## Experiment no - 01

**Experiment Name - Decision Table Approach for Solving Triangle Problem Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\* 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 decision-table approach, execute the test cases and discuss the results \*/

## Program:-

a=int(input()) b=int(input()) c=int(input())

def check\_Triangle(side1,side2,side3):

if (side1 != 0 and side2!=0 and side3 !=0) or (side1<side2+side3 and side2 < side1+side3 and side3<side1+side2):

if side1 == side2 and side2 == side3: print("Triangle is Equilateral")

elif side1 == side2 or side1 == side3 or side2 == side3: print("Triangle is Isoceless")

else:

print("Triangle is Scalen")

else:

print("Triangle cannot be formed")

check\_Triangle(a,b,c)

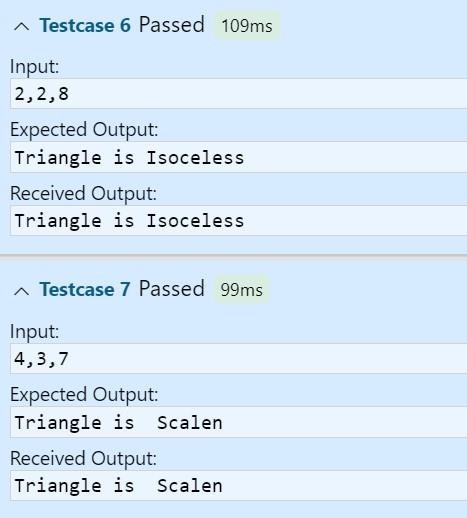
Test Case Name : Decision table for triangle problem Test Data : Enter the 3 Integer Value( a , b And c ) Pre-condition : a < b + c , b < a + c and c < a + b

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

Input data decision Table

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 < a + b | - | - | F | T | T | T | T | T | T | T | T |
| C4 : a = b | - | - | - | T | T | T | T | F | F | F | F |
| C5 : a = c | - | - | - | T | T | F | F | T | T | F | F |
| C6 : b = c | - | - | - | T | F | T | F | T | F | T | F |
| **Actions** | a1 : Not a triangle | X | X | X |  |  |  |  |  |  |  |  |
| a2 : Scalene triangle |  |  |  |  |  |  |  |  |  |  | X |
| a3 : Isosceles triangle |  |  |  |  |  |  | X |  | X | X |  |
| a4 : Equilateral triangle |  |  |  | X |  |  |  |  |  |  |  |
| a5 : Impossible |  |  |  |  | X | X |  | X |  |  |  |

## Test Cases:



Triangle Problem -Decision Table Test cases for input data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Case Id** | **Description** | **Input Data** | | | **Expected Output** | **Actual Output** | **Status** |
| **a** | **b** | **c** |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  |  |  |

## Experiment no - 02

**Experiment Name - Boundary Value Analysis for triangle problem Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\* 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 \*/

## Program:-

a = int(input()) b = int(input()) c = int(input())

c1 = a >= 1 and a <= 10 c2 = b >= 1 and b <= 10 c3 = c >= 1 and c <= 10 res = "n"

while(not (c1 and c2 and c3)): if(not c1):

print("The value of a is not permitted") if(not c2):

print("The value of b is not permitted") if not c3:

print("The value of c is not permitted") res = "y"

break if(res == "n"):

if a < b+c and b < a+c and c < a+b:

istr = 'y' else:

istr = 'n' if(istr == 'y'):

if(a== b) and(b == c):

print("Equilateral triangle") elif(a!=b) and (a!=c) and (b!=c):

print("saclene triangle") else:

print("isosceles triangle") else:

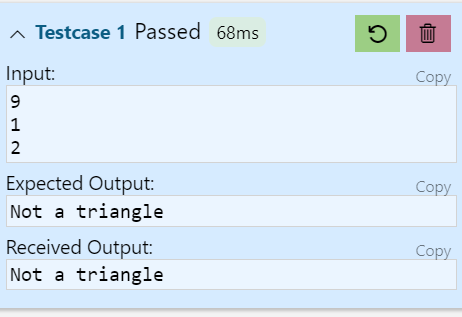
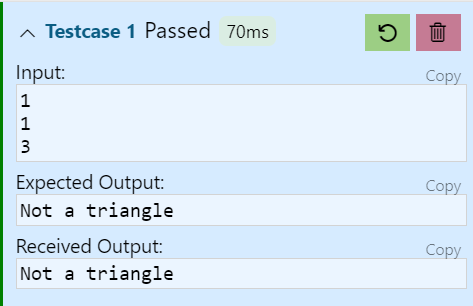
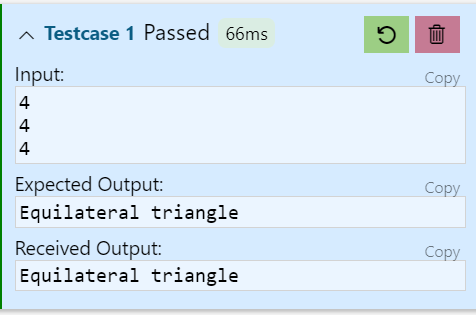
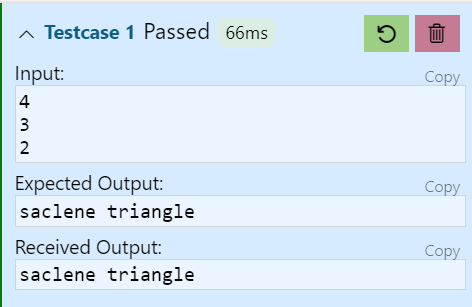
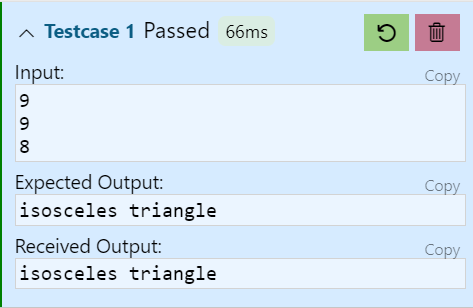
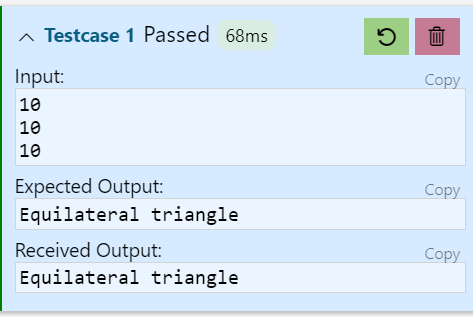
print("Not a triangle")

