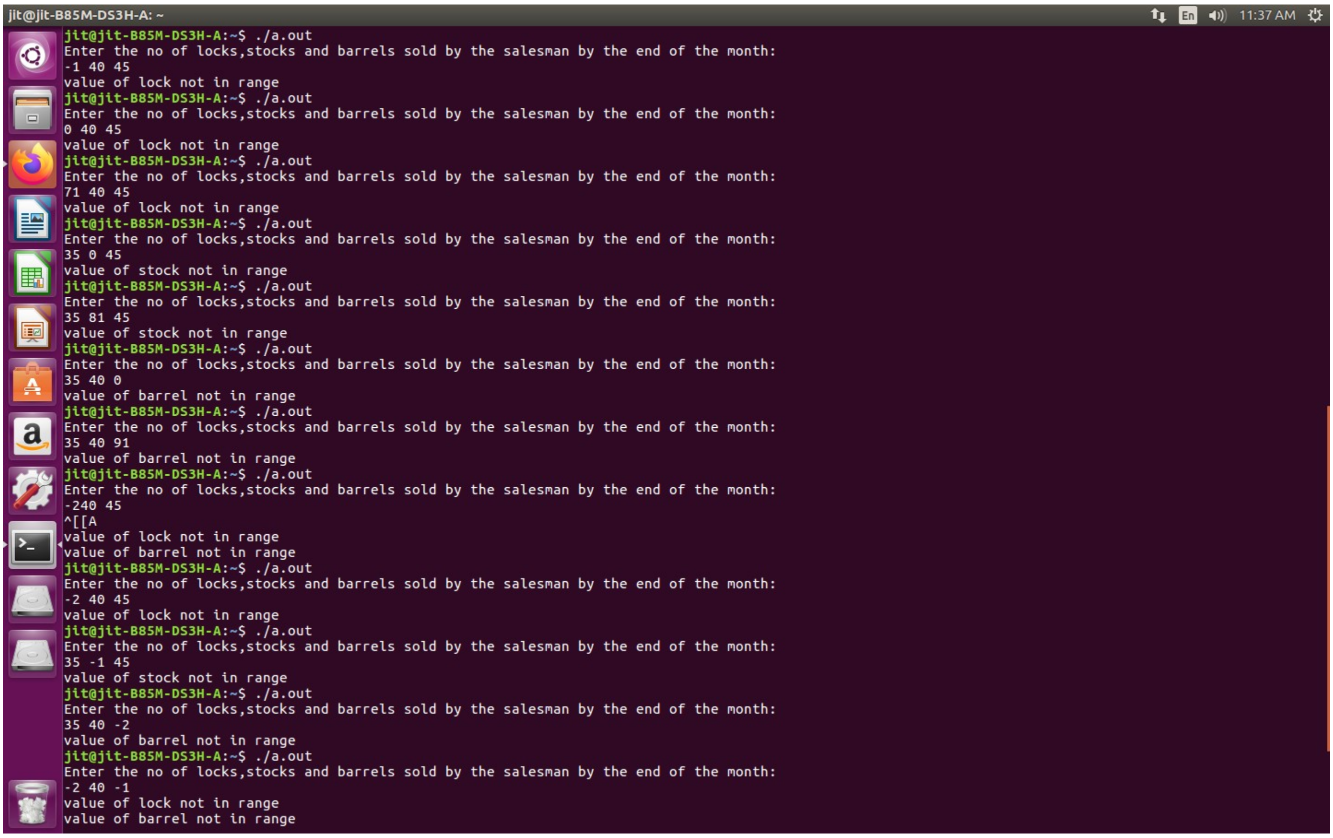
Case id	Description	Input data			Expected output	Actual output	Status
		Locks	Stocks	Barrels			
1.	WR1	-1	40	45	Terminates input loop and proceeds to calcualte sales and commission		
2	WR2	0	40	45	Value of locks not in range		
3	WR3	71	40	45	Value of locks not in range		
4	WR4	35	0	45	Value of stocks not in range		
5	WR5	35	81	45	Value of stocks not in range		
6	WR6	35	40	0	Value of barrels not in range		
7	WR7	35	40	91	Value of barrels not in range		

STRONG ROBUST EQUIVALENCE CLASS

Case id	Description	Input data			Expected output	Actual output	Status
		Locks	Stocks	Barrels			
1.	SR1	-1	40	45	Value of locks not in range		
2	SR2	35	-1	45	Value of stocks not in range		
3	SR3	35	40	-1	Value of barrels not in range		
4	SR4	-2	-1	45	Value of locks and stocks not in range		
5	SR5	35	-1	-1	Value of stocks and barrels not in range		
6	SR6	-1	40	-1	Value of locks and barrels not in range		
7	SR7	-2	-2	-2	Value of locks, stocks and barrels not in range		

Output Snapshots:





```
jit@jit-B85M-DS3H-A: ~                                    11:37 AM
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
-1 40 45
value of lock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
0 40 45
value of lock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
71 40 45
value of lock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 0 45
value of stock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 81 45
value of stock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 40 0
value of barrel not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 40 91
value of barrel not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
-240 45
^[[A
value of lock not in range
value of barrel not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
-2 40 45
value of lock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 -1 45
value of stock not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
35 40 -2
value of barrel not in range
jit@jit-B85M-DS3H-A:~$ ./a.out
Enter the no of locks,stocks and barrels sold by the salesman by the end of the month:
-2 40 -1
value of lock not in range
value of barrel not in range
```

8. 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.

<u>Test Case Name : Decision Table Testing</u>

<u>Conditions:</u>

C1: 1<=locks<=70 ; C2: 1<=stocks<=70 ; C3: 1<=barrels<=70

C4: sales>1800 ; C5: 1800>sales>1000 ; C6: sales<1000

<u>Actions:</u>

a1: com1 = 0.1* sales ; a2: com2 = com1+0.15*(sales-1000) ;

