II JAI SRI GURUDEV II

SRI ADICHUNCHANAGIRI SHIKSHANA TRUST (R.)

S.J.C. INSTITUTE OF TECHNOLOGY

CHICKBALLAPUR - 562 101.

Laboratory Certificate

This is to certify that

Mr./Ms..... NITHYA. D. M. .....

bearing USN 1SJ18IS071 *Sem.....* VI *.....Branch.....* ISE ..... *has satisfactorily completed the Practical Experiments of* [SOFTWARE TESTING] ..... *Laboratory, Prescribed by the Visvesvaraya Technological University for the year.* 2020 - 21

Mark Obtained	
Total Marks	**25**

Head of the Department

Signature of the Teacher Incharge

Date :

INDEX

Sl No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
01	Design and develop a program to solve the triangle problems.	02-03	24/04/2021	–	
02	Design, develop code and run the program to solve the commission problem.	04-08	07/05/2021	–	
03	Design and develop code and run the program to implement the Next Date function.	09-11	08/05/2021	–	
04	Design, develop a prog-ram to solve the triangle problem using Equivalence class Testing	12-14	29/05/2021	–	
05	Design, develop a program to solve the Commission problem using Equivalen-ce class testing	15-18	05/06/2021	–	

INDEX

Sl No.	Name of the Experiment	Page No.	Date of Experiment	Date of Submission	Remarks
06	Design, develop code and run the program to implement the Next Date function using Equivalence class testing.	19-24	12/06/2021	-	
07	Design, develop code and run the program to implement the triangle problem using Equivalence class testing	25-27	19/06/2021	-	
08	Design, develop code and run the program to implement the commission problem using Equivalence class testing.	28-30	19/06/21	-	
09.	Design, develop code and run the program to implement the commission problem using dataflow testing.	32-35	24/07/21	-	
10.	Design, develop to run the code to implement binary search.	36-38	24/07/21	-	

Name of Experiment: Triangle problem Date: 24/04/2021 Page No. 02

Experiment No: 01 Experiment Result: Verified

PROGRAM-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 integer numbers (a, b, c) which represents 3 sides of the triangle.

R2: Based on the input it should determine if a triangle can be formed or not.

R3: If the requirement R1 satisfied then the System should determine the type of triangle, can be.

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

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

chandra's

Algorithm

Step 1: Input a, b and c .i.e three integer values which represents three side of the triangle

Step 2: if (a < (b+c)) and (b < (a+c)) and (c < (a+b)) then do step 3
else
Print not a triangle. Do step 6

Step 3: if (a = b) and (b = c) then
Print triangle formed is equilateral. Do step 6

Step 4: if (a != b) and (a != c) and (b != c) then
print triangle formed is scalene. Do step 6.

Step 5: print triangle formed is Isosceles.

Step 6: stop.

Name of Experiment: Triangle problem Date: 24/04/2021 Page No: 03

Experiment No: 01 Experiment Result: Verified

Program Code:

```
print("Enter the three sides of the triangle")
a=int(input("Enter the value of a"))
b=int(input("Enter the value of b"))
c=int(input("Enter the value of c"))
if(a>=10) or (b>=10) or (c>=10) or (a<1) or (b<1) or
   (c<1):
    print("out of Range")
    exit()
if(a<b+c) and (b<a+c) and (c<a+b):
    if(a==b) and (b==c):
        print("Equilateral triangle").
    elif (a!=b) and (a!=c) and (b!=c):
        print("scalene triangle")
    else:
        print("Isosceles triangle")
else:
    print("Triangle cannot be formed")
```

chandra's

Test cases:-

TC ID	Test Cases description	Input data			Expected output	Actual output	Status
		a	b	c			
1	I/p of a is min	1	5	5	Isosceles triangle	Isosceles triangle	P
2	I/p of a is min+	2	5	5	Isosceles triangle	Isosceles triangle	P
3	I/p of a is nom	5	5	5	Equilateral	Equilateral	P
4	I/p of a is max-	9	5	5	Isosceles	Isosceles triangle	P
5	I/p of a is max	10	5	5	Not a triangle	Not a triangle	P
6	I/p of b is min	5	1	5	Isosceles	Isosceles triangle	P
7	I/p of b is min+	5	2	5	Isosceles	Isosceles triangle	P
8	I/p of b is max-	5	9	5	Isosceles	Isosceles triangle	P
9	I/p of b is max	5	10	5	Not a triangle	Not a triangle	P
10	I/p of c is min	5	5	1	Isosceles	Isosceles triangle	P
11	I/p of c is min+	5	5	2	Isosceles	Isosceles triangle	P
12	I/p of c is max-	5	5	99	Isosceles	Isosceles triangle	P
13	I/p of c is max	5	5	10	Not a triangle	Not a triangle	P

Test Report:

1. Number of TC Executed: 13
2. Number of Defects raised: 0
3. Number of TC's passed: 13
4. Number of TC's failed: 0

Name of Experiment: Commission problem Date: 01-05-2021 Page No. 04

Experiment No: 02 Experiment Result: Verified

PROGRAM - 02

Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the prpective of boundry value testing, devise different test cases, execute these test cases and discuss the test results.

Requirements Specification

Problem defination: The commission problem includes a sales person in the former Arizona territory sold the rifle locks, stocks and barrels made by gunsmith in Missouri. Cost includes.

Locks = 45 $
stocks = 30 $
Barrels = 25 $

The sales person had to sell at least one complete rifle per month and production limits were such that the most the sales person 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 numbers of locks, stocks and barrels sold in the town. At the end of the month, the sales person sent a very short telegram showing -1

chandra's

lock sold. The gunsmith then knew the sales for the month were complete and computed the sales person's commission as follows.

On sales up to (and including) $1000 = 10%
On 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 that gave the total number of locks, stocks and barrels sold, the salesperson's total dollar sales and the finally the commission.