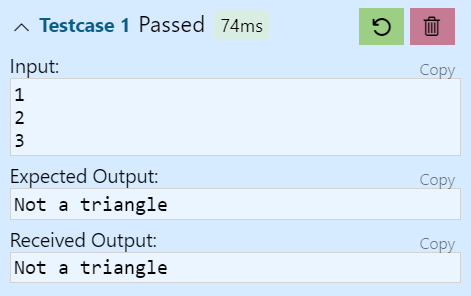
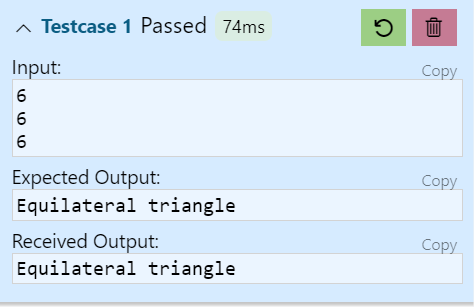
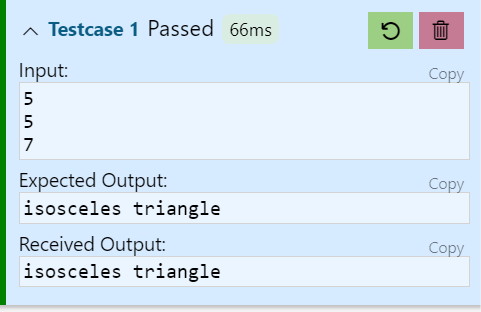
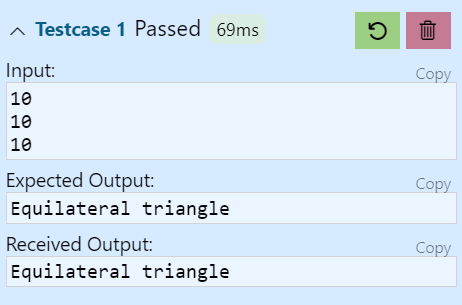
Test Case Name :Boundary Value Analysis for triangle problem Test Data : Enter the 3 Integer Value( a , b And c )

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 form a triangle

Test Cases:-





Triangle Problem -Boundary value Test cases for input data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Case Id** | **Description** | **Input Data** | | | **Expected Output** | **Actual Output** | **Status** |
| **a** | **b** | **c** |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |

Experiment no - 03

Experiment Name - Equivalence class Analysis for triangle problem

## Name – Mayur Santosh Hajare Roll No - 54

**Class - TYIT Batch – T4**

Date of Conduction –

**/\* 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 Equivalence class analysis, execute the test cases and discuss the results \*/**

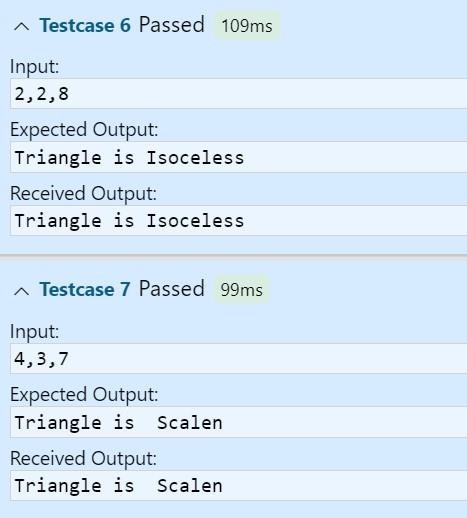
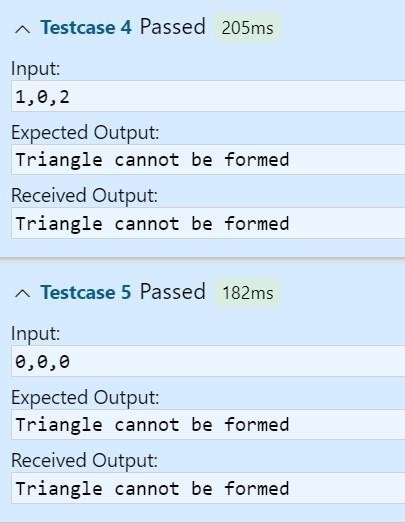
**Test Case Name :Equivalence class Analysis for triangle problem**

**Test Data : Enter the 3 Integer Value( a , b And c ) Classes: { (1,3),(3,5),(5,7)(7,10) }**

**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 form a triangle**

## Test Cases:-



Triangle Problem - Equivalence Class Test cases for input data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Weak Equivalence class Testing** | | | | | | | |
| **Case Id** | **Description** | **Input Data** | | | **Expected Output** | **Actual Output** | **Status** |
| **a** | **b** | **C** |
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |

**Weak Robust Equivalence Class Testing**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |

**Strong Robust Equivalence class Testing**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |
|  |  |  |
| 15 |  |  |  |  |  |  |  |
|  |  |  |
| 16 |  |  |  |  |  |  |  |
|  |  |  |
| 17 |  |  |  |  |  |  |  |
|  |  |  |
|  |  |  |

## Experiment no - 04

**Experiment Name - Boundary Value for Commission Problem Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

## /\* Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of boundary value, derive test cases, execute these test cases and discuss the test results \*/

**/\* Assumption price for lock=45.0, stock=30.0 and barrels=25.0 production limit could sell in a month 70 locks,80 stocks and 90 barrels commission on sales = 10 % <= 1000 and 15 % on 1000 to 1800 and 20 % on above 1800\*/**

Program:-

lock = int(input()) stock = int(input()) barrel = int(input())

lock\_price = 45

stock\_price = 30

barrel\_price = 25

total\_sell = lock\*lock\_price + stock\*stock\_price + barrel\*barrel\_price if total\_sell > 7800:

print("Sells are Exceeding!") elif total\_sell < 100:

print("No Sells") elif total\_sell <= 1000:

commision = (total\_sell\*10)/100 print("Total sell:"+str(total\_sell))

print("The commision of Total sell is Rs."+str(commision)) elif total\_sell >1000 and total\_sell<=1800:

commision = (total\_sell\*18)/100 print("Total sell:"+str(total\_sell))

print("The commision of Total sell is Rs."+str(commision)) elif total\_sell > 1800:

commision = (total\_sell\*20)/100 print("Total sell:"+str(total\_sell))

print("The commision of Total sell is Rs."+str(commision))

**Test Case Name : Boundary Value for Commission Problem Test data :** price Rs for lock - 45.0 , stock - 30.0 and barrel - 25.0

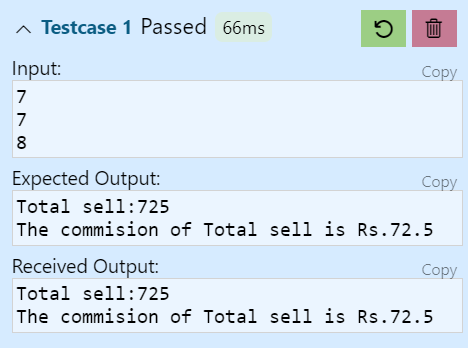
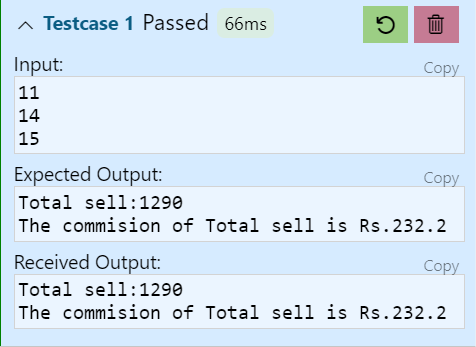
sales = total lock \* lock price + total stock \* stock price + total barrel \* barrel price