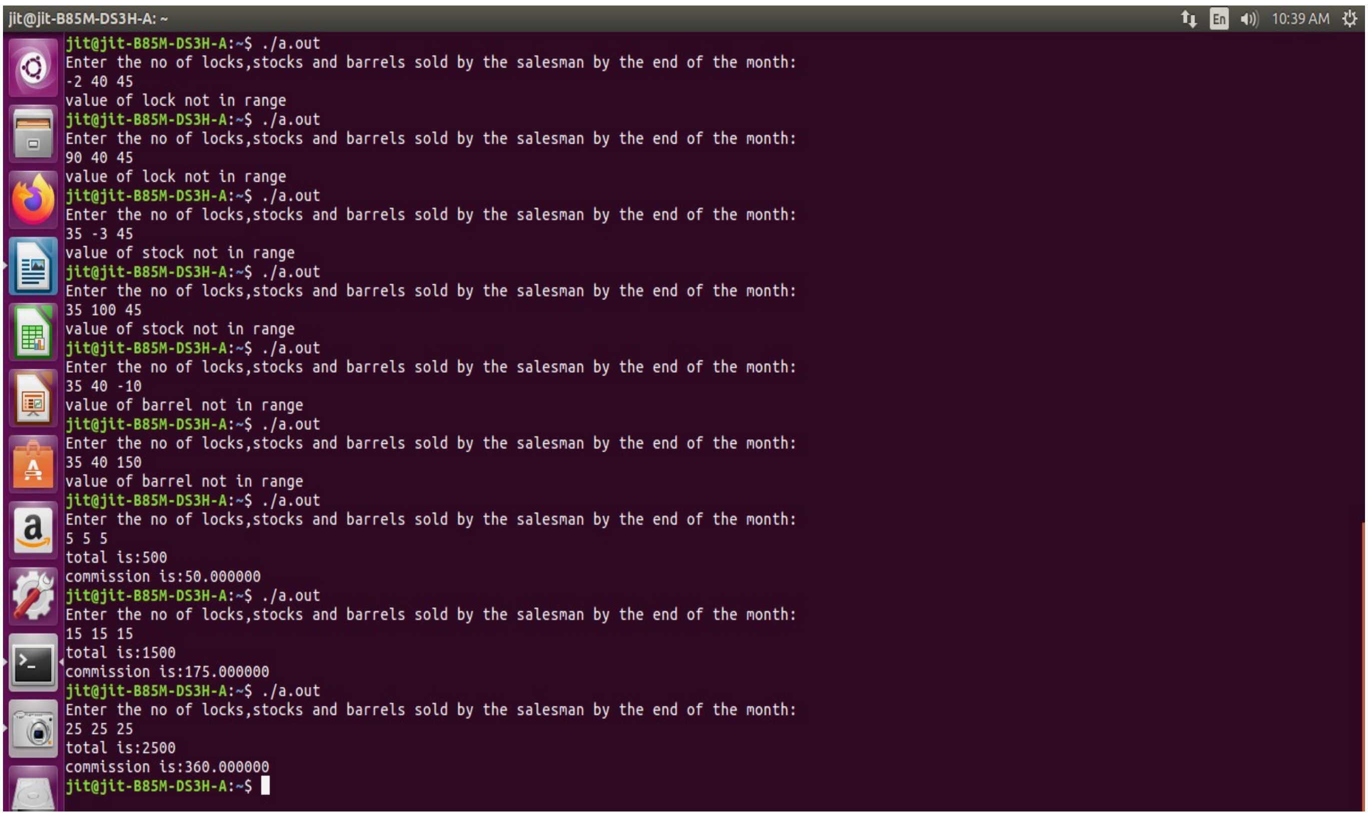
a3: com2+0.20*(sales-1800) ; a4: Out of range

	RULES →	R1	R2	R3	R4	R5	R6
CONDITIONS	C1: 1<=locks<=70	F	T	T	T	T	T
	C2: 1<=stocks<=70	-	F	T	T	T	T
	C3: 1<=barrels<=70	-	-	F	T	T	T
	C4: sales>1800	-	-	-	T	T	T
	C5: 1800>sales>1000	-	-	-	T	T	F
	C6: sales<1000	-	-	-	T	F	T
ACTIONS	a1: com1 = 0.1* sales						X
	a2: com2 = com1+0.15*(sales-1000)					X	
	a3: com2+0.20*(sales-1800)				X		
	a4: Out of range	X	X	X			

DECISION TABLE:

Case Id	Description	Input Data			Expected Output	Actual Output	Status
1	DT1	-2	40	45	Out of range		
2	DT2	90	40	45	Out of range		
3	DT3	35	-3	45	Out of range		
4	DT4	35	100	45	Out of range		
5	DT5	35	40	-10	Out of range		
6	DT6	35	40	95	Out of range		
7	DT7	5	5	5	500		
8	DT8	15	15	15	1500		
9	DT9	25	25	25	2500		

Output Snapshots:



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

<u>Test Case Name : Data Flow Testing</u>

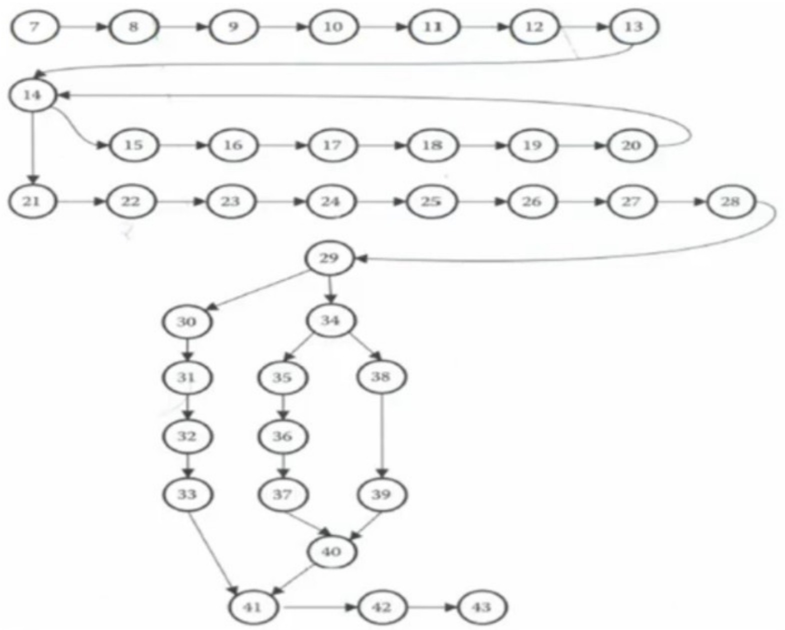
ALGORITHM:

```
1. Program Commission (INPUT,OUTPUT)
2. Dim locks, stocks, barrels as Integer
3. Dim lockPrice, stockPrice, barrelPrice as Real
4. Dim totalLocks, totalStocks, totalBarrels as Integer
5. Dim locksSales, stockSales, barrelSales as Real
6. Dim sales and commission as Real
7.  lockPrice = 45.0
8.   stockPrice = 30.0
9.   barrelPrice = 25.0
10.  totalLocks = 0
11.  totalStocks = 0
12.  totalBarrels = 0
13. Input(locks)
14. While NOT(locks = -1)
15.       Input(stocks,barrels)
16.        totalLocks =  totalLocks+locks
17.        totalStocks =  totalStocks+stocks
18.        totalBarrels =  totalBarrels+barrels
19.       Input(locks)
20. Endwhile
21. Output("Locks sold:" , totalLocks)
22.  Output("Stocks sold:" , totalStocks)
23.  Output("Barrels sold:" , totalBarrels)
24.  locksSales =  lockPrice *  totalLocks
25.  stockSales = stockPrice *  totalStocks
26.  barrelSales  = barrelPrice *  totalBarrels
```

27. slaes = locksSales + stockSales + barrelSales

28. Output (“Total sales:”, slaes)
29. if(sales>1800)
    30. then
    31. commission = 0.10* sales
32. commission = commission + 0.15*800.0
33. commission = commission + 0.20*(sales-1800.0)
34. Else if (sales >1000)
35. Then
36. commission = 0.10* 1000.0
37. commission = commission + 0.15*(sales-1000.0)
38. Else
39. commission = 0.10 * sales
40. Endif
41. Endif
42. Output(“Commission is” , commission)
43. End Commission

Path diagram:



The D-U Table:

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

NEXT DATE PROBLEM

```
#include<stdio.h>
#include<stdlib.h>
void main(){

int days=31,month,year,date;
while(1){
printf("\nEnter The MM-DD-YYYY\n");
scanf("%2d%2d%4d",&month,&date,&year);

if(year>=1812 && year<=2012){
if(month<=12 && month>=1){
if(date<=days && date>0){
if(date==days && month<12){
date=1;
month+=1;
}
else if(date<days){
date+=1;
}
else if(date==days && month==12){
date=1;
month=1;
year+=1;
}
printf("\nNext Date Is : %d-%d-%4d\n",month,date,year);
}
else{
if(date<1 || date>days){
printf("\nInvalid Date\n");
}
if(month<1 || month>12){
printf("\nInvalid Month\n");
}
if(year<1812 || month>2012){
printf("\nInvalid Year\n");
}
exit(0);
}
}
else{
if(date<1 || date>days){
printf("\nInvalid Date\n");
}
if(month<1 || month>12){
```

```
printf("\nInvalid Month\n");
}
if(year<1812 || month>2012){
printf("\nInvalid Year\n");
}
exit(0);
}
}
else{
if(date<1 || date>days){
printf("\nInvalid Date\n");
}
if(month<1 || month>12){
printf("\nInvalid Month\n");
}
if(year<1812 || year>2012){
printf("\nInvalid Year\n");
}
exit(0);
}
}
}
```

