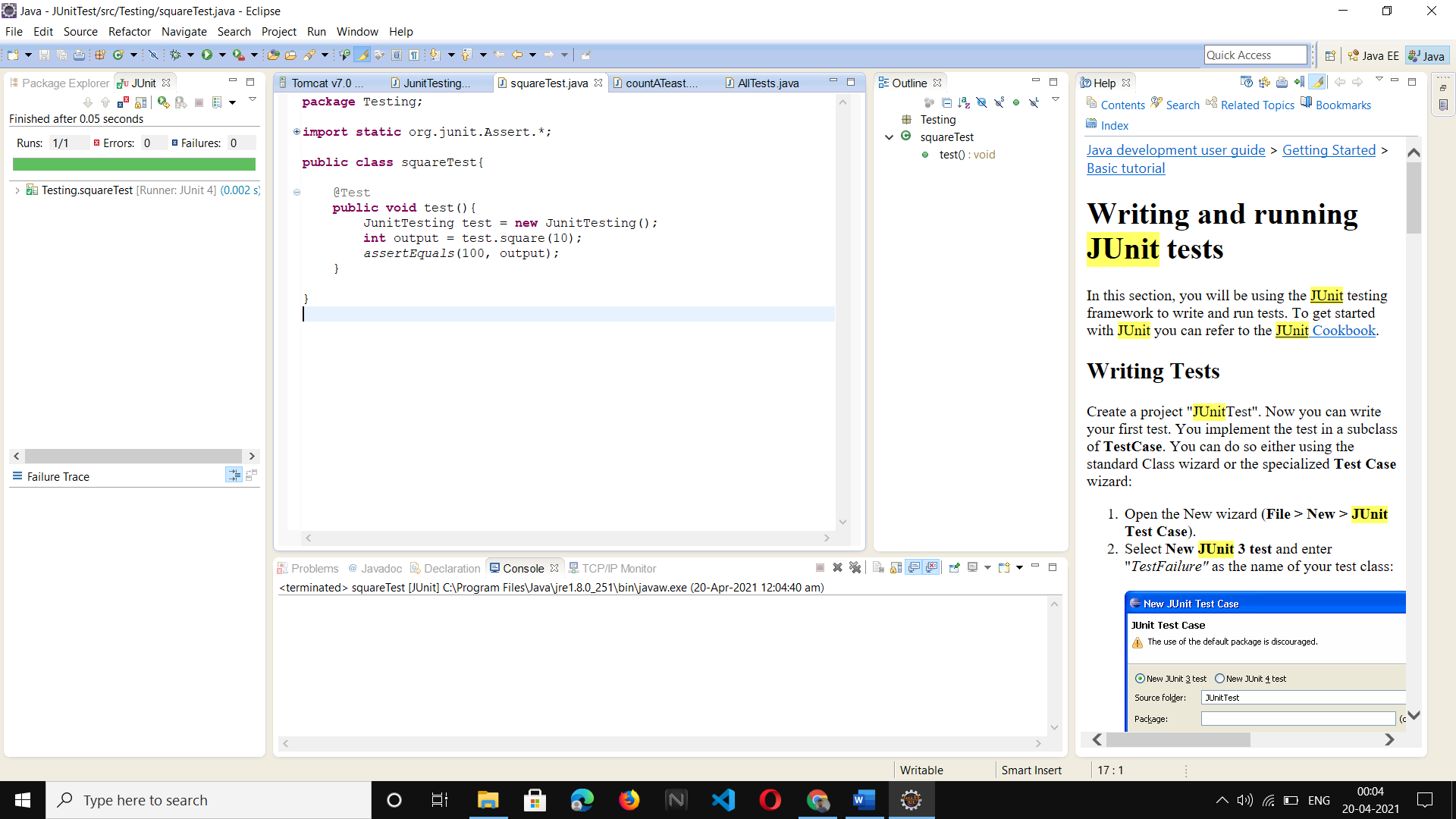
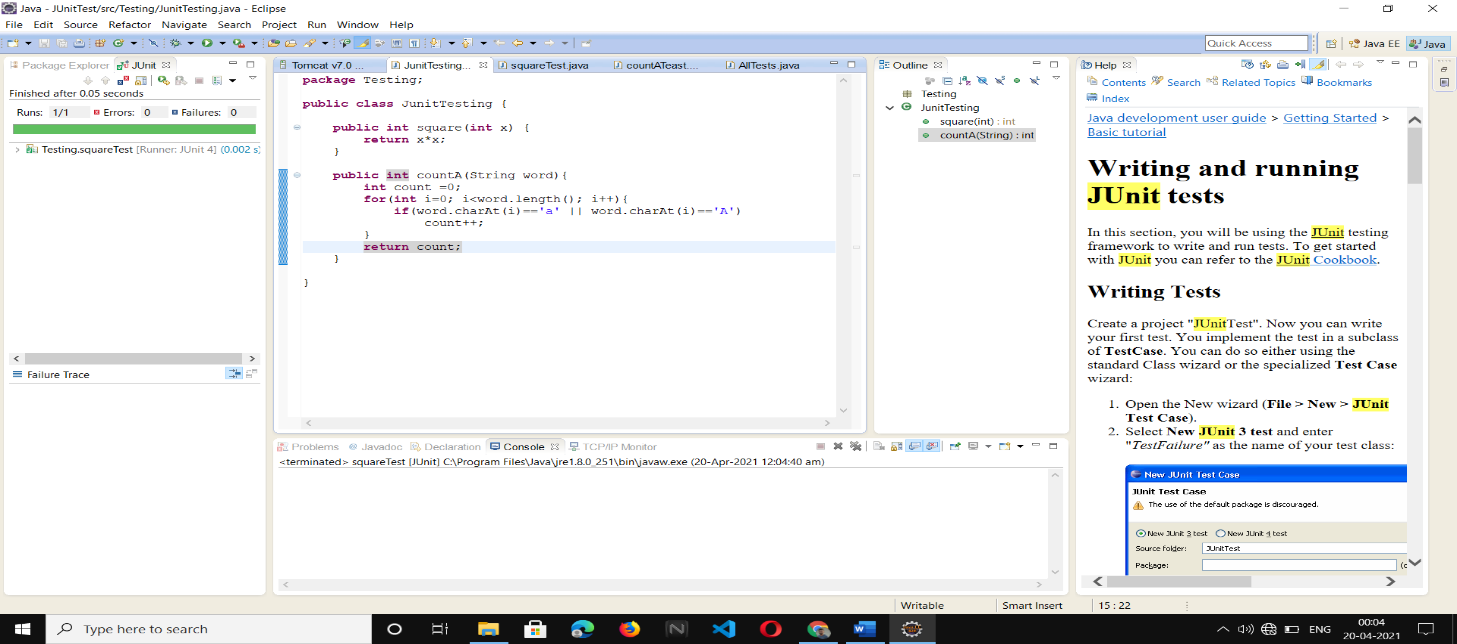