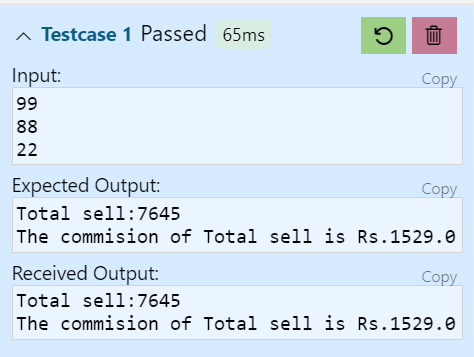
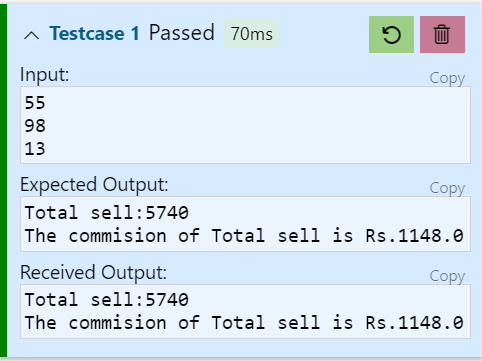
commission : 10% up to sales Rs 1000 , 15 % of the next Rs 800 and 20 % on any sales in excess of 1800

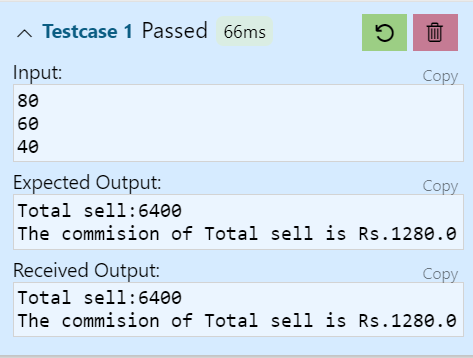
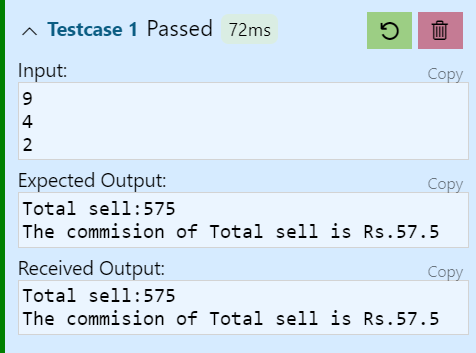
**Pre-condition :** lock = -1 to exit and 1< =lock < = 70 , 1<=stock <=80 and 1<=barrel<=90

**Brief Description :** The salesperson had to sell at least one complete rifle per month.

## Test cases:-







CHECKING BOUNDARY VALUE FOR LOCKS, STOCKS AND BARRELS AND COMMISSION

Commission Problem Output Boundary Value Analysis Cases

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | Actual output | | Status |
| Total Locks | Total Stocks | Total Barrels | Sales | Comm- ission | Sales | Comm  -ission |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |

Output Special Value Test Cases

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseId | Description | Input Data | | | Expected Output | | Actual output | |  |
| Total Locks | Total Stocks | Total Barrels | Sales | Commission | Sales | Commission | Status |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |

## Experiment no - 05

**Experiment Name - Equivalence Class for Commission Problem Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\* Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of Equivalence Class, derive test cases, execute these test cases and discuss the test results \*/

Test Case Name :Equivalence Class for Commission Problem 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

Brief Description : The salesperson had to sell at least one complete rifle per month. Checking boundary value for locks, stocks and barrels and commission

Valid Classes

L1 ={LOCKS :1 <=LOCKS<=70}

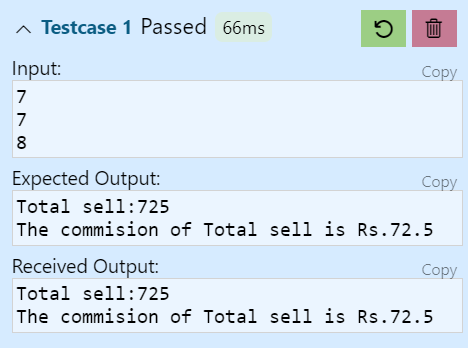
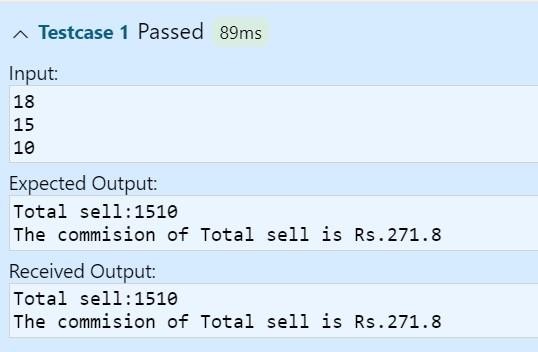
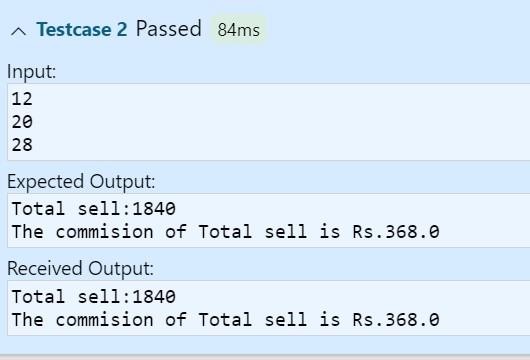
L2 ={Locks=-1}(occurs if locks=-1 is used to control input iteration) L3 ={stocks : 1<=stocks<=80} L4= {barrels :1<=barrels<=90}

## Invalid Classes

L3 ={locks: locks=0 OR locks<-1} L4 ={locks: locks> 70} S2 ={stocks : stocks<1} S3 ={stocks : stocks >80}

B2 ={barrels : barrels <1} B3 =barrels : barrels >90}

## Test Cases:-



Commission Problem Output Equivalence Class Testing ( Weak & Strong Normal Equivalence Class

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | Actual output | | Status |
| Total  Locks | Total  Stocks | Total  Barrels | Sales | Commission | Sales | Commiss  ion |
| 1 |  |  |  |  |  |  |  |  |  |

Weak Robustness Equivalence Class

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CaseId | Description | Input Data | | | Expected Output | Actual output | Status |
| Locks | Stocks | Barrels |
| WR1 |  |  |  |  |  |  |  |
| WR2 |  |  |  |  |  |  |  |
| WR3 |  |  |  |  |  |  |  |
| WR4 |  |  |  |  |  |  |  |
| WR5 |  |  |  |  |  |  |  |
| WR6 |  |  |  |  |  |  |  |
| WR7 |  |  |  |  |  |  |  |

Strong Robustness Equivalence Class

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| CaseId | Description | Input Data | | | Expected Output | Actual output | Status |
| Locks | Stocks | Barrels |
| SR1 |  |  |  |  |  |  |  |
| SR2 |  |  |  |  |  |  |  |
| SR3 |  |  |  |  |  |  |  |
| SR4 |  |  |  |  |  |  |  |
|  |  |  |
| SR5 |  |  |  |  |  |  |  |
|  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SR6 |  |  |  |  |  |  |  |
|  |  |  |
| SR7 |  |  |  |  |  |  |  |
|  |  |  |
|  |  |  |

Some addition equivalence Boundary checking

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CaseId | Description | Input Data | | | Expected Output | | Actual output | |  |
| Total Locks | Total Stocks | Total Barrels | Sales | Commission | Sales | Commission | Status |
| OR1 |  |  |  |  |  |  |  |  |  |
| OR2 |  |  |  |  |  |  |  |  |  |
| OR3 |  |  |  |  |  |  |  |  |  |

## Experiment no - 06

**Experiment Name - Decision Table for Commission Problem Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\* Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of Decision Table approach, derive test cases, execute these test cases and discuss the test results \*/