OR

```
#include<stdio.h>
int 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,date,year:\n");
	scanf("%d%d%d",&m,&d,&y);
	ndays=month[m-1];
	if(y<1812 || y>2012)
	{
		printf("invalid input year\n");
	}
	if(d<=0||d>ndays)
	{
		printf("invalid input day\n");
	}
	if(m<1||m>12)
	{
		printf("invalid input month");
	}
	if(m==2)
	{
		if(y%100==0)
		{
```

```
                if(y%400==0)
                ndays = 29;
            }
            else if(y%4 == 0)
            ndays = 29;
        }
        nd=d+1;
        nm=m;
        ny=y;
        if(nd>ndays)
        {
            nd=1;
            nm++;
        }
        if(nm>12)
        {
            nm=1;
            ny++;
        }
        printf("Given date is %d:%d:%d",d,m,y);
        printf("\nNext day's date is %d:%d:%d\n",nd,nm,ny);
}
```

3. 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.

<u>Test Case Name :Boundary Value Analysis</u>

Considering Date program, we have three variables day, month and year.

Min Min+ Nom Max- Max values for the following variables are:

<u>day</u> :1, 2, 15, 30, 31

<u>month</u>: 1, 2, 6, 11, 12

<u>year</u>: 1812, 1813, 1912, 2011, 2012

TEST CASES:

Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1		12	31	1812	Invalid i/p						
2		12	31	2012	1	1	2013				
3		12	31	2013	Invalid i/p						
4		1	1	1812	1	2	1812				
5		1	1	1813	1	2	1813				
6		1	1	1912	1	2	1912				
7		1	1	2011	1	2	2011				
8		1	1	2012	1	2	2012				
9.		1	2	1812	1	2	1812				
10		1	2	1813	1	3	1813				
11		1	2	1912	1	3	1912				
12		1	2	2011	1	3	2011				
13		1	2	2012	1	3	2012				
14		1	15	1812	1	16	1812				
15		1	15	1813	1	16	1813				
16		1	15	1912	1	16	1912				
17		1	15	2011	1	16	2011				
18		1	15	2012	1	16	2012				
19		1	30	1812	1	31	1812				

20		1	30	1813	1	31	1813				
21		1	30	1912	1	31	1912				
22		1	30	2011	1	31	2011				
23		1	30	2012	1	31	2012				
24		1	31	1812	2	1	1812				
25		1	31	1813	2	1	1813				
26.		1	31	1912	2	1	1912				
27		1	31	2011	2	1	2011				
28		1	31	2012	2	1	2012				

6. 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.

<u>Test Case Name : Equivalence class partitioning</u>

Design:

<u>1st attempt</u>:

Intervals

Valid intervals : M1 : { month : 1<= month <= 12}
D1 : { day : 1<= day <= 31}
Y1 : {year: 1812<= year <= 2012}

WEAK AND STRONG NORMAL EQUIVALENCE CLASS

Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1	Valid input for all	6	15	1912	6	16	1912				

Since no. of variables is equal to number of valid classes , only one weak normal equivalence class test case occurs which is the same as the strong normal equivalence class test case.

WEAK ROBUST EQUIVALENCE CLASS

Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1.	WR1	6	15	1912	6	16	1912				
2	WR2	6	-1	1912	Day not in range						
3	WR3	6	32	1912	Day not in range						
4	WR4	-1	15	1912	Month not in range						
5	WR5	13	15	1912	Month not in range						
6	WR6	6	15	1811	Year not in range						
7	WR7	6	15	2013	Year not in range						

STRONG ROBUST EQUIVALENCE CLASS

Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1.	SR1	-1	15	1912	Invalid						
2	SR2	6	-1	1912	Day not in range						
3	SR3	6	15	1811	Year not in range						
4	SR4	-1	-1	1912	Day and Month not in range						
5	SR5	6	-1	1811	Month and Year not in range						
6	SR6	-1	15	1811	Day and Year not in range						
7	SR7	-1	-1	1811	Day, Month and Year not in range						

2nd attempt:

Equivalence classes

M1 : {month : month has 30 days}
M2 : {month : month has 31 days}
M3 : {month : month is february}

D1 : { day : 1<= day <= 28}
D2 : { day : day = 29}
D3 : { day : day = 30}
D4 : { day : day = 31}

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

WEAK NORMAL EQUIVALENCE CLASS

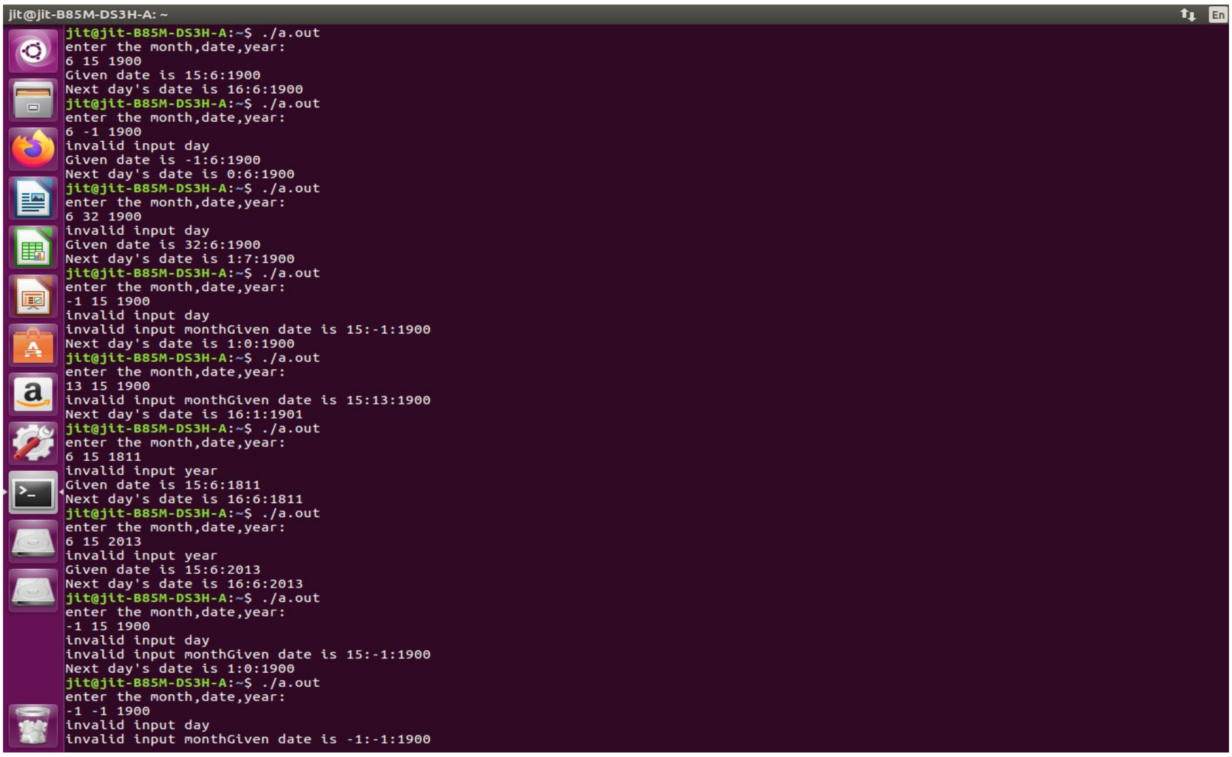
Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1.	WN1	6	14	2000	6	15	2000				
2.	WN2	7	29	1996	7	30	1996				
3.	WN3	2	30	2002	Impossible						
4.	WN4	6	31	2000	Impossible						