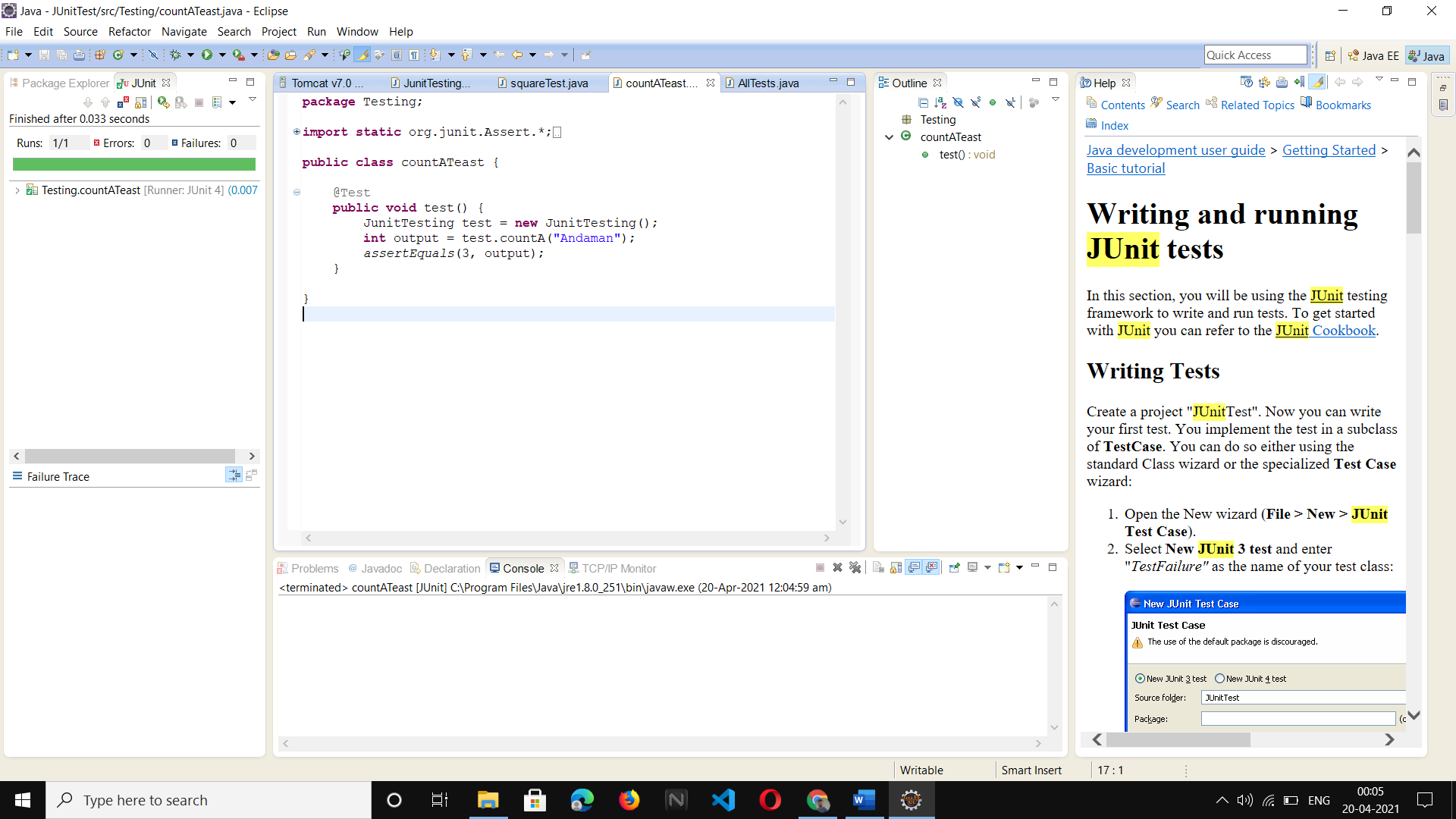
Name- Aman Jain