Program code

```
flag = 0
locks = int (input ("Enter the total number of locks"))
stocks = int (input ("Enter the total number of stocks"))
barrels = int (input ("Enter the total number of barrels"))
if locks<0 or locks>70 or stocks<0 or stocks>80 or
    barrels<0 or barrels>90:
        flag=1
if flag==1:
    print ("invalid input")
    exit()
totalsales = (locks*45.0)+(stocks*30.0)+(barrels*25.0)
if totalsales<1000:
    commission = 0.10*totalsales
```

```
elif totalsales < 1800:
    commission = 0.10 * 1000
    commission = commission + (0.15 * (totalsales - 1000))
else:
    commission = 0.10 * 1000
    commission = commission + (0.15 * 800)
    commission = commission + (0.20 * (totalsales - 1800))
print("The totalsale is %d", totalsales)
print("The commission is %f", commission)
```

Algorithm

Step 1: Define lockprice = 45.0, stockprice = 30.0, barrelprice = 25.0

Step 2: Input locks

Step 3: while (lock != -1) ' input device 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) go to step 8 else go to step 9

Step 8: commission = 0.10 * 1000.0; commission = commission + 0.15 * 800.0; commission = commission + 0.20 * (sales - 1800.0);

Step 9: if (sales > 1000.0) goto step 10 else goto step 11

Step 10: commission = 0.10 * 1000.0; commission + 0.15 * (sales - 1000.0);

Step 11: output ("commission is $", commission);

Step 12: exit

Test Cases :-

TC ID	Test cases description	Input data			Sales	Expected output (Commission)	Actual output	Status
		locks	stocks	barrels				
1	Input test cases for locks = 1, stocks = 1, barrels = 1	1	1	1	100	10	10	P
2	Input test cases for locks = 1, stocks = 1, barrels = 2	1	1	2	125	12.5	12.5	P
3	Input test cases for locks = 1, stocks = 2, barrels = 1	1	2	1	130	13	13.0	P
4	Input test cases for locks = 2, stocks = 1, barrels = 1	2	1	1	145	14.5	14.5	P
5	Input test cases for locks = 5, stocks = 5, barrels = 5	5	5	5	500	50	50	P
6	Input test cases for locks = 9, stocks = 10, barrels = 10	9	10	10	955	95.5	95.5	P

7	Input test cases for locks = 10, stocks = 9, barrels = 10	10	9	10	970	97	97	P
8	Input test cases for locks = 10, stocks = 10, barrels = 9	10	10	9	975	97.5	97.5	P
9	Input test cases for locks = 10, stocks = 10, barrels = 10	10	10	10	1000	100	100	P
10	Input test cases for locks = 10, stocks = 10, barrels = 11	10	10	11	1025	103.75	103.75	P
11	Input test cases for locks = 10 stocks = 11 barrels = 10	10	11	10	1030	104.5	104.5	P
12	Input test cases for locks = 11 stocks = 10 barrels = 10	11	10	10	1045	106.75	106.75	P
13	Input test cases for locks = 14 stocks = 14 barrels = 14	14	14	14	1400	160	160	P
14	Input test cases for locks = 17 stocks = 18 barrels = 18	17	18	18	1755	213.25	213.25	P

15	Input test cases for locks = 18 stocks = 17 barrels = 18	18	17	18	1770	215.5	215.5	P
16	Input test cases for locks = 18 stocks = 18 barrels = 17	18	18	17	1775	216.25	216.25	P
17	Input test cases for locks = 18 stocks = 18 barrels = 18	18	18	18	1800	220	220	P
18	Input test cases for locks = 18 stocks = 18 barrels = 19	18	18	19	1825	225	225	P
19	Input test cases for locks = 18 stocks = 19 barrels = 18	18	19	18	1830	226	226	P
20	Input test cases for locks = 19 stocks = 18 barrels = 18	19	18	18	1845	229	229	P
21	Input test cases for locks = 48 stocks = 48 barrels = 48	48	48	48	4800	820	820	P
22	Input test cases for locks = 69 stocks = 80 barrels = 90	69	80	90	7755	1411	1411	P
23	Input test cases for locks = stocks = barrels =	70	79	90	7770	1414	1414	P

24	Input test cases for locks = 70 Stocks = 80 Barrels = 89	70	80	89	7775	1415	1415	P
25	Input test cases for locks = 70 Stocks = 80 barrels = 90	70	80	90	7800	1420	1420	P

Test Report:

1. Number of Tc Executed : 25
2. Number of Defects raised: 0
3. Number of TC's passed: 25
4. Number of TC's failed: 0

Name of Experiment... Next date problem Date... 08/05/2021

Experiment No... 03 Experiment Result... Verified Page No. 09

PROGRAM - 03

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.

Requirement

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

The constraints are:

C1: $1 \leq$ month ≤ 12

C2: $1 \leq$ day ≤ 31

C3: $1812 \leq$ year ≤ 2012

If any one condition out of C1, C2 and C3 fails, then this function produces an output "Value of month not in range".

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

chandra's

Name of Experiment: Next date problem Date: 08/05/2021 Page No. 10

Experiment No: 03 Experiment Result: Verified

Program Code:

```
months = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
day = int (input ("enter the day"))
month = int (input ("enter the month"))
year = int (input ("enter the year"))
num_days = months[month - 1]
if day<1 or day>num_days:
    print ("invalid day")
    exit()
if month<1 or month>12:
    print ("invalid month")
    exit()
if year<1812 or year>2012:
    print ("invalid year")
    exit()
if month == 2:
    if year%4 == 0:
        if year%100 == 0:
            if year%400 != 0:
                num_days = 28
        else:
            num_days = 29
numday = day+1
nummonth = month
numyear = year
if numday > num_days:
    numday = 1
```

chandra's

Name of Experiment: Next date problem Date: 03/05/2021

Experiment No: 03 Experiment Result: Verified Page No. 11

```
nunmonth += 1
if nunmonth > 12:
    nunmonth = 1
    nunyear += 1
print("Given date is %d:%d:%d", day, month, year)
print("Next date is %d:%d:%d", nunday, nunmonth,
                                nunyear)
```

chandra's

Algorithm:

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

Step 2: If MM is 01, 03, 05, 07, 08, 10 do Step 3 else Step 6

Step 3: If DD < 31 then do Step 4 else if DD = 31 do Step 5
else output (invalid date).