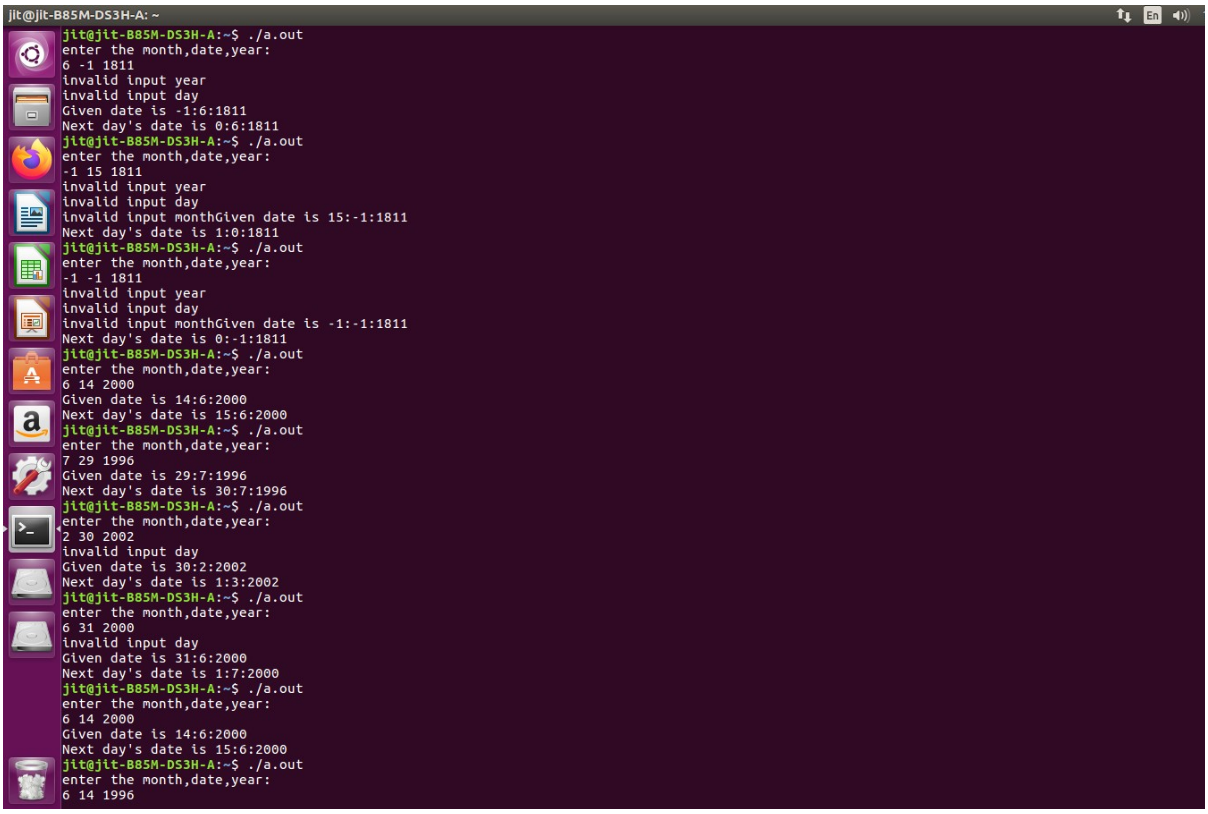
STRONG NORMAL EQUIVALENCE CLASS

Case id	Description	Input data			Expected Output			Actual output			Status
		M	D	Y	M	D	Y	M	D	Y	
1	SN1	6	14	2000	6	15	2000				
2	SN2	6	14	1996	6	15	1996				
3	SN3	6	14	2002	6	15	2002				
4	SN4	6	29	2000	6	30	2000				
5	SN5	6	29	1996	6	30	1996				
6	SN6	6	29	2002	6	30	2002				
7	SN7	6	30	2000	7	1	2000				
8	SN8	6	30	1996	7	1	1996				
9.	SN9	6	30	2002	7	1	2002				
10	SN10	6	31	2000	Invalid						
11	SN11	6	31	1996	Invalid						
12	SN12	6	31	2002	Invalid						
13	SN13	7	14	2000	7	15	2000				
14	SN14	7	14	1996	7	15	1996				
15	SN15	7	14	2002	7	15	2002				
16	SN16	7	29	2000	7	30	2000				
17	SN17	7	29	1996	7	30	1996				
18	SN18	7	29	2002	7	30	2002				
19	SN19	7	30	2000	7	31	2000				

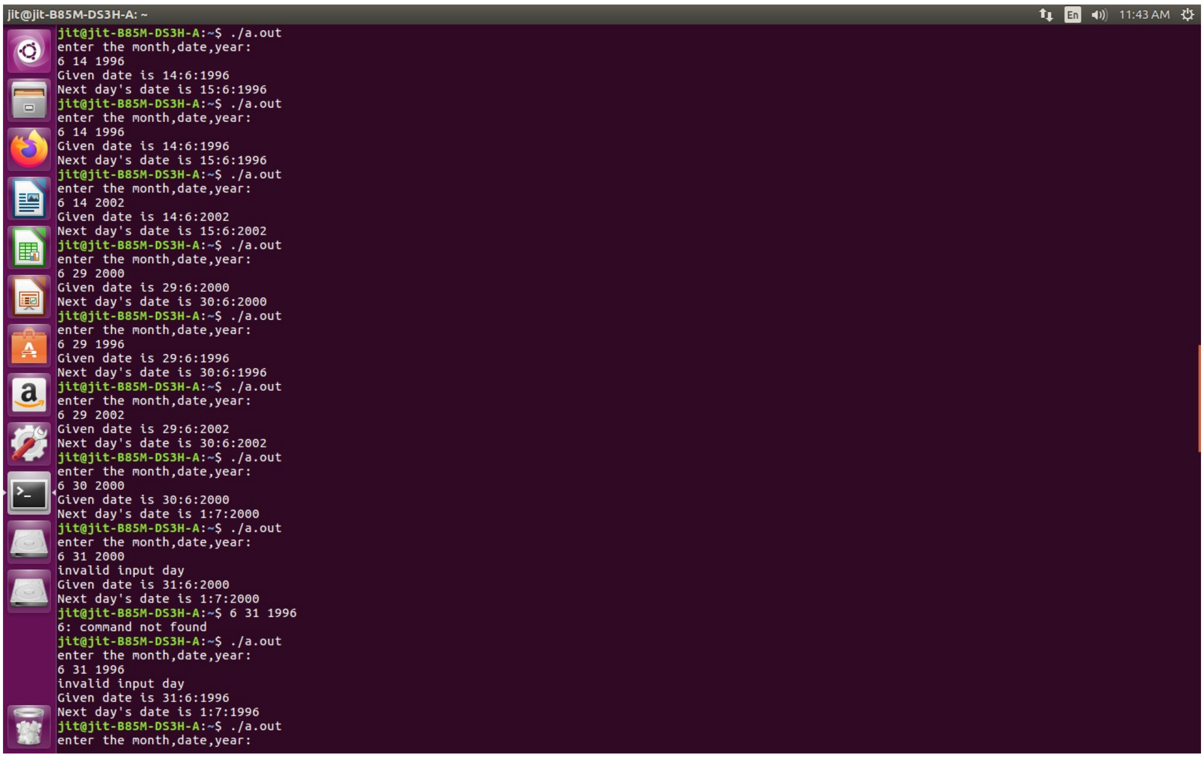
20	SN20	7	30	1996	7	31	1996				
21	SN21	7	30	2002	7	31	2002				
22	SN22	7	31	2000	8	1	2000				
23	SN23	7	31	1996	8	1	1996				
24	SN24	7	31	2002	8	1	2000				
25	SN25	2	14	2000	2	15	2000				
26.	SN26	2	14	1996	2	15	1996				
27	SN27	2	14	2002	2	15	2002				
28	SN28	2	29	2000	Invalid						
29	SN29	2	29	1996	3	1	1996				
30	SN30	2	29	2002	Invalid						
31	SN31	2	30	2000	Invalid						
32	SN32	2	30	1996	Invalid						
33	SN33	2	30	2002	Invalid						
34	SN34	2	31	2000	Invalid						
35	SN35	2	31	1996	Invalid						
36	SN36	2	31	2002	Invalid						