USN- 1BM18IS146

1.) Junit







2.) Boundary value analysis program

Design and develop a program in a language of your choice to solve the triangle problem defined as follows: Accept three integers which are supposed to be the three sides of triangle and determine if the three values represent an equilateral triangle, isosceles triangle, scalene triangle, or they do not form a triangle at all. Derive test cases for your program based on boundary value analysis, execute the test cases and discuss the results

#include<stdio.h>

int main()

{

int a,b,c,c1,c2,c3;

char istriangle;

do {

printf("\nenter 3 integers which are sides of triangle\n");

scanf("%d%d%d",&a,&b,&c);

printf("\na=%d\tb=%d\tc=%d",a,b,c);

c1 = a>=1 && a<=10;

c2= b>=1 && b<=10;

c3= c>=1 && c<=10;

if (!c1)

printf("\n the value of a=%d is not the range of permitted value", a);

if (!c2)

printf("\nthe value of b=%d is not the range of permitted value", b);

if (!c3)

printf("\nthe value of c=%d is not the range of permitted value", c);

} while (! (c1 && c2 && c3));

// to check is it a triangle or not

if( a<b+c && b<a+c && c<a+b )

istriangle='y';

else

istriangle ='n';

if (istriangle=='y')

if ((a==b) && (b==c))

printf("equilateral triangle\n");

else if ((a!=b) && (a!=c) && (b!=c))

printf("scalene triangle\n");

else

printf("isosceles triangle\n");

else

printf("Not a triangle\n");

return 0

}

Test Case Name: Boundary Value Analysis for triangle problem

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

Pre-condition: 1 ≤ a ≤ 10, 1 ≤ b ≤ 10 and 1 ≤ c ≤ 10 and a < b + c, b < a + c and c < a + b