Step 4: tomorrow day = DD + 1 goto Step 18

Step 5: tomorrowday = 1; tomorrowmonth = month + 1 goto Step 18

Step 6: if MM is 04, 06, 09, 11 do Step 7

Step 7: if DD < 30 then do to 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 Step 4 else Step 10.

Step 10: tomorrowday = 1, tomorrowmonth = 1, tomorrowyear = year + 1; goto Step 18.

Step 11: if MM is 2

Step 12: if DD < 28 do Step 4 else do Step 13

Step 13: if DD = 28 and YYYY is a leap year do Step 14 else Step 15.

Step 14: tomorrowday = 29 goto Step 18.

Step 15: tomorrow day = 1; tomorrow = 3; goto Step 18.

Step 16: if DD = 29 then do Step 15 else Step 17.

Step 17: output ("Cannot have feb", DD.) Step 19

Step 18: output (tomorrow day, tomorrow month, tomorrow year);

Step 19: exit

Test cases:

TC ID	Description	Input			Expected output	Actual output	Status
		MM	DD	YYYY			
1.	Input of month is min	01	15	1912	01/16/1912	01/16/1912	P
2.	Input of month is min+	02	15	1912	02/16/1912	02/16/1912	P
3.	Input of month is nom	06	15	1912	06/16/1912	06/16/1912	P
4.	Input of month is max-	11	15	1912	11/16/1912	11/16/1912	P
5.	I/p of month is max	12	15	1912	12/16/1912	12/16/1912	P
6.	I/p of day is min	06	01	1912	06/02/1912	06/02/1912	P
7.	I/p of day is min+	06	02	1912	06/03/1912	06/03/1912	P
8.	I/p of day is max-	06	30	1912	07/01/1912	07/01/1912	P
9.	I/p of day is max	06	31	1912	invalid day	invalid day	P
10.	I/p of year is min	06	15	1812	06/16/1812	06/16/1812	P

11.	I/p of year is min+	06	15	1813	06/16/1813	06/16/1813	P
12.	I/p of year is max-	06	15	2011	06/16/2011	06/16/2011	P
13.	I/p of year is max-	06	15	2012	06/16/2012	06/16/2012	P

Test Report:

1. Number of TC executed : 13
2. Number of Defects raised : 0
3. Number of TC's passed : 13
4. Number of TC's failed : 0

Name of Experiment: triangle problem Date: 22/05/2021 Page No. 12

Experiment No.: 04 Experiment Result: Verified

PROGRAM - 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 positive integer numbers (a, b, c) which represents 3 sides of the triangle.

R2. Based on the input it should determine if a triangle can be formed or not.

R3. If the requirement R1 satisfied then the system should determine the type of triangle, can be
- Equilateral (i.e all the three sides are equal)
- Isosceles (i.e two sides are equal)
- scalene (i.e all the three sides are unequal)

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

chandra's

Name of Experiment: Triangle problem Date: 22/05/2021 Page No. 13

Experiment No.: 04 Experiment Result: Verified

Program code:

```
print ("Enter the three sides of the triangle")
a = int (input ("Enter the value of a"))
b = int (input ("Enter the value of b"))
c = int (input ("Enter the value of c"))
if (a>=10) or (b>=10) or (c>=10) or (a<1) or (b<1) or
    (c<1):
    print ("Out of range")
    exit()
if (a<b+c) and (b<a+c) and (c<a+b):
  if (a==b) and (b==c):
      print ("Equilateral triangle")
  elif (a!=b) and (a!=c) and (b!=c):
      print ("Scalene triangle").
  else:
      print ("Isosceles triangle")
else:
    print ("Triangle cannot be formed!")
```

chandra's

Algorithm:

Step 1 : Input a, b and c .i.e. three integer values which represents three side of the triangle.

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

Step 3: if .(a=b) and .(b=c.) then
Print triangle formed is equilateral. Do step 6.

Step 4: if .(a!=b) and .(a!=c) and .(b!=c) then
Print triangle formed is scalene. Do step 6.

Step 5 : print triangle formed is Isosceles.

Step 6 : Stop.

Test cases :-

First attempt : Weak Normal equivalence class (WN1)

TC ID	Test case description	Input data			Expected output	Actual output	Status
		a	b	c			
1	WN1	5	5	5	Equilateral triangle	Equilateral triangle	P
2	WN2	2	2	5	Isosceles triangle	Isosceles triangle	P
3	WN3	3	4	5	Scalene triangle	Scalene triangle	P

4	WN4	4	1	2	Not a triangle	Not a triangle	P
5	WR1	-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	P
6	WR2	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	P
7	WR3	5	5	-1	Value of c is not in the range of permitted value	Value of c is not in the range of permitted values	P
8	WR4	11	5	5	Value of a is not in the range of permitted value	Value of a is not in the range of permitted values	P
9	WR5	5	11	5	Value of b is not in the range of permitted value	Value of b is not in the range of permitted value	P
10	WR6	5	5	11	Value of c is not in the range of permitted value	Value of c is not in the range of permitted value	P

Second attempt

TC ID	Test case description	Input data			Expected output	Actual output	Status
		a	b	c			
1	SR1	-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	P
2	SR2	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	P
3	SR3	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	P

4	SR4	-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	P
5	SR5	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	P
6	SR6	-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	P
7	SR7	-1	-1	-1	Value of a, b, c is not in the range of permitted values	Values of a, b, c is not in the range of permitted values	P

Test Report

1. Number of TC executed : 17
2. Number of Defects raised : 0
3. Number of TC's passed : 17
4. Number of TC's failed : 0

Name of Experiment: Commission problem Date: 05/06/2021

Experiment No: 05 Experiment Result: Verified Page No. 15

PROGRAM - 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 Specification.

Problem definition: The commission problem includes a sales person in the former Arizona territory sold the rifle locks, stocks and barrels made by gunsmith in Missouri. Cost includes.