Test Case Name :Decision Table for Commission Problem

**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

**Brief Description :** The salesperson had to sell at least one complete rifle per month.

Input data decision Table

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **RULES** | | **R1** | **R2** | **R**  **3** | **R4** | **R5** | **R**  **6** | **R7** | **R8** | **R10** |
| **Conditions** | C1: Locks = -1 | T | F | F | F | F | F | F | F | F |
| C2 : 1 ≤ Locks ≤ 70 | - | T | T | F | T | F | F | F | T |
| C3 : 1 ≤ Stocks ≤ 80 | - | T | F | T | F | T | F | F | T |
| C4 : 1 ≤ Barrels ≤ 90 | - | F | T | T | F | F | T | F | T |
| **Actions** | a1 : Terminate the input loop | X |  |  |  |  |  |  |  |  |
| a2 : Invalid locks input |  |  |  | X |  | X | X | X |  |
| a3 : Invalid stocks input |  |  | X |  | X |  | X | X |  |
| a4 : Invalid barrels input |  | X |  |  | X | X |  | X |  |
| a5 : Calculate total locks, stocks and barrels |  | X | X | X | X | X | X |  | X |
| a5 : Calculate Sales | X |  |  |  |  |  |  |  |  |
| a6: proceed to commission decision table | X |  |  |  |  |  |  |  |  |

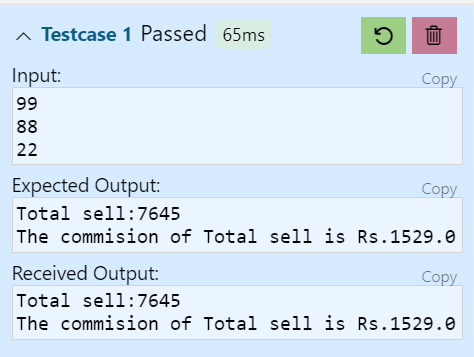
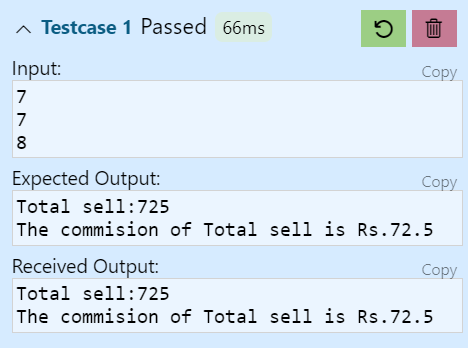
Commission calculation Decision Table (Precondition : lock = -1)

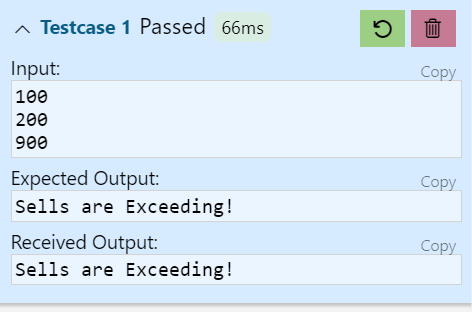
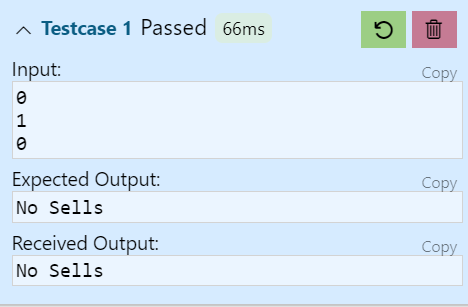
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RULES** | | **R1** | **R2** | **R3** | **R4** |
| **Condition** | C1 : tlocks>0 && tstocks>0 && tbarrels>0 | T | T | T | F |
| C1 : Sales > 0 AND Sales ≤ 1000 | T | F | F |  |
| C2 : Sales > 1001 AND sales ≤ 1800 |  | T | F |  |
| C3 : sales ≥1801 |  |  | T |  |
| **Actions** | A1 : Cannot calculate the commission |  |  |  | X |
| A2 : comm= 10%\*sales | X |  |  |  |
| A3 : comm = 10%\*1000 + (sales-1000)\*15% |  | X |  |  |
| A4 : comm = 10%\*1000 + 15% \* 800 + (sales-1800)\*20% |  |  | X |  |

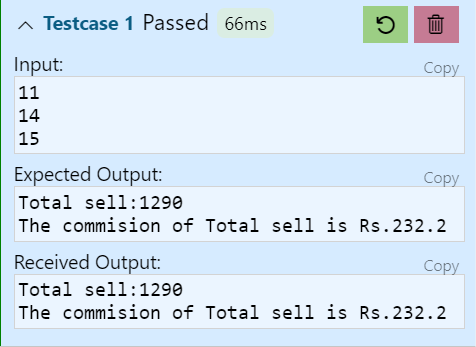
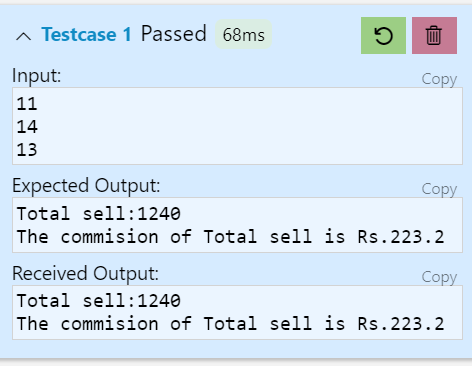
Precondition : Initial Value Total Locks= 0 , Total Stocks=0 and Total Barrels=0

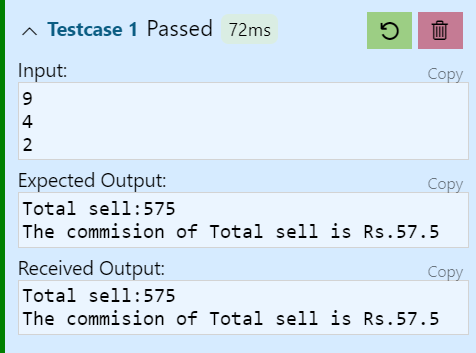
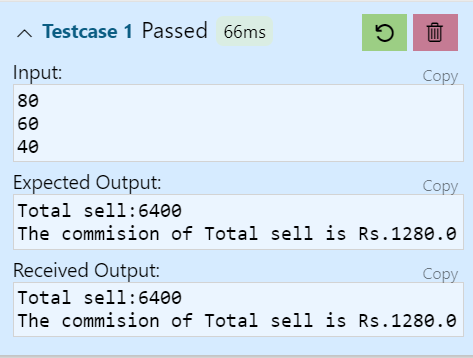
Precondition Limit :Total locks, stocks and barrels should not exceed the limit 70,80 and 90 respectively NIR-Not in Range SAEO-Same as Expected Output

Test Cases:-







Commission Problem Decision Table Test cases for input data

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Output | | | | | Actual Output | Status |
| Locks | Stocks | Barrels | Terminate  i/p loop | Calculate  commission | Total  locks | Total  stocks | Total  Barrels |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |

Commission Problem -Decision Table Test cases for commission calculation L-locks S-Stocks

B-Barrels

Precondition : Locks = -1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | | Actual Output | Status |
| L | S | B | Expected Sales | Actual Sales | Expected Commission | Actual Commission |
| 1 |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |

## Experiment no - 07

**Experiment Name – Binary search – Path Testing Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\* Design, develop a 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

\*/

Program:-

def binarySearch(array, x, low, high): while low <= high:

mid = (low + high) //2 if array[mid] == x:

return mid

elif x > array[mid]:S low = mid + 1

else:

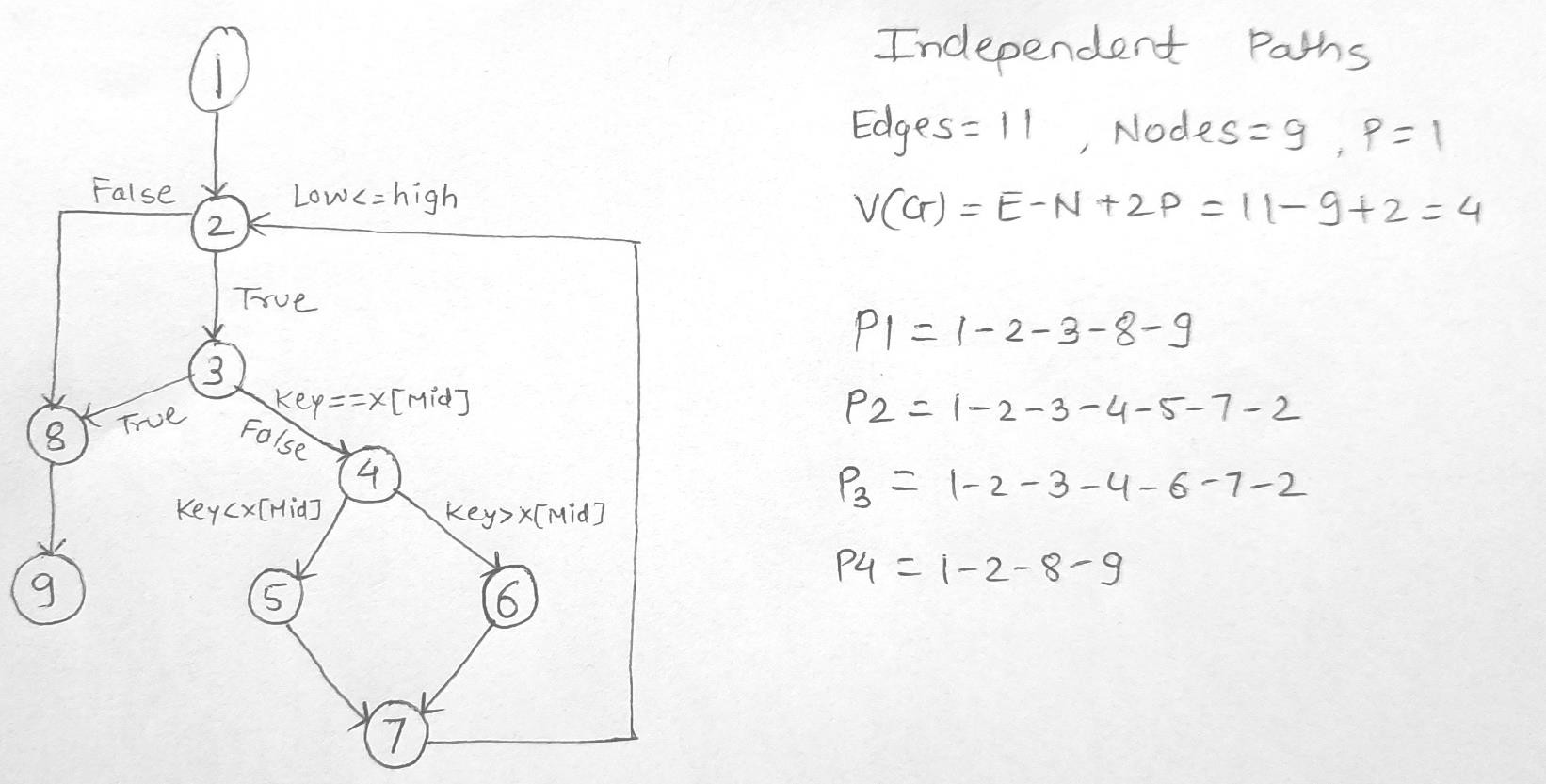
high = mid - 1 return -1

array = input().split(" ") x = input()

result = binarySearch(array, x, 0, len(array)-1) if result != -1:

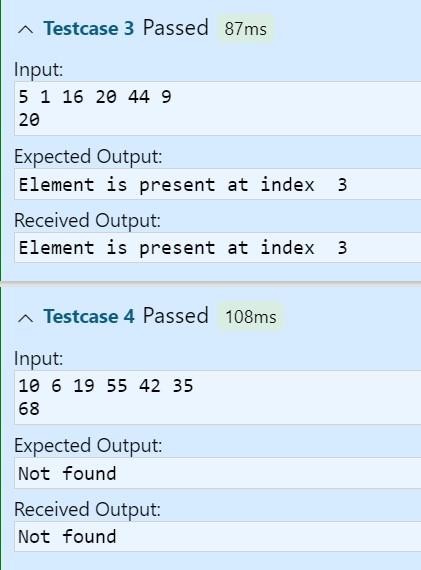
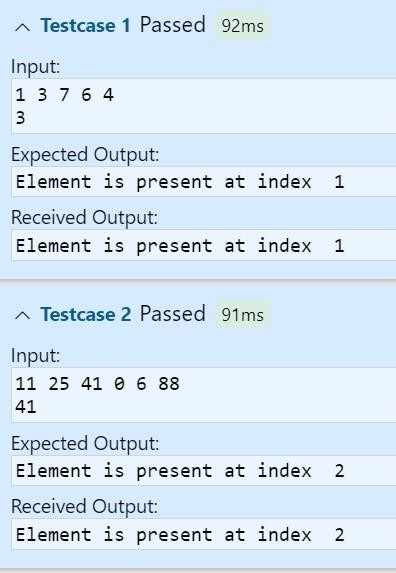
print("Element is present at index ",str(result)) else:

print("Not found")

Program Graph – for Binary Search



# Test Cases:-



**Test Cases – Binary Search**

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | | **Expected Output** | **Actual Output** |
| **X[]** | **key** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Experiment no - 08

**Experiment Name – Insertion Sort Name –Mayur Santosh Hajare Roll No - 31**

**Class - TYIT Batch – T2**

### . Date of Conduction –

**/\*Design, develop, code and run the program in any suitable language to implement the Insertion Sort Algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.\*/**

Program:-

def insertionSort(arr, n): for i in range(1,n):

k = arr[i] j = i - 1

while j >= 0 and k < arr[j]: arr[j+1] = arr[j]

j -= 1

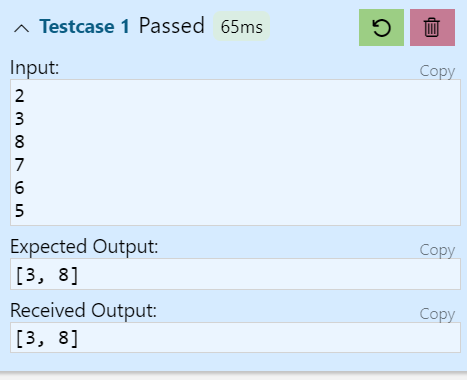
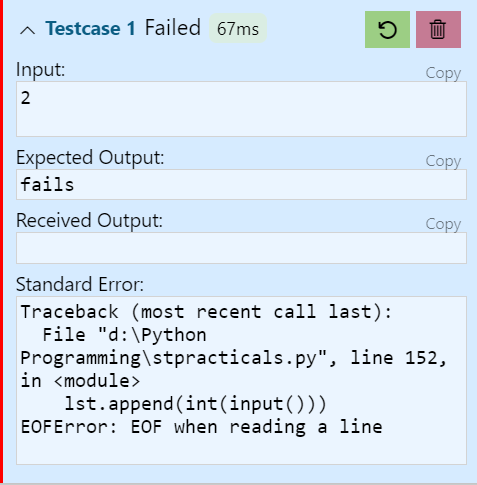
arr[j+1] = k