Output Snapshots:

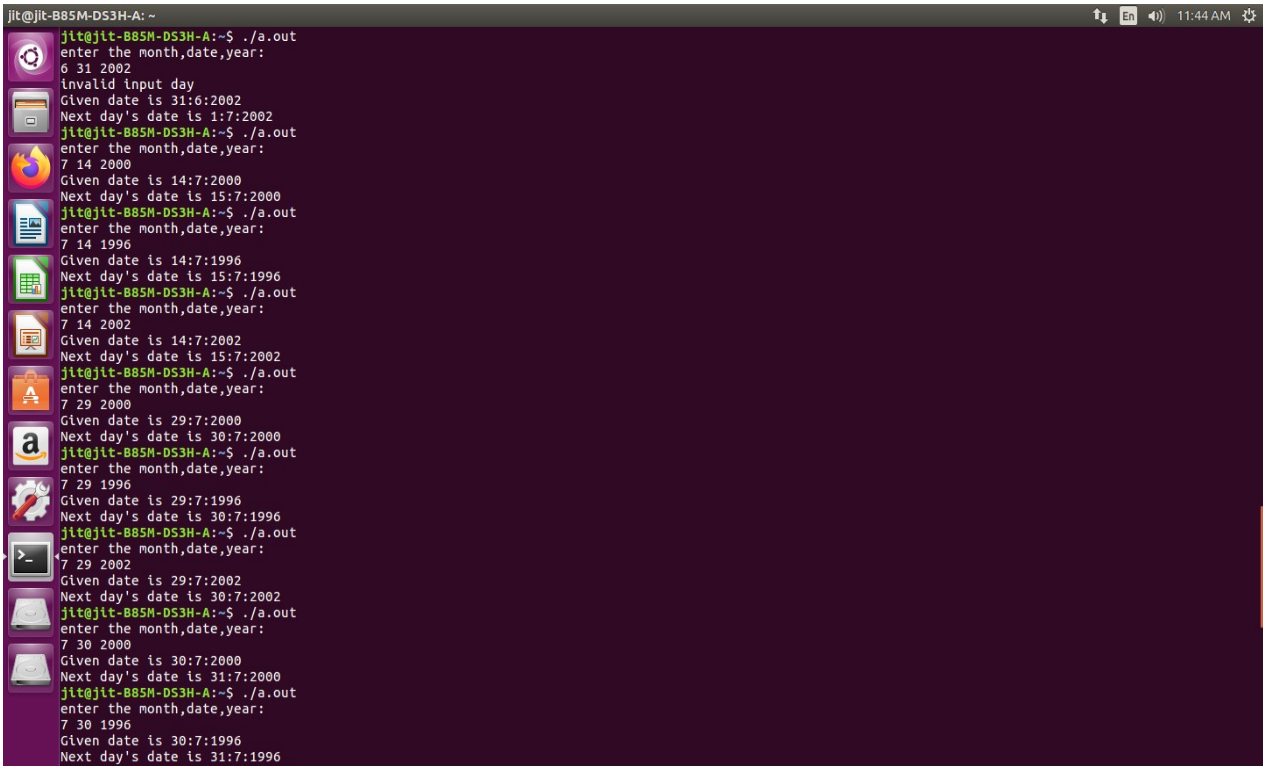




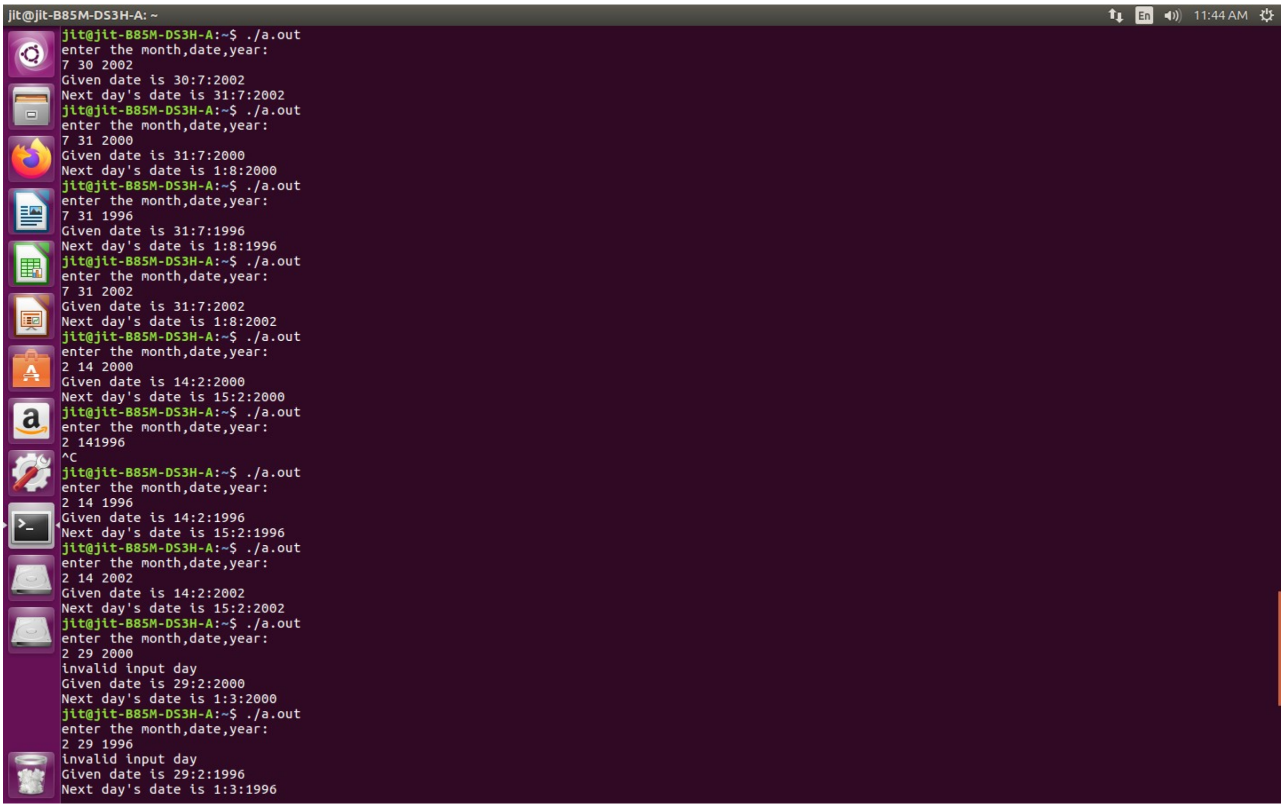
```
jit@jit-B85M-DS3H-A: ~
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 -1 1811
invalid input year
invalid input day
Given date is -1:6:1811
Next day's date is 0:6:1811
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
-1 15 1811
invalid input year
invalid input day
invalid input monthGiven date is 15:-1:1811
Next day's date is 1:0:1811
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
-1 -1 1811
invalid input year
invalid input day
invalid input monthGiven date is -1:-1:1811
Next day's date is 0:-1:1811
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 2000
Given date is 14:6:2000
Next day's date is 15:6:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 29 1996
Given date is 29:7:1996
Next day's date is 30:7:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 30 2002
invalid input day
Given date is 30:2:2002
Next day's date is 1:3:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 31 2000
invalid input day
Given date is 31:6:2000
Next day's date is 1:7:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 2000
Given date is 14:6:2000
Next day's date is 15:6:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 1996
```