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

Triangle Problem -Boundary value Test cases for input data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Case id** | **Description** | **Input data** | | | **Expected output** | **Actual output** | **Comments** |
| a | b | c |
| 1 | Enter the min value for a, b and c | 1 | 1 | 1 | Equilateral triangle | Equilateral triangle | Outputs Matched, Test passed |
| 2 | Enter min value for 2 items and min+1 for any one item | 1 | 1 | 2 | Can’t form a triangle | Can’t form a triangle | Pass |
| 3 | Enter min value for 2 items and min+1 for any one item | 1 | 2 | 1 | Can’t form a triangle | Can’t form a triangle | Pass |
| 4 | Enter min value for 2 items and min+1 for any one item | 2 | 1 | 1 | Can’t form a triangle | Can’t form a triangle | Pass |
| 5 | Enter normal value for 2 items and 1 item is min value | 5 | 5 | 1 | Isosceles triangle | Isosceles triangle | Pass |
| 6 | Enter normal value for 2 items and 1 item is min value | 5 | 1 | 5 | Isosceles triangle | Isosceles triangle | Pass |
| 7 | Enter normal value for 2 items and 1 item is min value | 1 | 5 | 5 | Isosceles triangle | Isosceles triangle | Pass |
| 8 | Enter the normal value for a, b, and c | 5 | 5 | 5 | Equilateral triangle | Equilateral triangle | Pass |
| 9 | Enter the normal value for 2 items and 1 item is max value | 5 | 5 | 10 | Not a triangle | Not a triangle | Pass |
| 10 | Enter the normal value for 2 items and 1 item is max value | 5 | 10 | 5 | Not a triangle | Not a triangle | Pass |
| 11 | Enter the normal value for 2 items and 1 item is max value | 10 | 5 | 5 | Not a triangle | Not a triangle | Pass |
| 12 | Enter the max value for 2 items and max-1 for any one item | 10 | 10 | 9 | Isosceles triangle | Isosceles triangle | Pass |
| 13 | Enter the max value for 2 items and max-1 for any one item | 10 | 9 | 10 | Isosceles triangle | Isosceles triangle | Pass |
| 14 | Enter the max value for 2 items and max-1 for any one item | 9 | 10 | 10 | Isosceles triangle | Isosceles triangle | Pass |
| 15 | Enter the values of sides such that it forms a scalene triangle | 2 | 5 | 4 | Scalene Triangle | Scalene Triangle | Pass |
| 16 | Enter max value for a, b and c | 10 | 10 | 10 | Equilateral triangle | Equilateral triangle | Pass |

3.) Design, develop, code and run the program in any suitable language to solve the commission problem. Analyse it from the perspective of boundary value testing, derive different test cases, execute these test cases and discuss the test results

**The Commission Problem**

A rifle salesperson in the former Arizona Territory sold rifle locks, stocks, and barrels made by a gunsmith in Missouri. Locks cost $45, stocks cost $30, and barrels cost $25. The salesperson had to sell at least one lock, one stock, and one barrel (but not necessarily 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 salesperson sent a telegram to the Missouri gunsmith with the number of locks, stocks, and barrels sold in that town. At the end of a 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: 10% on sales up to (and including) $1000, 15% on the next $800, and 20% on any sales in excess of $1800.

**Test Cases for the Commission Problem**

Instead of going through 125 test cases again, we will look at some more interesting test cases for the commission problem. This time, we will look at boundary values derived from the output range, especially near the threshold points of $1000 and $1800 where the commission percentage changes.

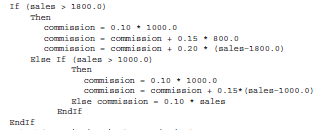
Test data : price Rs for lock - 45.0 , stock - 30.0 and barrel - 25.0

sales = total lock \* lock price + total stock \* stock price + total barrel \* barrel price

commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any sales in excess of 1800

Pre-condition : lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

Code snippet:



Code:

**#include <stdio.h>**

**int main()**

**{**

**float stocks, locks, barrels, sales=0.0, comission=0.0;**

**float barrelp= 25.0;**

**float stockp = 30.0;**

**float lockp = 45.0;**

**printf("\n Enter the number of locks: ");**

**scanf("%f", &locks);**

**printf("\n Enter the number of stocks: ");**

**scanf("%f", &stocks);**

**printf("\n Enter the number of barrels: ");**

**scanf("%f", &barrels);**

**sales = locks \* lockp + stocks \* stockp + barrels \* barrelp;**

**printf("\nSales = %d", (int)sales);**

**if(sales> 1800.0)**

**{**

**comission = 0.10\*1000;**

**comission = comission + 0.15\*800.0;**

**comission = comission + 0.20\*(sales-1800.0);**

**}**

**else if(sales >1000.0)**

**{**

**comission = 0.10 \* 1000.0;**

**comission = comission + 0.15\*(sales-1000.0);**

**}**

**else**

**comission = 0.10 \* sales;**

**printf("\nComission = %f", comission);**

**return 0;**

**}**

**Checking boundary value for locks, stocks and barrels and commission**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Input data** | | | **Expected output** | | **Actual output** | |  |  | |
| **Case id** | **Description** | **Total Locks** | **Total stocks** | **Total barrels** | **Sales** | **Commission** | **Sales** | **Commission** | **Status** | **Comment** | |
| **1** | Enter the min value for locks, stocks and barrels | 1 | 1 | 1 | 100 | 10 | 100 | 10.00 | Pass | Output minimum |
| **2** | Enter the min value for 2 items and min +1 for  any one item | 1 | 1 | 2 | 125 | 12.5 | 125 | 12.50 | Pass | Output minimum + |
| **3** | 1 | 2 | 1 | 130 | 13 | 130 | 13.00 | Pass | Output minimum + |
| **4** | 2 | 1 | 1 | 145 | 14.5 | 145 | 14.50 | Pass | Output minimum + |
| **5** | Enter the value sales approximately mid value  between 100 to 1000 | 5 | 5 | 5 | 500 | 50 | 500 | 50.00 | Pass | Midpoint |
| **6** | Enter the values to calculate the commission for  sales nearly less than 1000 | 10 | 10 | 9 | 975 | 97.5 | 975 | 97.50 | Pass | Border point - |
| **7** | 10 | 9 | 10 | 970 | 97.5 | 970 | 97.00 | Fail | Border point - |
| **8** | 9 | 10 | 10 | 955 | 95.5 | 955 | 95.50 | Pass | Border point - |
| **9** | Enter the values sales exactly equal to 1000 | 10 | 10 | 10 | 1000 | 100 | 1000 | 100.00 | Pass | Border point |
| **10** | Enter the values to calculate the commission for  sales nearly greater than 1000 | 10 | 10 | 11 | 1025 | 103.75 | 1025 | 103.75 | Pass | Border point + |
| **11** | 10 | 11 | 10 | 1030 | 104.5 | 1030 | 104.50 | Pass | Border point + |
| **12** | 11 | 10 | 10 | 1045 | 106.75 | 1042 | 106.75 | Pass | Border point + |
| **13** | Enter the value sales approximately mid value between 1000 to 1800 | 14 | 14 | 14 | 1400 | 160 | 1400 | 160.00 | Pass | Midpoint |
| **14** | Enter the values to calculate the commission for  sales nearly less than 1800 | 18 | 18 | 17 | 1775 | 216.25 | 1775 | 216.25 | Pass | Border point - |
| **15** | 18 | 17 | 18 | 1770 | 215.5 | 1770 | 215.50 | Pass | Border point - |
| **16** | 17 | 18 | 18 | 1755 | 213.25 | 1755 | 213.25 | Pass | Border point - |
| **17** | Enter the values sales exactly equal to 1800 | 18 | 18 | 18 | 1800 | 220 | 1800 | 220.00 | Pass | Border point |
| **18** | Enter the values to calculate the commission for  sales nearly greater than 1800 | 18 | 18 | 19 | 1825 | 225 | 1825 | 225.00 | Pass | Border point + |
| **19** | 18 | 19 | 18 | 1845 | 229 | 1830 | 226.00 | Fail | Border point + |
| **20** | 19 | 18 | 18 | 1845 | 229 | 1845 | 229.00 | Pass | Border point + |
| **21** | Enter the values normal value for lock, stock and  barrel | 48 | 48 | 48 | 4800 | 820 | 4800 | 820.00 | Pass | Midpoint |
| **22** | Enter the max value for 2 items and max - 1 for  any one item | 70 | 80 | 89 | 7775 | 1415 | 7775 | 1415.00 | Pass | Output maximum - |
| **23** | 70 | 79 | 90 | 7770 | 1414 | 7770 | 1414.00 | Pass | Output maximum - |
| **24** | 69 | 80 | 90 | 7755 | 1411 | 7755 | 1411.00 | Pass | Output maximum - |
| **25** | Enter the max value for locks, stocks and barrels | 70 | 80 | 90 | 7800 | 1420 | 7800 | 1420.00 | Pass | Output maximum |

**Output Special Value Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Input data** | | | **Expected output** | | **Actual output** | |  |  | |
| **Case id** | **Description** | **Total Locks** | **Total stocks** | **Total barrels** | **Sales** | **Commission** | **Sales** | **Commission** | **Status** | **Comment** | |
| **1** | Enter the random values such that to calculate  commission for sales nearly less than 1000 | 11 | 10 | 8 | 995 | 99.5 | 995 | 99.50 | Pass | Border point - |
| **2** | Enter the random values such that to calculate  commission for sales nearly greater than 1000 | 10 | 11 | 9 | 1005 | 100.75 | 1005 | 100.75 | Pass | Border point + |
| **3** | Enter the random values such that to calculate  commission for sales nearly less than 1800 | 18 | 17 | 19 | 1795 | 219.25 | 1795 | 219.25 | Pass | Border point - |
| **4** | Enter the random values such that to calculate  commission for sales nearly greater than 1800 | 18 | 19 | 17 | 1805 | 221 | 1805 | 221.00 | Pass | Border point + |

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 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.