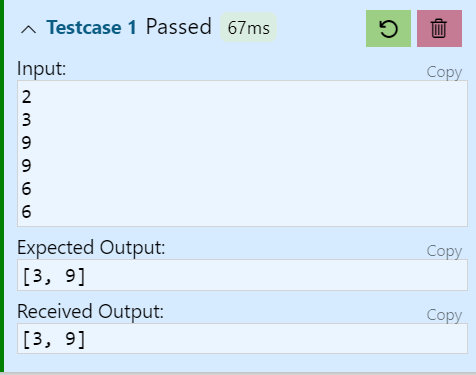
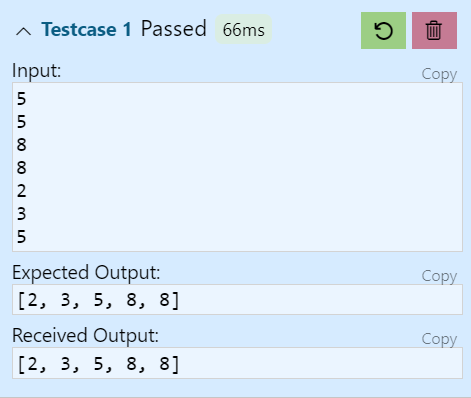
lst = []

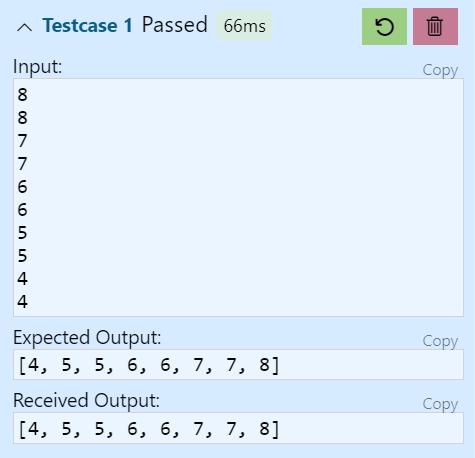
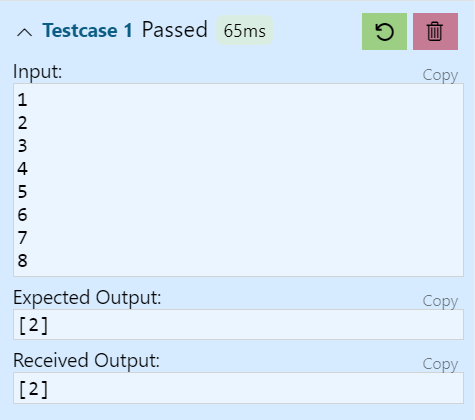
n = int(input()) for i in range(n):

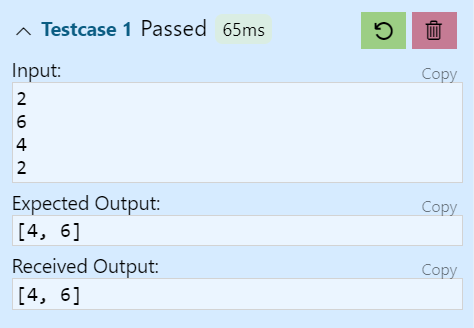
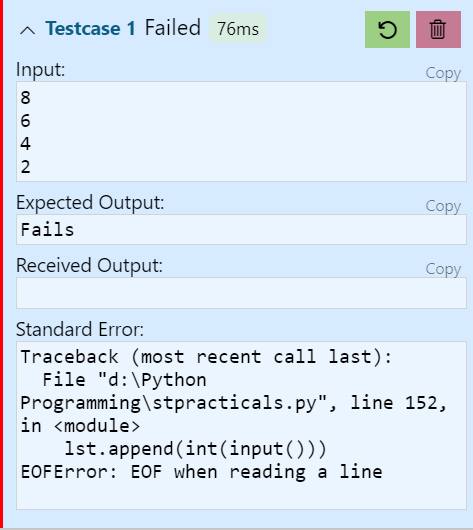
lst.append(int(input())) insertionSort(lst, n) print(lst)

**Test cases:-**





**Test Cases – Insertion Sort**

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | | **Expected Output** | **Remarks** |
| **X[]** | **First, last** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Experiment no - 09

**Experiment Name – Quick Sort - Path Testing Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

/\*Design, develop, code and run the program in any suitable language to implement the Quick-Sort Algorithm. Determine the basis paths and using them derive different test cases, execute these test cases and discuss the test results.\*/

## Program:-

def pivot\_fun(lst, i, l): pivot = lst[i]

left = i+1 right = l while True:

while (left <= right) and (lst[left] <= pivot): left += 1

while (left <= right) and (lst[right] >= pivot): right -= 1

if right < left: break

else:

lst[left],lst[right] = lst[right], lst[left] lst[i], lst[right] = lst[right], lst[i]

return right

def quicksort(lst, i , j): if i < j:

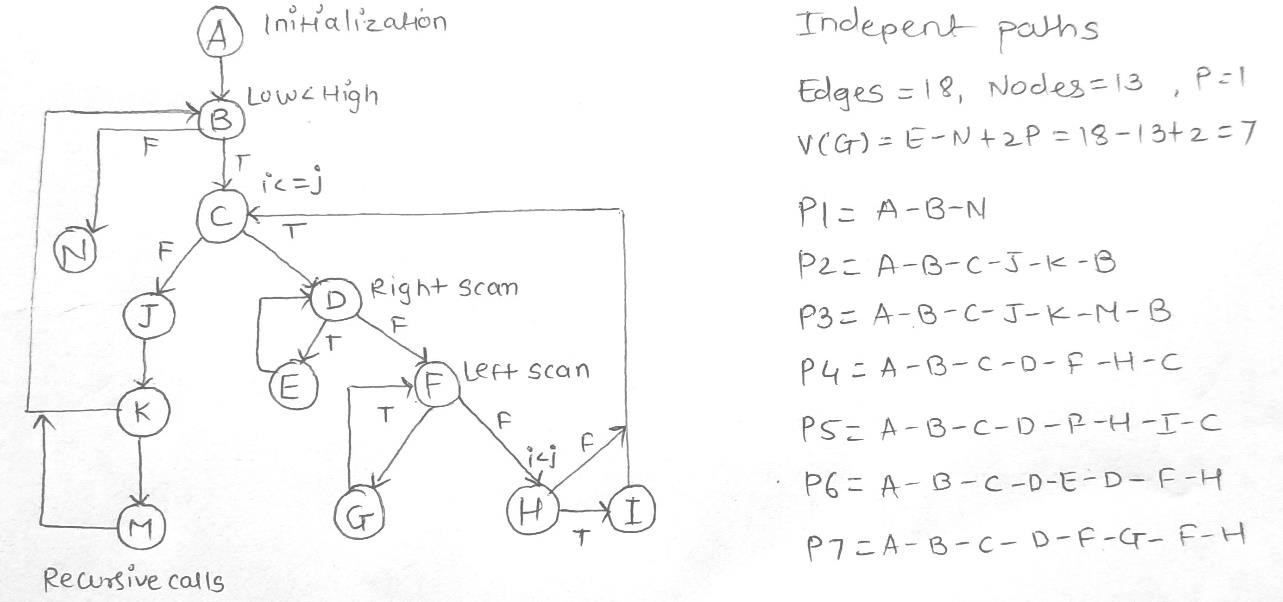
p = pivot\_fun(lst, i, j) quicksort(lst, i, p-1) quicksort(lst, p+1, j)