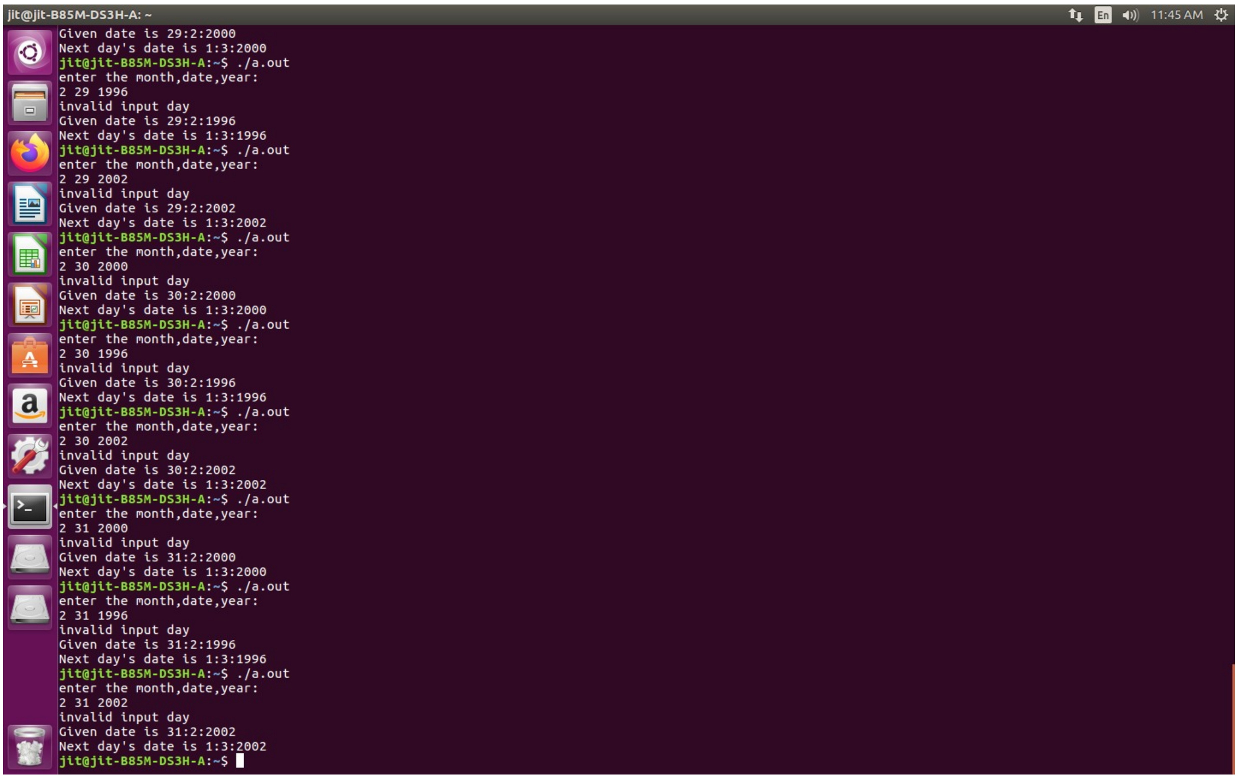
```
jit@jit-B85M-DS3H-A: ~
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 1996
Given date is 14:6:1996
Next day's date is 15:6:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 1996
Given date is 14:6:1996
Next day's date is 15:6:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 14 2002
Given date is 14:6:2002
Next day's date is 15:6:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 29 2000
Given date is 29:6:2000
Next day's date is 30:6:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 29 1996
Given date is 29:6:1996
Next day's date is 30:6:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 29 2002
Given date is 29:6:2002
Next day's date is 30:6:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 30 2000
Given date is 30:6:2000
Next day's date is 1:7:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 31 2000
invalid input day
Given date is 31:6:2000
Next day's date is 1:7:2000
jit@jit-B85M-DS3H-A:~$ 6 31 1996
6: command not found
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 31 1996
invalid input day
Given date is 31:6:1996
Next day's date is 1:7:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
```



```
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
6 31 2002
invalid input day
Given date is 31:6:2002
Next day's date is 1:7:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 14 2000
Given date is 14:7:2000
Next day's date is 15:7:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 14 1996
Given date is 14:7:1996
Next day's date is 15:7:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 14 2002
Given date is 14:7:2002
Next day's date is 15:7:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 29 2000
Given date is 29:7:2000
Next day's date is 30:7:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 29 1996
Given date is 29:7:1996
Next day's date is 30:7:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 29 2002
Given date is 29:7:2002
Next day's date is 30:7:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 30 2000
Given date is 30:7:2000
Next day's date is 31:7:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 30 1996
Given date is 30:7:1996
Next day's date is 31:7:1996
```



```
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 30 2002
Given date is 30:7:2002
Next day's date is 31:7:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 31 2000
Given date is 31:7:2000
Next day's date is 1:8:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 31 1996
Given date is 31:7:1996
Next day's date is 1:8:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
7 31 2002
Given date is 31:7:2002
Next day's date is 1:8:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 14 2000
Given date is 14:2:2000
Next day's date is 15:2:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 141996
^C
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 14 1996
Given date is 14:2:1996
Next day's date is 15:2:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 14 2002
Given date is 14:2:2002
Next day's date is 15:2:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 29 2000
invalid input day
Given date is 29:2:2000
Next day's date is 1:3:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 29 1996
invalid input day
Given date is 29:2:1996
Next day's date is 1:3:1996
```

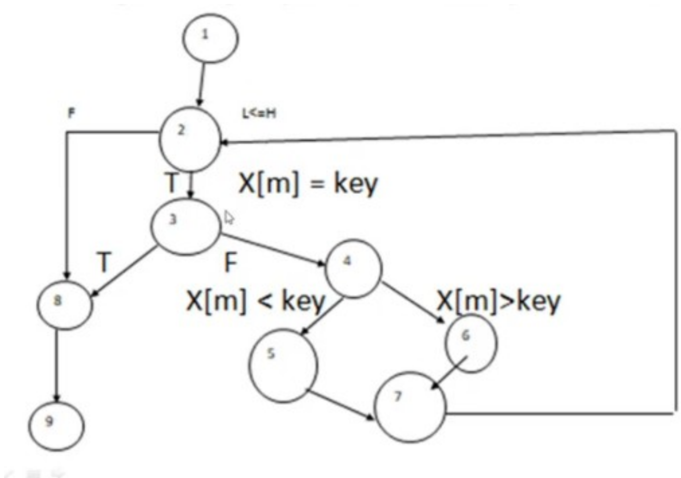

jit@jit-B85M-DS3H-A: ~
11:45 AM
Given date is 29:2:2000
Next day's date is 1:3:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 29 1996
invalid input day
Given date is 29:2:1996
Next day's date is 1:3:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 29 2002
invalid input day
Given date is 29:2:2002
Next day's date is 1:3:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 30 2000
invalid input day
Given date is 30:2:2000
Next day's date is 1:3:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 30 1996
invalid input day
Given date is 30:2:1996
Next day's date is 1:3:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 30 2002
invalid input day
Given date is 30:2:2002
Next day's date is 1:3:2002
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 31 2000
invalid input day
Given date is 31:2:2000
Next day's date is 1:3:2000
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 31 1996
invalid input day
Given date is 31:2:1996
Next day's date is 1:3:1996
jit@jit-B85M-DS3H-A:~$ ./a.out
enter the month,date,year:
2 31 2002
invalid input day
Given date is 31:2:2002
Next day's date is 1:3:2002
jit@jit-B85M-DS3H-A:~$

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.