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

Pre-condition : 1 ≤ a ≤ 10 , 1 ≤ b ≤ 10 and 1 ≤ c ≤ 10 and a < b + c , b < a + c and c < a + b

Brief Description : Check whether given value for a equilateral, isosceles , Scalene triangle or Not a triangle

Code :

#include<stdio.h>

int main()

{

int a,b,c,c1,c2,c3;

char istriangle;

do {

printf("\nEnter 3 integers which are sides of triangle\n");

scanf("%d%d%d",&a,&b,&c);

printf("\na=%d\tb=%d\tc=%d",a,b,c);

c1 = a>=1 && a<=10;

c2= b>=1 && b<=10;

c3= c>=1 && c<=10;

if (!c1)

printf("\nThe value of a=%d is not the range of permitted value", a);

if (!c2)

printf("\nThe value of b=%d is not the range of permitted value", b);

if (!c3)

printf("\nThe value of c=%d is not the range of permitted value", c);

} while (! (c1 && c2 && c3));

// to check is it a triangle or not

if( a<b+c && b<a+c && c<a+b )

istriangle='y';

else

istriangle ='n';

if (istriangle=='y')

if ((a==b) && (b==c))

printf("Equilateral triangle\n");

else if ((a!=b) && (a!=c) && (b!=c))

printf("Scalene triangle\n");

else

printf("Isosceles triangle\n");

else

printf("Can't form a triangle\n");

return 0;}

**Triangle Problem -Equivalence Class Test cases for input data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Weak Normal Equivalence Class Testing** | | | | | | | |
| **Case id** | **Description** | **Input data** | | | **Expected output** | **Actual output** | **Comments** |
| a | b | c |
| 1 | Enter value for a , b and c | 5 | 5 | 5 | Equilateral triangle | Equilateral triangle | Pass |
| 2 | Enter value for a , b and c | 2 | 2 | 3 | Isosceles triangle | Isosceles triangle | Pass |
| 3 | Enter value for a , b and c | 3 | 4 | 5 | Scalene triangle | Scalene triangle | Pass |
| 4 | Enter value for a , b and c | 4 | 1 | 2 | Not a triangle | Not a triangle | Pass |
| **Weak Robust Equivalence Class Testing** | | | | | | | |
| 5 | Enter one invalid input and two valid value for a, b and c | -1 | 5 | 5 | Value of a is not in the range of permitted values | Value of a is not in the range of permitted values | Pass |
| 6 | Enter one invalid input and two valid value for a, b and c | 5 | -1 | 5 | Value of b is not in the range of permitted values | Value of b is not in the range of permitted values | Pass |
| 7 | Enter one invalid input and two valid value for a, b and c | 5 | 5 | -1 | Value of c is not in the range of permitted values | Value of c is not in the range of permitted values | Pass |
| 8 | Enter one invalid input and two valid value for a, b and c | 11 | 5 | 5 | Value of a is not in the range of permitted values | Value of a is not in the range of permitted values | Pass |
| 9 | Enter one invalid input and two valid value for a, b and c | 5 | 11 | 5 | Value of b is not in the range of permitted values | Value of b is not in the range of permitted values | Pass |
| 10 | Enter one invalid input and two valid value for a, b and c | 5 | 5 | 11 | Value of c is not in the range of permitted values | Value of c is not in the range of permitted values | Pass |
| **Strong Robust Equivalence Class Testing** | | | | | | | |
| 11 | Enter one invalid input and two valid value for a , b and c | -1 | 5 | 5 | Value of a is not in the range of permitted values | Value of a is not in the range of permitted values | Pass |
| 12 | Enter one invalid input and two valid value for a , b and c | 5 | -1 | 5 | Value of b is not in the range of permitted values | Value of b is not in the range of permitted values | Pass |
| 13 | Enter one invalid input and two valid value for a , b and c | 5 | 5 | -1 | Value of c is not in the range of permitted values | Value of c is not in the range of permitted values | Pass |
| 14 | Enter two invalid input and one valid value for a , b and c | -1 | -1 | 5 | Value of a,b is not in the range of permitted values | Value of a,b is not in the range of permitted values | Pass |
| 15 | Enter two invalid input and one valid value for a , b and c | 5 | -1 | -1 | Value of b,c is not in the range of permitted values | Value of b,c is not in the range of permitted values | Pass |
| 16 | Enter two invalid input and one valid value for a , b and c | -1 | 5 | -1 | Value of a,c is not in the range of permitted values | Value of a,c is not in the range of permitted values | Pass |
| 17 | Enter all invalid inputs | -1 | -1 | -1 | Value of a,b,c is not in the range of permitted values | Value of a,b,c is not in the range of permitted values | Pass |