lst = []

l = int(input()) for i in range(l):

lst.append(int(input())) quicksort(lst,0, l-1) print(lst)

## Program Graph – Quick sort

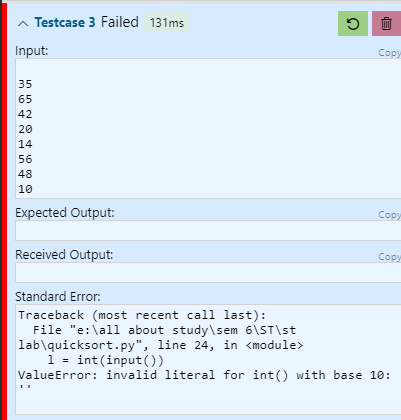
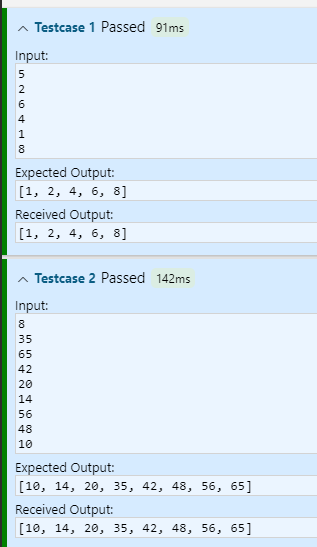


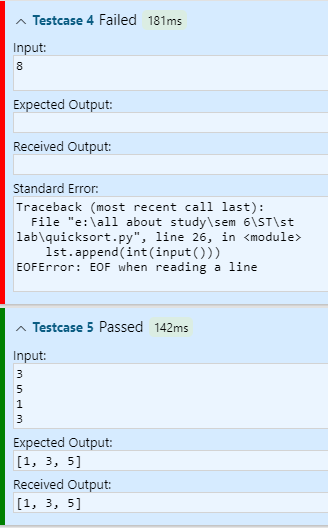
Pre-Conditions/Issues:

Array has only one Element, Two Elements, Three Elements (6 Possibilities)

Array has Elements in ASC/DSC/Arbitrary( Any of the Permutations) EX: 3 elements: 123, 132, 213, 231,312, 321, 222,111,333

Test cases





# Test Cases – Quick Sort

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Paths** | **Inputs** | | **Expected Output** | **Remarks** |
| **X[]** | **First, last** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Experiment no - 10

**Experiment Name – Absolute Letter Grading Path Testing Name – Mayur Santosh Hajare**

## Roll No - 54 Class - TYIT Batch – T4

**Date of Conduction –**

**/\* 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:-

arr = []

for i in range(5): arr.append(int(input()))

sum = sum(arr) per = sum/5 print(per)

if per>90:

print("Destinction") elif per > 75:

print("First class") elif(per >50):

print("Good") else:

print("Fail")

Pre-Conditions/Issues:

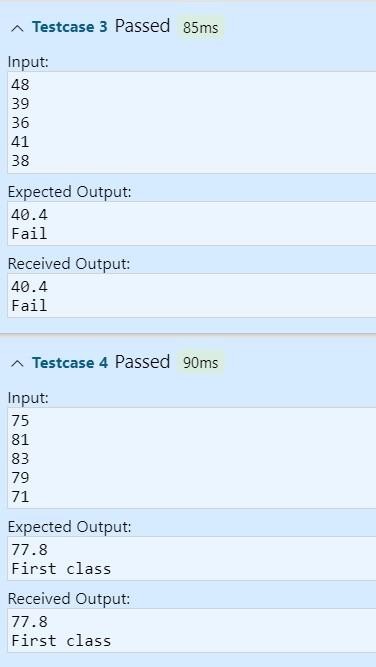
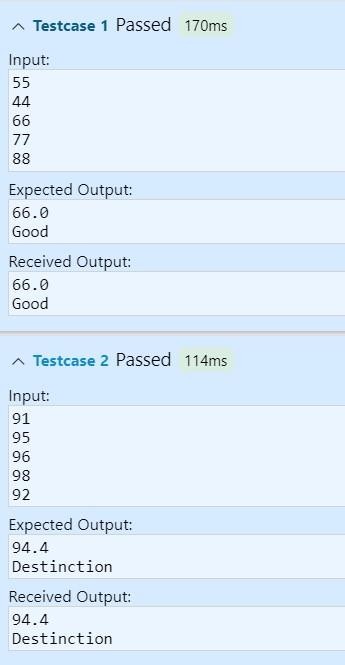
Percentage Per is a positive Float Number

# Test Cases – Absolute Grading

|  |  |  |  |
| --- | --- | --- | --- |
| **Paths** | **Inputs** | **Expected Output** | **Remarks** |
| **Per** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Test Cases :-**



## Experiment no - 11

**Experiment Name – Boundary Value Analysis for Next date program Name – Mayur Santosh Hajare**

## Roll No - 54

**Class - TYIT Batch – T4**

## Date of Conduction –

/\* 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. \*/

Program:-

def leap(year):

if year % 4 == 0 and year % 100 != 0 or year % 400 == 0: return True

else:

return False def check(day, month):

if (month == 4 or month == 6 or month == 9 or month == 11) and day == 31: return True

else:

return False

print()

day = int(input()) month = int(input()) year = int(input()) flag = 'y'

tom\_day = day tom\_month = month tom\_year = year

if day < 1 or day > 31: print("Invalid Day") flag = 'n'

elif month < 1 or month > 12: print("Invalid Month") flag = 'n'

if check(day, month): print("Invalid Day and Month") flag = 'n'

elif year < 1812 or year > 2023: print("Invalid Year")

flag = 'n' if month == 2:

if leap(year) and day > 29: print("Invalid input for Leap Year") flag = 'n'

elif not leap(year) and day > 28: print("Invalid input for non Leap Year") flag = 'n'

if flag == 'y':

if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10: if day < 31:

tom\_day = day + 1 else:

tom\_day = 1 tom\_month = month + 1

if month == 4 or month == 6 or month == 9 or month == 11: if day < 30:

tom\_day = day + 1 else:

tom\_day = 1 tom\_month = month + 1

if month == 12: if day < 31:

tom\_day = day + 1 else:

tom\_day = 1

tom\_month = 1 tom\_year = year + 1

if month == 2: if day < 28:

tom\_day = day + 1

elif leap(year) and day == 28: tom\_day = day + 1

elif day == 28 or day == 29: tom\_day = 1

tom\_month = 3

a= []

a.append(str(tom\_day)) a.append(str(tom\_month)) a.append(str(tom\_year)) if flag == 'n':

print("Enter all the valid entries") else:

print("Next Date is: ", "/".join(a))