Locks = 45 $
stocks = 30 $
Barrels = 25 $

The sales person had to sell at least one complete rifle per month and production limits were such that the most the sales person 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 numbers of locks, stocks and barrels sold in the town. At the end of the month, the sales person sent a very short telegram showing -1

chandra's

lock sold. The gunsmith then knew the sales for the month were complete and computed the sales person's commission as follows.

On sales up to (and including) $1000 = 10%
On sales up to (and including) $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 the finally the commission.

Program Code

```
flag = 0
locks = int (input ("Enter the total number of locks"))
stocks = int (input ("Enter the total number of stocks"))
barrels = int (input ("Enter the total number of barrels"))
if locks < 0 or locks > 70 or stocks < 0 or stocks > 80 or
   barrels < 0 or barrels > 90:
      flag = 1
if flag == 1:
   print ("invalid input")
   exit()
totalsales = (locks * 45.0) + (stocks * 30.0) + (barrels * 25.0)
if totalsales < 1000:
   commission = 0.10 * totalsales
```

Name of Experiment: Commission problem Date: 05/06/2021

Experiment No: 05 Experiment Result: Verified Page No: 17

```
elif totalsales < 1800:
    commission = 0.10 * 1000
    commission = commission + (0.15 * (totalsales - 1000))
else:
    commission = 0.10 * 1000
    commission = commission + (0.15 * 800)
    commission = commission + (0.20 * (totalsales - 1800))
print("The totalsale is %d", totalsales)
print("The commission is %f", commission)
```

chandra's

Algorithm

Step 1: Define lockprice = 45.0, stockprice = 30.0, barrelprice = 25.0.

Step 2: Input locks

Step 3: While (locks != -1) 'Input device 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 Step 8 else goto Step 9

Step 8: commission = 0.10 * 1000.0; commission = commission + 0.15 * 800.0; commission = commission + 0.20 * (sales - 1800.0);

Step 9: if (sales > 1000.0) goto Step 10 else goto Step 11

Step 10: commission = 0.10 * 1000.0; commission = commission + 0.15 * (sales - 1000.0);

Step 11: output ("commission is $", commission);

Step 12: exit

Test Cases :-

First attempt : Weak Robust test cases

TC ID	Test case description	Input data			Sales	Expected output (Commision)	Actual output	Status
		locks	stocks	barrels				
1	WR1	10	10	10	100	10	10	P
2	WR2	-1	40	45	Program terminates		Program terminates	P
3	WR3	-2	40	45	Value of locks not in the range 1---70		Value of locks not in the range 1--70	P
4	WR4	71	40	45	Value of locks not in the range 1---70		Value of locks not in the range 1--70	P
5	WR5	35	-1	45	Values of stocks not in the range. 1--80		Value of stocks not in the range 1--80	P
6	WR6	35	81	45	Values of stocks not in the range 1---80		Value of stocks not in the range 1--80	P
7	WR7	35	40	-1	Values of barrels not in the range 1---90		Value of barrels not in the range 1---90	P
8	WR8	35	40	90	Values of barrels not in the range 1--90		Values of barrels not in the range 1---90	P

Second Attempt: Strong robust test cases

TC ID	Test case description	Input data			Sales	Expected output	Actual output	Status
		locks	stocks	barrels				
1	SR1	-2	40	45	Value of locks not in the range 1---70		Value of locks not in the range 1--70	P
2	SR2	35	-1	45	Value of stocks not in the range 1--80		Value of stocks not in the range 1---80	P
3	SR3	35	40	-1	Value of barrels not in the range 1--90		Value of barrels not in the range 1---90	P
4	SR4	-2	-1	45	Value of locks not in the range 1---70 Value of stocks not in the rang -e 1--80		Value of lock not in the range 1--70 Value of stocks not in the range 1---80	P
5	SR5	-2	40	-1	Value of locks not in the range 1---70 Value of barrels not in the range 1--90		Value of locks not in the range 1---70 Value of barrels not in the range 1---90	P
6	SR6	35	-1	-1	Values of stocks and barrels not in range 1---80, 1---90 with respectively		Value of stocks and barrels not in range 1--80, 1---90 with resp -ectively	P

7	SRT	-2	-1	-1	Value of locks not in the range 1--70 Value of stocks not in the range 1--80 Value of barrels not in the range 1--90	Value of locks not in the range 1--70 Value of stocks not in the range 1--80 Value of barrels not in the range 1--90	P

Test Report

1. Number of TC executed : 15
2. Number of Defects raised : 0
3. Number of TC's passed : 15
4. Number of TC's failed : 0

Name of Experiment: Next date function Date: 12/06/2021

Experiment No: 06 Experiment Result: Verified Page No: 19

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

Requirement

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

The constraints are:

C1: $1 \leq month \leq 12$

C2: $1 \leq day \leq 31$

C3: $1812 \leq year \leq 2012$

If any one condition out of C1, C2 and C3 fails, then this function produces an output "Value of month not in range."

A year is called as a leap year if it is divisible by 4, unless it is a century year.

Name of Experiment: Nextdate function Date: 12/06/2021 Page No. 20

Experiment No: 06 Experiment Result: Verified

Centuary years are leap years only if they are multiples of 400. So 1992, 1996, and 2000 are leap years while 1900 is not a leap year.

Program Code:

```
months = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
day = int (input ("Enter the day"))
month = int(input ("Enter the month"))
year = int(input ("Enter the year"))
num_days = months[month - 1]
if day<1 or day>num_days:
    print ("invalid day")
    exit()
if month<1 and month>12:
    print ("invalid month")
    exit()
if year<1812 and year>2012:
    print ("invalid year")
    exit()
if month == 2:
  if year % 4 == 0:
    if year % 100 == 0:
      if year % 400 != 0:
        num_days = 28
    else:
      num_days = 29
```

chandra's

Name of Experiment: Nextdate function Date: 12/06/2021

Experiment No: 06 Experiment Result: Verified


```
numday = day+1
nummonth = month
numyear = year
if numday > num_days:
    numday = 1
    nummonth += 1
if nummonth > 12:
    nummonth = 1
    numyear += 1
print("Given date is %d:%d:%d", day, month, year)
print("Next date is %d:%d:%d", numday, nummonth,
                                numyear)
```

Algorithm

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

Step 2: If MM is 01, 03, 05, 07, 08, 10 do step3 else step 6

Step 3: If DD < 31 then do step 4 else if DD = 31 do step 5 else output (invalid date). step 4: tomorrowday = DD + 1 goto step 18

Step 5: tomorrowday = 1; tomorrowmonth = month + 1
goto step 18

Step 6: If MM is 04, 06, 09, 11 do step 7

Step 7: if DD < 30 then do to 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 step 4 else step 10.

