

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

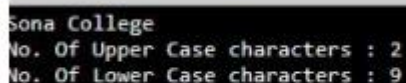
**AIM:**

To accept 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 c in s:  
        if c.isupper():  
            d["UPPER_CASE"]+=1  
        elif c.islower():  
            d["LOWER_CASE"]+=1  
        else:  
            pass  
    print("No. Of Upper Case characters :",d["UPPER_CASE"])  
    print("No. Of Lower Case characters :",d["LOWER_CASE"])  
string_test(input())
```

**OUTPUT:**

A screenshot of a terminal window showing the output of the Python program. The text displayed is: "Sona College", "No. Of Upper Case characters : 2", and "No. Of Lower Case characters : 9".

```
Sona College  
No. Of Upper Case characters : 2  
No. Of Lower Case characters : 9
```

**LINK:**

<http://103.53.53.18/mod/vpl/forms/submissionview.php?id=325&userid=1653>

**RESULT:**

Thus the 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.

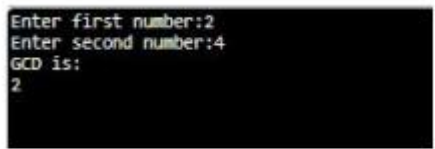
**AIM:**

To find the greatest common divisor (gcd) of two integers using recursion.

**PROGRAM:**

```
def gcd(x,y):  
    gcd=1  
    if x % y==0:  
        return y  
    for k in range(int(y/2),0,-1):  
        if x % k ==0 and y % k ==0:  
            gcd=k  
            break  
    return gcd  
x=int(input("Enter first number:"))  
y=int(input("Enter second number:"))GCD=gcd(x,y)  
print("GCD is:")  
print(GCD)
```

**OUTPUT:**



```
Enter first number:2  
Enter second number:4  
GCD is:  
2
```

**LINK:**

**RESULT:**

Thus the python program to find the greatest common divisor (gcd) of two integers using recursion.

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.

**AIM:**

To manage the items which it sells.25 min  
Write a python program to implement the class

**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())
```

## OUTPUT:

```
10000
10
15
11550.0
9.37125
```

## LINK:

<http://103.53.53.18/mod/vpl/forms/submissionview.php?id=328&userid=1653>

## RESULT:

Thus the python program to implement the class diagram 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.

### AIMI

To find validity of a string of parentheses.

### PROGRAM:

```
def valid_paren(input_str):
    stack=[]
    for paren in input_str:
        if paren == '(' or paren == '[' or paren == '{':
            stack.append(paren)
```

```
else:
if not stack:
print("invalid")
return
else:
top=stack[-1]
if paren ==')' and top =='(' or ¥
paren ==']' and top =='[' or ¥
paren =='}' and top =='{' :
stack.pop()
if not stack:
print("valid")
else:
print("invalid")input1=input()
valid_paren(input1)
```

### OUTPUT:

A terminal window with a black background. The input string '(){}[]' is entered on the first line. On the second line, the word 'valid' is printed in white text.

### LINK:

<http://103.53.53.18/mod/vpl/forms/submissionview.php?id=327&userid=1653>

### RESULT:

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