BINARY SEARCH

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

Program Graph for Binary Search



Independent Paths:

#Edges=11, #Nodes=9, #P=1

V(G)= E-N+2P = 11-9+2 = **4**

P1: 1-**2**-3-8-9

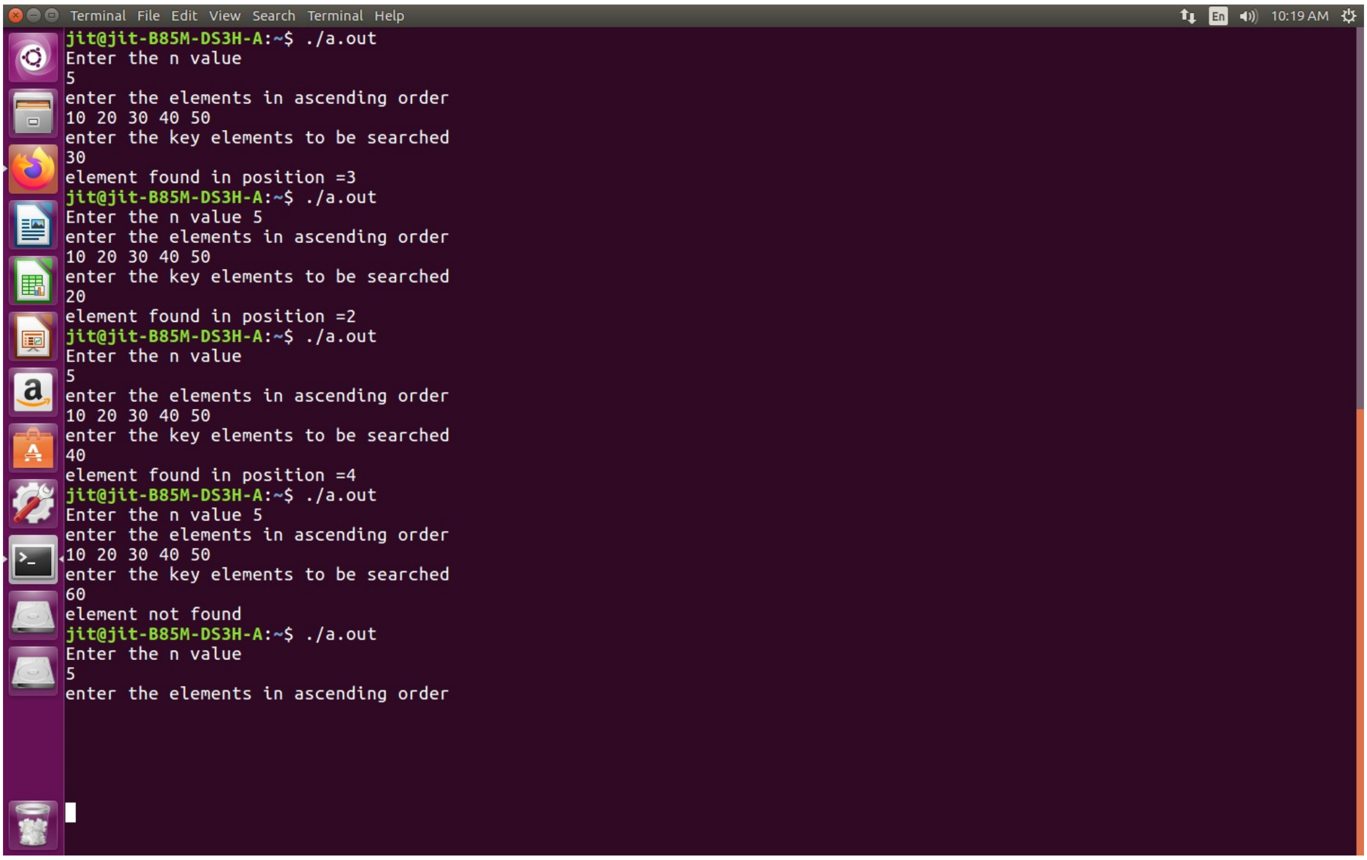
P2: 1-**2**-3-4-5-7-**2**

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

P4: 1-**2**-8-9

Paths	Inputs X[ ]	Inputs key	Expected Outputs	Remarks
P1: 1-2-3-8-9	{10,20,30,40,50}	30	Success	Key X[] and Key==X[mid]
P2: 1-2-3-4-5-7-2	{10,20,30,40,50}	20	Repeat and Success	Key < X[mid] Search 1st Half
P3: 1-2-3-4-6-7-2	{10,20,30,40,50}	40	Repeat and Success	Key > X[mid] Search 2nd Half
P4: 1-2-8-9	{10,20,30,40,50}	60	Repeat and Failure	Key X[]
P4: 1-2-8-9	Empty	Any Key	Failure	Empty List

Output snapshot:



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.

QUICK SORT

```
#include<stdio.h>
A. 1 void quicksort(int x[10],int first,int last)
B.      2 {
        3 int temp,pivot,i,j;
C. 4 if(first<last)
D.      5 {
        6 pivot=first;
        7 i=first;
        8 j=last;
E. 9 while(i<j)
F. 10 {
G. 11 while(x[i]<=x[pivot] && i<last)
H. 12 i++;
I. 13 while(x[j]>x[pivot])
J. 14 j--;
K. 15 if(i<j)
L.      16 {
        17 temp=x[i];
        18 x[i]=x[j];
        19 x[j]=temp;
        20 }
M. 21 }
N.      22 temp=x[pivot];
        23 x[pivot]=x[j];
        24 x[j]=temp;
        25 quicksort(x,first,j-1);
P. 26 quicksort(x,j+1,last);
Q. 27 }
O. 28 }

int main()
{
int a[20],i,key,n;
printf("enter the size of the array max of 20 elements");
scanf("%d",&n);
if(n>0)
{
printf("enter the elements of the array");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
quicksort(a,0,n-1);
```