Step 10: tomorrowday = 1, tomorrowmonth = 1, tomorrowyear = year + 1; goto step 18.

Step 11: if MM is 2

Step 12: if DD < 28 do step 4 else do step 13.

Step 13: if DD = 28 and YYYY is a leap year do step 14
else step 15.

Step 14: tomorrowday = 29 goto step 18.

Step 15: tomorrowday = 1; tomorrowmonth = 3, goto step 18.

Step 16 : if DD = 29 then do step 15 else step 17.

Step 17 : output ("cannot have feb", DD) step 19

Step 18 : output (tomorrowday, tomorrowmonth, tomorrowyear);

Step 19 : exit.

Test cases

First attempt : Weak and Strong normal

TC ID	Description	Input			Expected output	Actual output	Status
		MM	DD	YYYY			
1	WN1, SN1	06	15	1990	06|16|1990	06|16|1990	P

Weak robust

TC ID	Description	Inputs			Expected output	Actual output	Status
		MM	DD	YYYY			
1	WR1	6	15	1912	6|16|1912	6|16|1912	P
2	WR2	-1	15	1912	Value of month not in the range 1---12	Value of month not in the range 1--12	P
3	WR3	13	15	1912	Value of month not in the range 1---12	Value of month not in the range 1---12	P
4	WR4	6	-1	1912	Value of ~~month~~ day not in the range 1--31	Value of day not in the range 1---31	P

5	WR5	6	32	1912	Value of day not in the range 1---31	Value of day not in the range 1---31	P
6	WR6	6	15	1811	Value of year not in the range 1812---2012	Value of year not in the range 1812---2012	P
7	WR7	6	15	2013	Value of year not in the range 1812---2012	Value of year not in the range 1812---2012	P

Strong Robust

TC ID	Discription	Inputs			Expected output	Actual output	status
		MM	DD	YYYY			
1	SR1	-1	15	1912	Value of month not in the range 1---12	Value of month not in the range 1---12	P
2	SR2	6	-1	1912	Value of day not in the range 1---31	Value of day not in the range 1---31	P
3	SR3	6	15	1811	Value of year not in the range 1812---2012	Value of year not in the range 1812---2012	P
4	SR4	-1	-1	1912	Value of month not in the range 1---12 value of day not in the range 1---31	Value of month not in the range 1---12 value of day not in the range 1---31	P
5	SR5	6	-1	1811	Value of day not in the range 1---31 Value of year not in the range 1812---2012	Value of day not in the range 1---31 Value of year not in the range 1812---2012	P

6	SR6	-1	15	1811	Value of month not in the range 1--12 Value of year not in the range. 1812 — 2012	Value of month not in the range 1---12 Value of year not in the range 1812--- 2012	P
7	SR7	-1	-1	1811	Value of month, day and year not in the range. 1---12, 1---31 and 1812--- 2012 with respectively	Value of month, day and year not in the range 1---12, 1---31 and 1812----2012 with respectively	P

Second Attempt

Weak normal.

TC ID	Description	Input			Expected output	Actual output	Status
		MM	DD	YYYY			
1	WN1	6	14	2000	6|15|2000	6|15|2000	P
2	WN2	7	29	1996	7|30|1996	7|30|1996	P
3	WN3	2	30	2002	Invalid input date	Invalid input date	P
4	WN4	6	31	2000	Invalid input date	Invalid input date	P

Strong Normal

TC ID	Description	Inputs			Expected output	Actual output	Status
		MM	DD	YYYY			
1	SN1	6	14	2000	6|15|2000	6|15|2000	P
2	SN2	6	14	1996	6|15|1996	6|15|1996	P
3	SN3	6	14	2002	6|15|2002	6|15|2002	P
4	SN4	6	29	2000	6|30|2000	6|30|2000	P
5	SN5	6	29	1996	6|30|1996	6|30|1996	P
6	SN6	6	29	2002	6|30|2002	6|30|2002	P
7	SN7	6	30	2000	Invalid i/p date	Invalid i/p date	P
8	SN8	6	30	1996	Invalid i/p date	Invalid i/p date	P
9	SN9	6	30	2002	Invalid i/p date	Invalid i/p date	P
10	SN10	6	31	2000	Invalid i/p date	Invalid i/p date	P
11	SN11	6	31	1996	Invalid i/p date	Invalid i/p date	P
12	SN12	6	31	2002	Invali- i/p date	Invalid i/p date	P
13	SN13	7	14	2000	7|15|2000	7|15|2000	P
14	SN14	7	14	1996	7|15|1996	7|15|1996	P
15	SN15	7	14	2002	7|15|2002	7|15|2002	P
16	SN16	7	29	2000	7|30|2000	7|30|2000	P
17	SN17	7	29	1996	7|30|1996	7|30|1996	P
18	SN18	7	29	2002	7|30|2002	7|30|2002	P
19	SN19	7	30	2000	7|31|2000	7|31|2000	P
20	SN20	7	30	1996	7|31|1996	7|31|1996	P
21	SN21	7	30	2002	7|31|2002	7|31|2002	P
22	SN22	7	31	2000	8|1|2000	8|1|2000	P
23	SN23	7	31	1996	8|1|1996	8|1|1996	P
24	SN24	7	31	2002	8|1|2002	8|1|2002	P
25	SN25	2	14	2000	2|15|2000	2|15|2000	P
26	SN26	2	14	1996	2|15|1996	2|15|1996	P
27	SN27	2	14	2002	2|15|2002	2|15|2002	P

