10) Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters .

Sample String: ‘Sona College of Technology’ ,

Expected Output:

No. of Upper case characters: 3

No. of Lower case Characters: 20.

Sample Code:

def string\_test(s):

    d={"UPPER\_CASE":0, "LOWER\_CASE":0}

    //Try your logic

    print ("No. of Upper case characters : ", d["UPPER\_CASE"])

    print ("No. of Lower case Characters : ", d["LOWER\_CASE"])

s = input()

string\_test(s)

AIM: Write a Python function that accepts a string and calculate the number of upper case letters and lower case letters .

PROGRAM:

def string\_test(s):

d={"UPPER\_CASE":0, "LOWER\_CASE":0}

for i in range(len(s)):

if(s[i].isupper()):

d["UPPER\_CASE"]+=1

elif(s[i].islower()):

d["LOWER\_CASE"]+=1

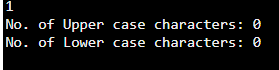
print("No. of Upper case characters:",d["UPPER\_CASE"])

print("No. of Lower case characters:",d["LOWER\_CASE"])

s=input()

p=string\_test(s)

LINK: <http://103.53.53.18/mod/vpl/forms/submissionview.php?id=325&userid=1646>

OUTPUT: 

RESULT: Thus the given python function that accepts a string and calculate the number of upper case letters and lower case letters is executed.

11) Write a Python program to find the greatest common divisor (gcd) of two integers using recursion.

Sample Code:

def gcd(a,b):

   //try your logic

a=int(input("Enter first number:"))

b=int(input("Enter second number:"))

GCD=gcd(a,b)

print("GCD is: ")

print(GCD)

AIM: Write a Python program to find the greatest common divisor (gcd) of two integers using recursion.

4

PROGRAM: def gcd(a,b):

if(a==0):

return a

if(b==0):

return b

if(a==b):

return b

if(a>b):

return(gcd(a-b,b))

else:

return gcd(a,b-a)

a=int(input("Enter first number:"))

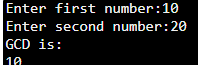
b=int(input("Enter second number:"))

GCD=gcd(a,b)

print("GCD is:")

print(GCD)

LINK: <http://103.53.53.18/mod/vpl/forms/submissionview.php?id=326&userid=1646>

OUTPUT: 

RESULT:Thus the given Python program to find the greatest common divisor (gcd) of two integers using recursion executed.

13) An apparel shop wants to manage the items which it sells.25 min  
Write a python program to implement the class diagram given below.

**Class Description:  
Apparel class:**

1. Initialize static variable counter to 100
2. In the constructor, auto-generate item\_id starting from 101 prefixed by "C" for cotton apparels and "S" for silk apparels. Example – C101, S102, S103, C104 etc.
3. **calculate\_price():** Add 5% service tax on the price of the apparel and update attribute, price with the new value

**Cotton class:**

1. While invoking parent constructor from child constructor, pass "Cotton" as item\_type
2. **calculate\_price():** Update attribute, price of Apparel class based on rules given below
   1. Add service tax on price by invoking appropriate method of Apparel class
   2. Apply discount on price
   3. Add 5% VAT on final price

AIM: Write a python program to implement the class diagram given below .

PROGRAM: class Apparel:

counter=100

def \_\_init\_\_(self,price,item\_type):

Apparel.counter+=1

self.\_\_item\_id=item\_type[0]+str(Apparel.counter)

self.\_\_price=price

self.\_\_item\_type=item\_type

def calculate\_price(self):

self.\_\_price+=self.\_\_price\*0.05

def get\_item\_id(self):

return self.\_\_item\_id

def get\_price(self):

return self.\_\_price

def get\_item\_type(self):

return self.\_\_item\_type

def set\_price(self,price):

self.\_\_price=price

return self.\_\_price

class Cotton(Apparel):

def \_\_init\_\_(self,price,discount):

super().\_\_init\_\_(price,'Cotton')

self.\_\_discount=discount

def calculate\_price(self):

super().calculate\_price()

price=self.get\_price()

price-=price\*(self.\_\_discount/100)

price+=price\*0.05

self.set\_price(price)

return price

def get\_discount(self):

return self.\_\_discount

class Silk(Apparel):

def \_\_init\_\_(self,price):

super().\_\_init\_\_(price,'Silk')

self.\_\_points=None

def calculate\_price(self):

super().calculate\_price()

if(self.get\_price()>10000):

self.\_\_points=10

else:

self.\_\_points=3

return self.set\_price(self.get\_price()+(self.get\_price()\*0.1))

def get\_points(self):

return self.\_\_points

silk=int(input())

cotton=int(input())

discount=int(input())

a=Silk(silk)

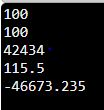
print(a.calculate\_price())

b=Cotton(cotton,discount)

print(b.calculate\_price())

LINK: <http://103.53.53.18/mod/vpl/forms/submissionview.php?id=328&userid=1646>

OTUPUT:



RESULT:Thus the given python program to implement the class diagram given below is executed.

14) Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order, for

example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.

input: ()[]{()}

output: valid

input: {{

output: invalid

Sample Code

def areBracketsBalanced(expr):

    stack = []

    //Try your logic

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_":

    expr = input()

    # Function call

    if areBracketsBalanced(expr):

        print("valid")

    else:

        print("invalid")

AIM: Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These brackets must be close in the correct order, for

example "()" and "()[]{}" are valid but "[)", "({[)]" and "{{{" are invalid.

PROGRAM: def areBracketsBalanced(expr):

stack=[]

open\_expressions=['{','[','(']

for i in range(len(expr)):

if(expr[i] in open\_expressions):

stack.append(expr[i])

else:

if not stack:

return False

curr=stack.pop()

if(curr == '{' and expr[i]!='}'):

return False

if(curr == '[' and expr[i]!=']'):

return False

if(curr == '(' and expr[i]!=')'):

return False

if stack:

return False

else:

return True

if \_\_name\_\_== "\_\_main\_\_":

expr=input()

if(areBracketsBalanced(expr)):

print("valid")

else:

print("invalid")

LINK: <http://103.53.53.18/mod/vpl/forms/submissionview.php?id=327&userid=1646>

OUTPUT: 

RESULT:Thus the given python class to find validity of a string of parentheses is executed.

4