**Equivalence classes:**

D1 = {<a, b, c>: a = b = c}

D2 = {<a, b, c>: a = b, a ≠ c}

D3 = {<a, b, c>: a = c, a ≠ b}

D4 = {<a, b, c>: b = c, a ≠ b}

D5 = {<a, b, c>: a ≠ b, a ≠ c, b ≠ c}

D6 = {<a, b, c>: a ≥ b + c}

D7 = {<a, b, c>: b ≥ a + c}

D8 = {<a, b, c>: c ≥ a + b}

5.) Design, develop, code and run the program in any suitable language to implement the Next Date function. Analyse it from the perspective of equivalence class value testing, derive different test cases, execute these test cases and discuss the test results.

Code snippets:

if(day<1 || day>31)

{

printf("value of day, not in the range 1...31\n");

flag='n';

}

if(month<1 || month>12)

{

printf("value of month, not in the range 1....12\n");

flag='n';

}

else if(check(day,month))

{

printf("value of day, not in the range day<=30");

flag='n';

}

………….

………..

if(year<=1812 || year>=2012)

{

printf("value of year, not in the range 1812.......2012\n");

flag='n';

}

if(month==2)

{

if(isleap(year) && day>29)

{

printf("invalid date input for leap year");

flag='n';

}

else if(!(isleap(year))&& day>28)

{

printf("invalid date input for not a leap year");

flag='n';

}

}

………..

……….

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:if(day<31)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=month+1;

}

break;

case 4:

case 6:

case 9:

case 11: if(day<30)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=month+1;

}

break;

case 12: if(day<31)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=1;

if(year==2013)

{

printf("the next day is out of boundary value of year\n");

tomm\_year=year+1;

}

else

tomm\_year=year+1;

}

break;

case 2:

if(day<28)

tomm\_day=day+1;

else if(isleap(year)&& day==28)

tomm\_day=day+1;

else if(day==28 || day==29)

{

tomm\_day=1;

tomm\_month=3;

}

break;

}

// print the next date values as day, month, year format

**Code:**

#include<stdio.h>

int check(int day,int month)

{

if((month==4||month==6||month==9 ||month==11) && day==31)

return 1;

else

return 0;

}

int isleap(int year)

{

if((year%4==0 && year%100!=0) || year%400==0)

return 1;

else

return 0;

}

int main()

{

int day,month,year,tomm\_day,tomm\_month,tomm\_year;

char flag;

do

{

flag='y';

printf("\nenter the today's date in the form of dd mm yyyy\n");

scanf("%d%d%d",&day,&month,&year);

tomm\_month=month;

tomm\_year= year;

if(day<1 || day>31)

{

printf("value of day, not in the range 1...31\n");

flag='n';

}

if(month<1 || month>12)

{

printf("value of month, not in the range 1....12\n");

flag='n';

}

else if(check(day,month))

{

printf("value of day, not in the range day<=30");

flag='n';

}

if(year<=1812 || year>2012)

{

printf("value of year, not in the range 1812.......2012\n");

flag='n';

}

if(month==2)

{

if(isleap(year) && day>29)

{

printf("invalid date input for leap year");

flag='n';

}

else if(!(isleap(year))&& day>28)

{

printf("invalid date input for not a leap year");

flag='n';

}

}

}while(flag=='n');

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:if(day<31)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=month+1;

}

break;

case 4:

case 6:

case 9:

case 11: if(day<30)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=month+1;

}

break;

case 12: if(day<31)

tomm\_day=day+1;

else

{

tomm\_day=1;

tomm\_month=1;

if(year==2013)

{

printf("the next day is out of boundary value of year\n");

tomm\_year=year+1;

}

else

tomm\_year=year+1;

}

break;

case 2:

if(day<28)

tomm\_day=day+1;

else if(isleap(year)&& day==28)

tomm\_day=day+1;

else if(day==28 || day==29)

{

tomm\_day=1;

tomm\_month=3;

}

break;

}

printf("next day is : %d %d %d",tomm\_day,tomm\_month,tomm\_year);

return 0;