28	SN28	2	29	2000	3/1/2000	3/1/2000	P
29	SN29	2	29	1996	3/1/1996	3/1/1996	P
30	SN30	2	29	2002	Invalid i/p date	Invalid i/p date	P
31	SN31	2	30	2000	Invalid i/p date	Invalid i/p date	P
32	SN32	2	30	1996	Invalid i/p date	Invalia i/p date	P
33	SN33	2	30	2002	Invalid i/p date	Invalid input date	P
34	SN34	2	31	2000	Invalid i/p date	Invalid i/p date	P
35	SN35	2	31	1996	Invalid i/p date	Invalid i/p date	P
36	SN36	2	31	2002	Invalid i/p date	Invalid i/p date	P

Test Report

1: Number of test cases Executed : 55

2: Number of Defects Raised: 0

2: Number of Test cases passed : 55

4: Number of Test cases failed : 0

Name of Experiment: Triangle problem Date: 19/06/2021

Experiment No: 07 Experiment Result: Verified Page No. 24

PROGRAM-07

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

Requirements :

R1: The system should accept 3 positive integer numbers (a, b, c) which represent 3 sides of the triangle. Based on the input it should determine if a triangle can be formed or not.

R2: If the requirement R1 satisfied then the system should determine the type of triangle, can be.

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

else suitable error message should be displayed. Here are

chandra's

Name of Experiment: Triangle problem Date: 19/05/2021

Experiment No: 07 Experiment Result: verified Page No. 25

assume that user gives three positive integer numbers as a input.

Program code:

```
print ("Enter the three sides of the triangle")
a=int (input ("Enter the value of a"))
b = int (input ("Enter the value of b"))
c = int (input ("Enter the value of c"))
if (a<b+c) and (b<a+c) and (c<a+b):
    if (a==) and (b==c):
        print ("Equilateral triangle")
    elif (a!=b) and (a!=c) and (b!=c):
        print ("scalene triangle")
    else:
        print ("Isoscales triangle")
else:
    print ("Triangle cannot be formed")
```

chandra's

Algorithm

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

Step 2 : if (a < (b+c)) and (b < (a+c)) and (c < (a+b)) then do step 3 else.
Print not a triangle.. Do step 6.

Step 3 : if (a = b) and (b = c) then
Print triangle formed is equilateral . Do step 6.

Step 4 : if (a != b) and (a != c) and (b != c) then
Print triangle formed is scalene. Do step 6.

Step 5 : print triangle formed is Isosceles.

Step 6 : stop.

Testing

Decision Table Approach

	Conditions	Condition Entry (Rules)										
		R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_{09}	R_{10}	R_{11}
C_1	: a<b+c?	F	T	T	T	T	T	T	T	T	T	T
C_2	: b<a+c?	-	F	T	T	T	T	T	T	T	T	T
C_3	: c<a+b?	-	-	F	T	T	T	T	T	T	T	T
C_4	: a==b?	-	-	-	F	T	T	T	F	F	T	F
C_5	: a==c?	-	-	-	T	F	T	F	T	F	T	F
C_6	: b==c?	-	-	-	T	T	F	F	F	T	T	F
Actions		Action Entries										
A_1	: not a triangle	X	X	X								
A_2	: Scalene											X
A_3	: Isosceles							X	X	X		
A_4	: equilateral										X	
A_5	: impossible				X	X	X					

Deriving test cases using Decision table approach

Test cases:

TC ID	Test case Description	a	b	c	Expected output	Actual output	Status
1	Testing for Req 1	H	1	2	Not a triangle	Not a triangle	P
2	Testing for Req 1	1	H	2	Not a triangle	Not a triangle	P
3	Testing for Req 1	1	2	H	Not a triangle	Not a triangle	P
4	Testing for Req 2	5	5	5	Equilateral triangle	Equilateral triangle	P
5	Testing for Req 2	2	2	3	Isosceles	Isosceles	P
6	Testing for Req 2	2	3	2	Isosceles	Isosceles	P
7	Testing for Req 2	3	2	2	Isosceles	Isosceles	P
8	Testing for Req 2	3	4	5	Scalene triangle	Scalene triangle	P

Test Report

1. Number of TC Executed : 08
2. Number of Defects raised : 0
3. Number of TC's passed : 08
4. Number of TC's failed : 0

Name of Experiment... Commission problem Date... 19/06/2021 Page No. 27

Experiment No... 08 Experiment Result... Verified

PROGRAM - 08

Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perpective 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. $1 \leq$ locks ≤ 70)
(i.e $1 \leq$ stocks ≤ 80)
(i.e $1 \leq$ barrels ≤ 90)

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

10% on sales up to (and including) $1000
15% on next $800
20% on any sales in excess of $1800

Also the System should compute the total dollar sales.

chandra's

Name of Experiment: Commission problem Date: 19/06/2021 Page No. 28

Experiment No: 08 Experiment Result: Verified

The system should output sales persons total dollar sale and his commission.

Program Code

```
flag = 0
locks = int(input("Enter the total number of locks"))
stocks = int(input("Enter the total number of stocks"))
barrels = int(input("Enter the total number of barrels"))
if locks < 0 or locks > 70 or stocks < 0 or stocks > 80 or
barrels < 0 or barrels > 90:
    flag = 1
if flag == 1:
    print("invalid input")
    exit()
totalsales = (locks * 45.0) + (stocks * 30.0) + (barrels * 25.0)
if totalsales < 1000:
    commission = 0.10 * totalsales
elif totalsales < 1800:
    commission = 0.10 * 1000
    commission = commission + (0.15 * (totalsales - 1000))
else:
    commission = 0.10 * 1000
    commission = commission + (0.15 * 800)
    commission = commission + (0.20 * (totalsales - 1800))
print("The totalsales is %d", totalsales)
print("The commission is %f", commission)
```

chandra's

Algorithm

Step 1: Define lockprice = 45.0, stockprice = 30.0, barrelprice = 25.0.

Step 2: Input locks.