Test Case Name : Boundary Value Analysis test cases for Next date program

**Test data :** Enter the three integer value

**Pre-condition :** Month 1 to 12 , DAY 1 TO 31 AND YEAR 1812 TO 2013 / we consider one corner for the input space

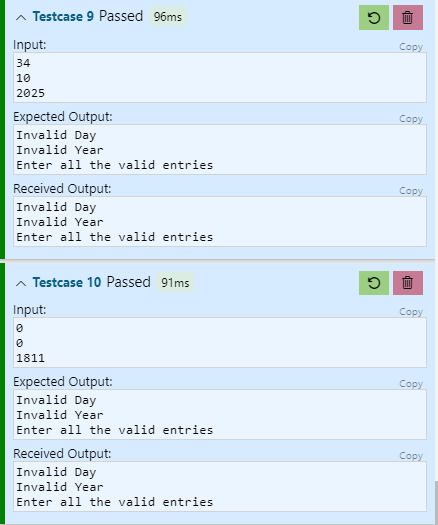
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Min** | **Min +1** | **Normal** | **Max -1** | **Max** |
| **Month** | 1 | 2 | 6 | 11 | 12 |
| **Day** | 1 | 2 | 15 | 30 | 31 |
| **Year** | 1812 | 1813 | 1912 | 2012 | 2013 |

Brief Description :

## Test Cases:-



Next date Output Boundary Value Analysis Cases



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | Actual output | | | Status |
| Month | day | year | Month | day | year | Month | day | year |
| 1 |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  |  |
| 12 |  |  |  |  |  |  |  |  |  |  |  |
| 13 |  |  |  |  |  |  |  |  |  |  |  |
| 14 |  |  |  |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  |  |  |  |  |  |
| 16 |  |  |  |  |  |  |  |  |  |  |  |
| 17 |  |  |  |  |  |  |  |  |  |  |  |
| 18 |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |
| 21 |  |  |  |  |  |  |  |  |  |  |  |
| 22 |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 23 |  |  |  |  |  |  |  |  |  |  |  |
| 24 |  |  |  |  |  |  |  |  |  |  |  |
| 25 |  |  |  |  |  |  |  |  |  |  |  |

Next date Output Special value test cases

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | Actual output | | | Status |
| month | day | year | month | day | year | month | day | year |  |
| 1 |  |  |  |  |  | | |  | | |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  | | |  | | |  |

## Experiment no - 12

**Experiment Name – Equivalence class for Next date Name – Mayur Santosh Hajare**

## Roll No - 54

**Class - TYIT Batch – T4**

## Date of Conduction –

/\* Design, develop, code and run the program in any suitable language to implement the NextDate function. Analyze it from the perspective of equivalence class analysis. Derive different test cases, execute these test cases and discuss the test results. \*/

**Test Case Name : Equivalence class test cases for Next date Test data :** Enter the three integer value

**Pre-condition :** Month 1 to 12 , DAY 1 TO 31 AND YEAR 1812 TO 2013

Valid Cases

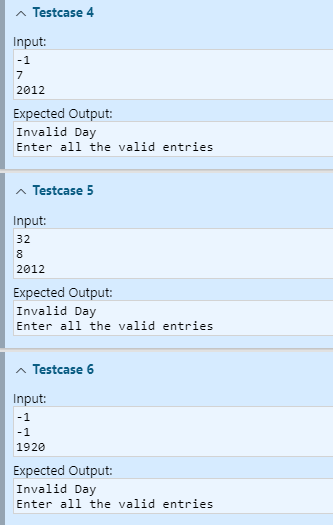
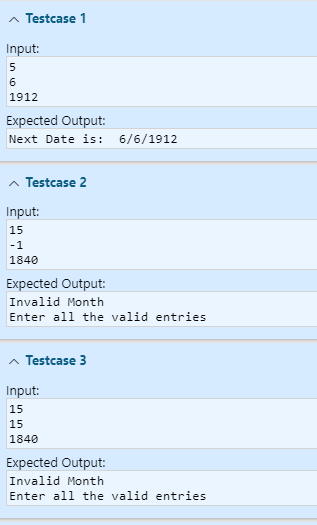
M1 = { month ; 1 ≤ month ≤ 12 } D1 = { day : 1 ≤ day ≤ 31 }

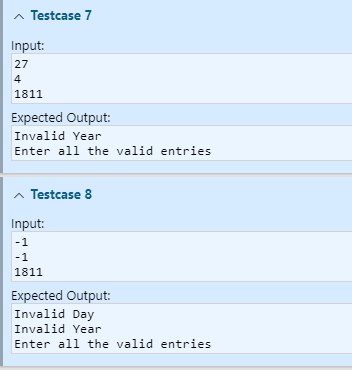
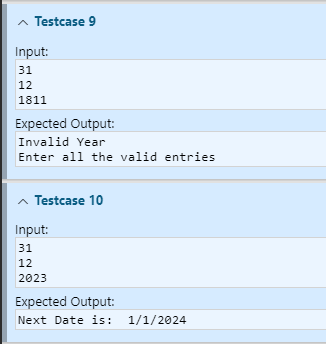
Y1 = { year : 1812 ≤ year ≤ 2013 }

**Invalid cases** M2 = {month : month < 1} M3 = {month : month > 12} D2 = {day : day < 1} D3 = {day : day > 31}

Y2 = {year : year < 1812} Y3 = {year : year > 2013}

## Test Cases:-



Next date Output Equivalence Class Testing ( Weak and Strong Normal Equivalence Class )

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | Actual output | | | Status |
| month | day | year | month | day | year | month | day | year |
| WN1, SN1 |  |  |  |  |  |  |  |  |  |  |  |

( Weak Robustness Equivalence Class )

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | Actual output | | | Status |
| month | day | year | month | day | year | mon  th | day | year |  |
| WR1 |  |  |  |  |  |  |  |  |  |  |  |
| WR2 |  |  |  |  |  | | |  | | |  |
| WR3 |  |  |  |  |  | | |  | | |  |
| WR4 |  |  |  |  |  | | |  | | |  |
| WR5 |  |  |  |  |  | | |  | | |  |
| WR6 |  |  |  |  |  | | |  | | |  |
| WR7 |  |  |  |  |  | | |  | | |  |

(Strong Robustness Equivalence Class )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | Actual Output | Status |
| month | day | year |
| SR1 |  |  |  |  |  |  |  |
| SR2 |  |  |  |  |  |  |  |
| SR3 |  |  |  |  |  |  |  |
| SR4 |  |  |  |  |  |  |  |
|  |  |  |
| SR5 |  |  |  |  |  |  |  |
|  |  |  |
| SR6 |  |  |  |  |  |  |  |
|  |  |  |
| SR7 |  |  |  |  |  |  |  |
|  |  |  |
|  |  |  |

Some addition equivalence Boundary checking

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Case Id | Description | Input Data | | | Expected Output | | | Actual Output | | | Status |
| day | month | year | day | month | year | day | month | year |
| 1 |  |  |  |  |  | | |  | | |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  | | |  | | |  |