}

**Valid equivalence classes are:**

M1 = {month: 1 ≤ month ≤ 12}

D1 = {day: 1 ≤ day ≤ 31}

Y1 = {year: 1812 ≤ year ≤ 2012}

**Invalid equivalence classes are:**

M2 = {month: month < 1}

M3 = {month: month > 12}

D2 = {day: day < 1}

D3 = {day: day > 31}

Y2 = {year: year < 1812}

Y3 = {year: year > 2012}

**Next date Output Equivalence Class Testing**

**Weak and Strong Normal Equivalence Class test case**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Input data | | | Expected output | | | Actual output | | |  |
| Case id | Description | Month | Day | Year | Month | Day | Year | Month | Day | Year | Comment |
| 1 | Enter the M1, D1 and Y1 valid  cases | 6 | 15 | 1912 | 6 | 16 | 1912 | 6 | 16 | 1912 | Pass |

**Weak Robust test cases**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Input data | | | Expected output | | | Actual output | | |  |
| Case id | Description | Month | Day | Year | Month | Day | Year | Month | Day | Year | Comment |
| 1 | Enter the M1, D1 and Y1 cases | 6 | 15 | 1912 | 6 | 16 | 1912 | 6 | 16 | 1912 | Pass |
| 2 | Enter the M2 , D1 and Y1 cases | -1 | 15 | 1912 | Value of month not in the range 1 ... 12 | | |  |  |  | Pass.  Actual output same as expected |
| 3 | Enter the M3 ,D1 and Y1 cases | 13 | 15 | 1912 | Value of month not in the range 1 ... 12 | | |  |  |  | Pass.  Actual output same as expected |
| 4 | Enter the M1, D2 and Y1 cases | 6 | -1 | 1912 | Value of day not in the range 1 ... 31 | | |  |  |  | Pass.  Actual output same as expected |
| 5 | Enter the M1, D3 and Y1 cases | 6 | 32 | 1912 | Value of day not in the range 1 ... 31 | | |  |  |  | Pass.  Actual output same as expected |
| 6 | Enter the M1, D1 and Y2 cases | 6 | 15 | 1811 | Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |
| 7 | Enter the M1, D1 and Y3 cases | 6 | 15 | 2013 | Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |

**Strong Robust test cases**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Input data | | | Expected output | | | Actual output | | |  |
| Case id | Description | Month | Day | Year | Month | Day | Year | Month | Day | Year | Comment |
| 1 | Enter the M2 , D1 and Y1 cases | -1 | 15 | 1912 | Value of month not in the range 1 ... 12 | | |  |  |  | Pass.  Actual output same as expected |
| 2 | Enter the M1, D2 and Y1 cases | 6 | -1 | 1912 | Value of day not in the range 1 ... 31 | | |  |  |  | Pass.  Actual output same as expected |
| 3 | Enter the M1, D1 and Y2 cases | 6 | 15 | 1811 | Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |
| 4 | Enter the M2 , D2 and Y1 cases | -1 | -1 | 1912 | Value of month not in the range 1 ... 12  Value of day not in the range 1 ... 31 | | |  |  |  | Pass.  Actual output same as expected |
| 5 | Enter the M1, D2 and Y2 cases | 6 | -1 | 1811 | Value of day not in the range 1 ... 31  Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |
| 6 | Enter the M2, D1 and Y2 cases | -1 | 15 | 1811 | Value of month not in the range 1 ... 12  Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |
| 7 | Enter the M2, D2 and Y2 cases | -1 | -1 | 1811 | Value of month not in the range 1 ... 12  Value of day not in the range 1 ... 31  Value of year not in the range 1812 ... 2012 | | |  |  |  | Pass.  Actual output same as expected |

**Some additional equivalence Boundary checking**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Input data | | | Expected output | | | | | Actual output | | |  |
| Case id | Description | Month | Day | Year | Month | | Day | | Year | Month | Day | Year | Comment |
| 1 | Enter the M1 , D1 and Y2 cases | 12 | 31 | 1811 | Value of year not in the range 1812 ... 2012 | | | | |  |  |  | Pass.  Actual output same as expected |
| 2 | Enter the M1, D1 and Y1 cases | 12 | 31 | 2012 | 1 | 1 | | 2013 | | 1 | 1 | 2013 | Pass.  Actual output same as expected |
| 3 | Enter the M1, D1 and Y3 cases | 12 | 31 | 2013 | Value of year not in the range 1812 ... 2012 | | | | |  |  |  | Pass.  Actual output same as expected |