Step 3: While (locks != -1) 'input device 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 go to Step 9

Step 8: commission = 0.10 * 1000.0; commission = commission + 0.15 * 800.0; commission = commission + 0.20 * (sales - 1800.0);

Step 9: if (sales > 1000.0) goto Step 10 else goto Step 11

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 : Decision table approach

Conditions	Condition Entries (Rule)					
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 > 1000 ?	–	–	–	–	T	F
C6 : Sales ≥ 1000 ?	–	–	–	–	–	T
Action	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			

Deriving test cases using decision table approach:

Test cases

TC ID	Test cases description	Input data			sales	Expected output	Actual output	status
		locks	stocks	barrels				
1	Testing for Req 1 condition C_1	-2	40	45	out of range		Out of range	P
2	Testing for Req 1 Condition C_1	90	40	45	out of range		out of range	P
3	Testing for Req 1 Condition C_2	35	-2	45	Out of range		Out of range	P
4	Testing for Req 1 condition C_2	35	90	45	Out of range		Out of range	P
5	Testing for Req 1 Condition C_3	35	40	-1	Out of range		Out of range	P
6	Testing for Req 1 condition C_3	35	45	100	Out of range		Out of range	P
7	Testing for Req 2	5	5	5	500	50	50	P
8	Testing for Req 2	15	15	15	1500	1.75	175	P
9	Testing for Req 2	25	25	25	2500	360	360	P

Test Report

1. Number of TC Executed : 09
2. Number of Defects raised : 0
3. Number of TC's passed : 09
4. Number of TC's failed : 0

Name of Experiment: Commission problem Date: 24/07/2021

Experiment No: 09 Experiment Result: Verified Page No: 31

PROGRAM - 09

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

Requirement Specification

Problem definition: The commission problem includes a sales person in the former Arizona territory sold the rifle locks, stocks and barrels made by gunsmith in Missouri. Cost includes.

Locks = 45 $
Stocks = 30 $
Barrels = 25 $

The sales person had to sell atleast one complete rifle per month and production limits were such that the most the sales person 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.

chandra's

At the end of the month, the sales person sent a very short telegram showing -1 locks sold. The gunsmith then knew the sales for the month were complete and computed the sales person's commission as follows

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

The commission program produces a monthly sales report that gives the total number of locks, stocks, and barrels sold, the salesperson's total dollar sales, and the family the commission

Program code

```
flag = 0
locks = int(input("Enter the total number of locks"))
stocks = int(input("Enter the total number of stocks"))
barrels = int(input("Enter the total number of barrels"))
if lock < 0 or locks > 70 or stocks < 0 or stocks > 80 or
   barrels < 0 or barrels > 90:
flag = 1
if flag == -1:
   print("Invalid input")
   exit()
totalsales = (locks * 45.0) + (stocks * 30.0) + (barrels * 25.0)
if totalsales < 1000:
```

chandra's

```
        commission = 0.10 * totalsales.
elif totalsales < 1800:
        commission = 0.10 * 1000
        commission = commission + (0.15 * (totalsales - 1000))
else:
        commission = 0.10 * 1000
        commission = commission + (0.15 * 800)
        commission = commission + (0.20 * (totalsales - 1800))
print("The totalsale is %d", totalsales)
print("The commission is %f", commission)
```

Algorithm

Step 1: Define lockprice = 45.00, stockprice = 30.00, barrelprice = 25.00

Step 2: Input locks

Step 3: while (locks != -1) ! input device uses -1 to indicate end of data' goto step 12.

Step 4: input (stocks, barrels)

Step 5: Compute locksales, stocksales, barrelsales and ~~also~~ sales

Step 6: output ("Total sales", sales)

Step 7: if (sales > 1800.0) goto step 8 else goto step 9

Step 8: commission = 0.10 * 1000.0; commission = commission + 0.15 * 800.0; commission = commission + 0.20 * (sales - 1800.0);

Step 9: if (sales > 1000.0) goto step 10 else step 11.

Step 10: commission = 0.10 * 1000.0; commission = commission + 0.15 * (sales - 1000.0);

Step 11: output ("commission in $", commission);

Step 12: exit

Testing technique : Dataflow Testing.

Data-flow testing : Key steps

Given a code.

1. Number the lines
2. List the variables
3. List occurences and assign a category to each variable
4. Identify dev-pairs and their use (p-uses and c-uses)
5. Define test cases, depending on the required coverage.

Step 3

Table 1: List occurences and assign a category to each var

Line No.	Category		
	Definition	c-use	p-use
2, 3, 4	Locks, stocks, barrels		
5			Locks, stocks, barrels
6	Flag		
7			flag
10	totalsales	Locks, stocks, barrels	
11			totalsales
12	Commission	totalsales	
13			totalsales
14	Commission		
15	Commission	Commission, totalsales	
17	Commission		
18	Commission	Commission	
19	Commission	Commission, totalsales	
20, 21		Commission, totalsales	

Step 4

Table 2: Indentify def-pairs and their use (p-uses & c-uses)

Definition-use pair	Variables	
start-line → end-line	c-use	p-use
2, 3, 4 → 5		Locks, stocks, barrels
6 → 7		Flag
2, 3, 4, → 10	Locks, stocks, barrels	
10 → 11		totalsale
10 → 13		totalsales
12 → 21	Commission	
14 → 15	Commission	
15 → 21	Commission	
17 → 18	Commission	
18 → 19	Commission	
19 - 21	Commission	

Step 5

Variables	Du-pair	Sub-path	Inputs			Expected output	
			locks	stocks	barrels	t_sales	commiss-ion
locks, stocks, barrels	2,3,4→10	8, 9, 18	10	10	10	1000	100
locks, stocks, barrels	2,3,4→5	8,9	5	-1	22	Invalid input	Invalid input
Flag	6→7	11, 12, 13	-1	5	99	Invalid	input
totalsales	10→11	18, 19	5	5	5	500	50
totalsales	10→13	18, 19, 23	15	15	15	1500	175
commission	12→21	21, 34	5	5	5	500	50
commission	15→21	06, 34	15	15	15	1500	175
commission	19→21	32, 34	25	25	25	2500	260

Test Report

1. Number of TC Executed: 8
2. Number of Defects raised: 0
3. Number of TC's passed: 8
4. Number of TC's failed: 0

Name of Experiment: binary search Date: 24/07/2021 Page No. 35

Experiment No: 10 Experiment Result: Verified

PROGRAM - 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.