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

}
```

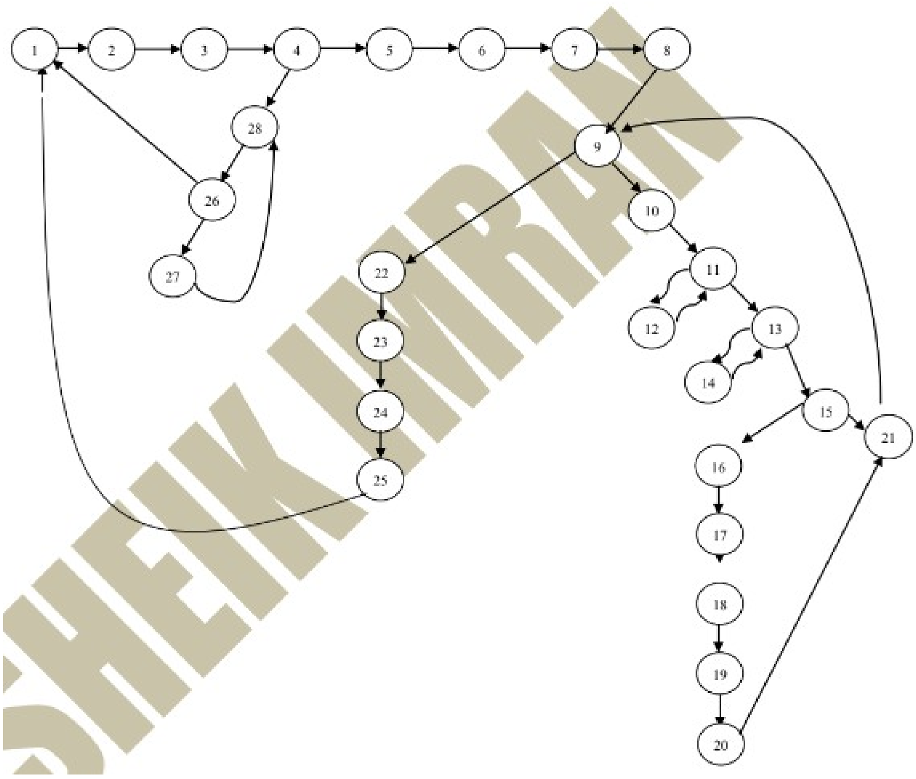
Cyclomatic Complexity

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

V(G) =e-n+p (for closed closed graph)

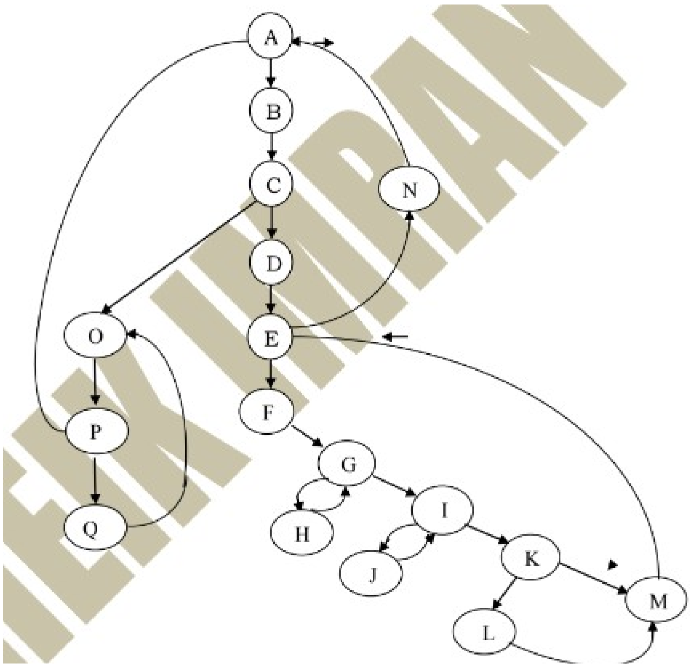
Where,
e is number of edges in DD-Path graph.
n is number of nodes in DD-Path graph.
p is number of regions connected.(always 1)
Number of linearly independent paths (Test cases) for a given graph G = 23-17+(1)
= 6+1
= 7 Test cases

Program graph



DD path graph

NODES	DDPATHS
1	A
2-3	B
4	C
5-8	D
9	E
10	F
11	G
12	H
13	I
14	J
15	K
16-20	L
21	M
22-25	N
26	P
27	Q
28	O



McCabe's Basis path method

Considering DD-Path graph of the program, first we need to find Baseline path. A baseline path consists of maximum number of decision nodes. Using Baseline path we start flipping each decision node for finding new paths.

Considering Quick Sort program
Considering DD-Path graph of Quick sort function, function starts at node A and Ends at node O. First, Base Line path is formed by considering all decision nodes as shown below.
Baseline Path: A B C D E F G I K M E N A B C O P A B C O.

Nodes which are bold and large are decision nodes. Now start flipping each decision node.

Flipping at C : A B C O.
Flipping at E : A B C D E N A B C O.
Flipping at G : A B C D E F G H G I K M E N A B C O.
Flipping at I : A B C D E F G I J I K M E N A B C O.
Flipping at K : A B C D E F G I K L M E N A B C O.
Flipping at P : A B C D E F G I K L M E N A B C O P A B C O.

Test Cases for Quick Sort Program

Test Cases	Description	Number of elements (n)	Array Elements	Comment
TC1	Enter the basis path consisting of all decision nodes ABCDEFGIKMENABCOPABCO.	----	Infeasible because path from G to I means no elements in array.	Invalid
TC2	Enter the basis path consisting of all decision nodes ABCO.	1	{9}	Valid
TC3	Enter the basis path consisting of all decision nodes ABCDENABCO.	----	Path C to D indicates if(first<last) is condition is true. So at first iteration while(x[i]<=x[pivot]&&i<last) condition also	Invalid

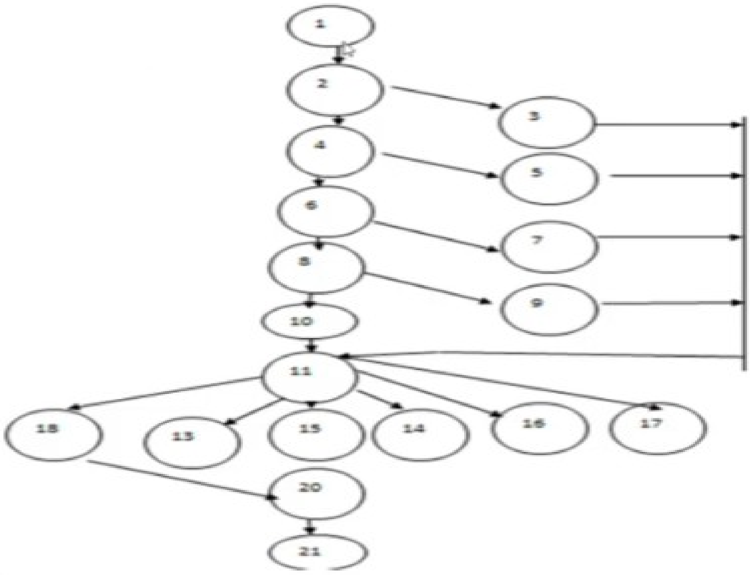
			should be true and path E to F should be present. But we have EN so Infeasible	
TC4	Enter the basis path consisting of all decision nodes ABCDEFGHGIKMENABCO.	2	{5,4 }	Valid
TC5	Enter the basis path consisting of all decision nodes ABCDEFGIJIKMENABCO.	----	Infeasible because path from G to I means no elements in array.	Invalid
TC6	Enter the basis path consisting of all decision nodes ABCDEFGIKLMENABCO.	----	Infeasible because path from G to I means no elements in array.	Invalid
TC7	Enter the basis path consisting of all decision nodes ABCDEFGIKLMENABCOPABCO.	----	Infeasible because path from G to I means no elements in array.	Invalid

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

Program

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

Program Graph



Independent Paths:

#Edges=22, #Nodes=19, #P=1

V(G)= E-N+2P = 22-19+2 = **05**

P1: 1-2-4-6-8-10-11-17-18-21 E Grade
P2: 1-2-4-6-8-9-11-16-18-21 D Grade
P3: 1-2-4-6-7-11-15-18-21 C Grade
P4: 1-2-4-5-11-14-18-21 B Grade
P5: 1-2-3-11-13-18-21 A Grade

Paths	Input(per)	Expected Output	Remarks
P1: 1-2-4-6-8-10-11-17-18-21	<60	E Grade	Pass
P2: 1-2-4-6-8-9-11-16-18-21	60 to 69	D Grade	Pass
P3: 1-2-4-6-7-11-15-18-21	70 to 79	C Grade	Pass
P4: 1-2-4-5-11-14-18-21	80 to 89	B Grade	Pass
P5: 1-2-3-11-13-18-21	>=90	A Grade	Pass