Requirements:

R1: The system should accept ~~and~~ n numbers 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

Program Code:

```
n = int(input("Enter the size of the list:"))
flag = 0
sortedlist = []
for i in range(n):
    sortedlist.append(input("Enter %d th element:"%i))
x = input("Enter the number to search:")
low = 0
high = n-1
```

chandra's

```
while (low <= high):
    mid = int ((low + high) / 2)
    if (x == sortedlist[mid]):
        flag = 1
        break
    elif (x < sortedlist[mid]):
        high = mid - 1
    else:
        low = mid + 1
if (flag == 1):
    print ("Successful search entered number %s is
        present at position : %d" % (x, mid))
else:
    print ("unsuccessful search entered number %s is
        not present at position" %x)
```

Algorithm

Step 1: Input value of n. Enter n integer numbers in array
int mid;

Step 2: Initialize low = 0, high = n-1;

Step 3: until (low <= high) do
mid = (low+high)/2;
if (a[mid] == key) then do step 5
else if (a[mid] > key) then do
high = mid-1
else
low = mid + 1

Step 4: Print unsuccessful search do step 6

Step 5: Print successful search, element found at position
mid+1

Step 6: stop

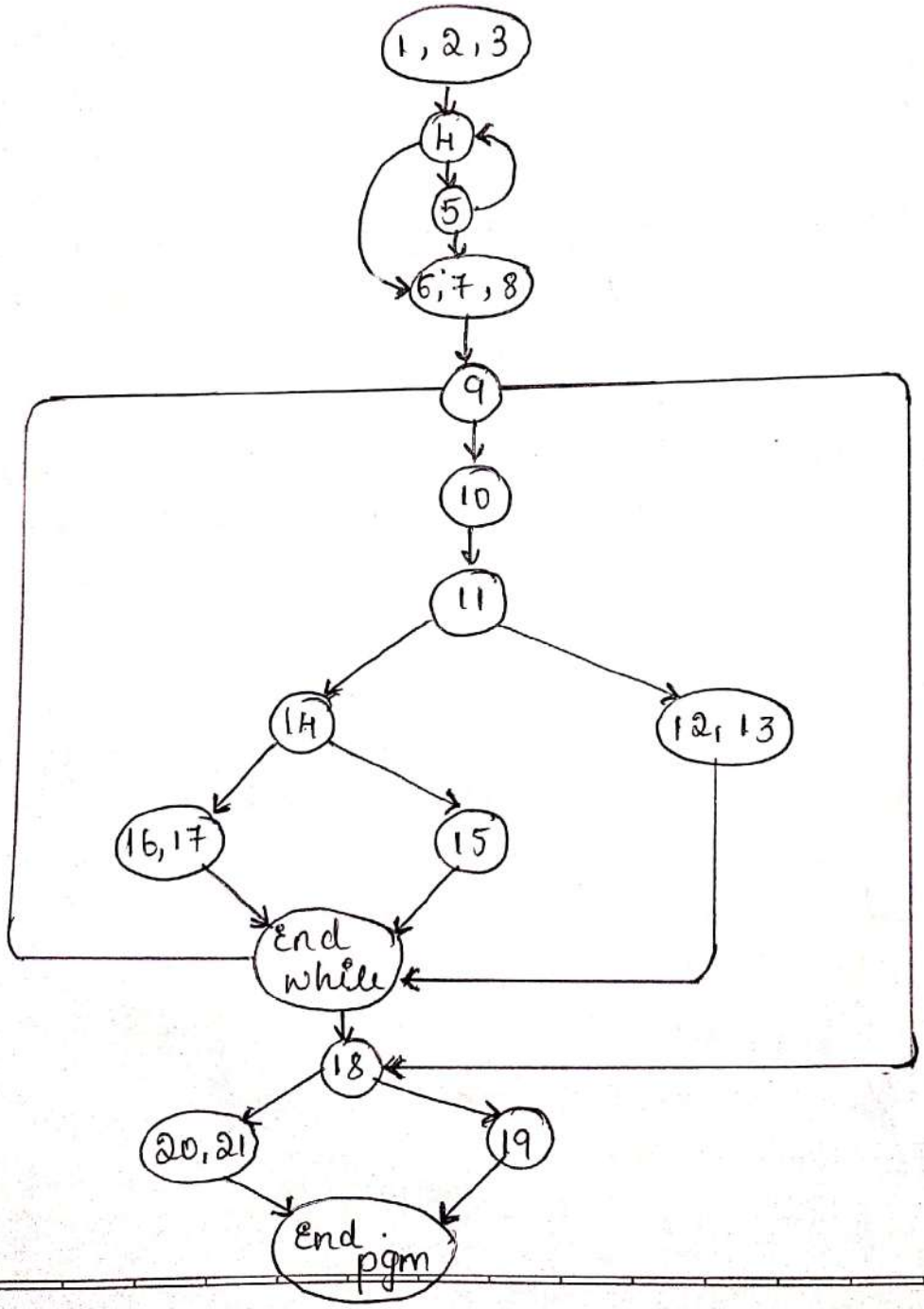
Testing

Technique used: Basis Path Testing

Basis path testing is a form of structural testing (white box testing). The method devised by McCabe to carry out basis path testing has four steps: these are

1. Compute the program graph
2. Calculate Cyclomatic complexity
3. Select a basis set of paths
4. Generate test cases for each of these paths.

Step 1: Program graph for binary search



Using the program graph we derive (Decision to Decision) DD path graph for binary search program.

DD path name	Program graph nodes
A	1, 2, 3
B	4
C	5
D	6, 7, 8
E	9
F	10
G	11
H	12, 13
I	14
J	15
K	16, 17
L	18
M	19
N	20, 21