**1.**Python program to calculate the number of lowercase and uppercase letters in a string

## Description:

Take a string as input and find the number of uppercase and lower case letters in the string and print the count.

Note: any spaces has to be ignored Eg: string is : 'This is Python' Upper case characters : 2

Lower case characters : 10 string=input("Enter string:") count1=0

count2=0

for i in string: if(i.islower()):

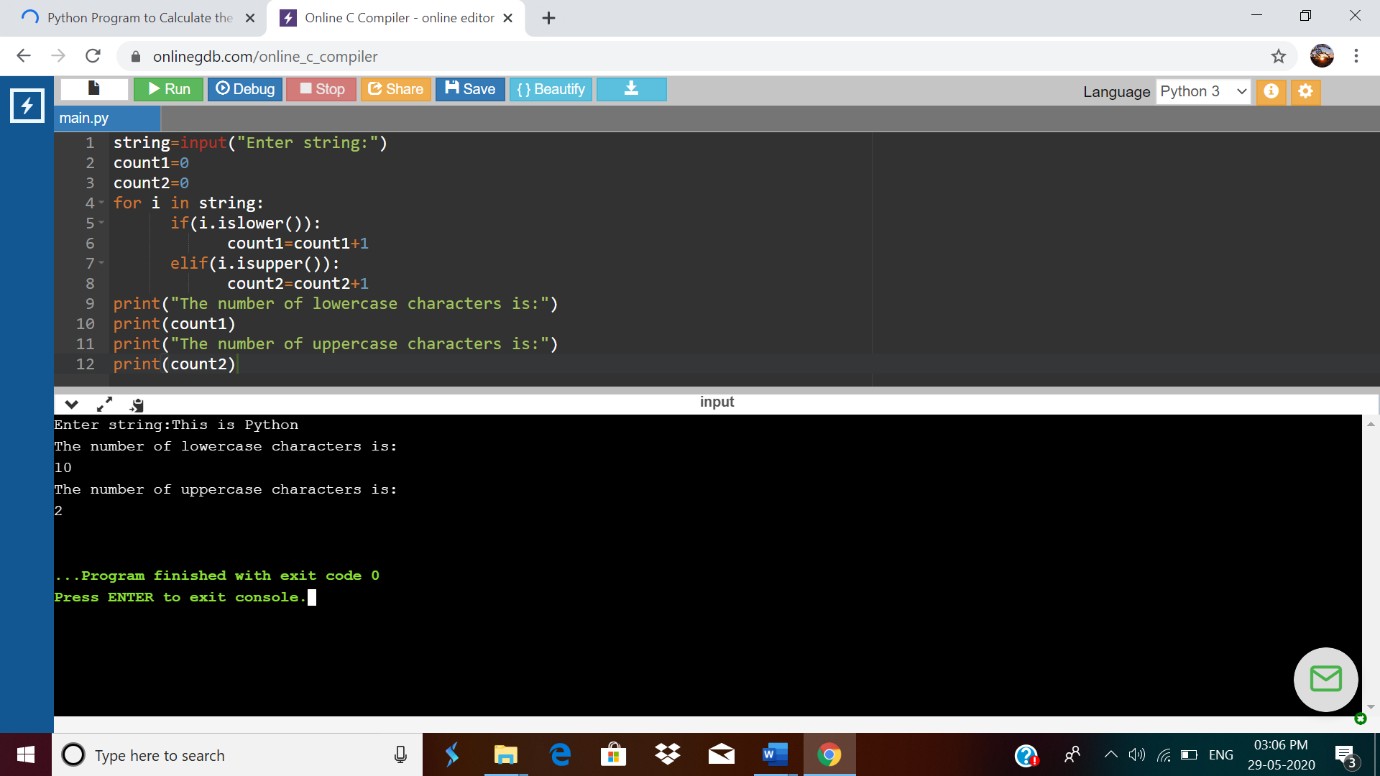
count1=count1+1

elif(i.isupper()): count2=count2+1

print("The number of lowercase characters is:") print(count1)

print("The number of uppercase characters is:") print(count2)

**output:**



2.We are given 3 strings: str1, str2, and str3. Str3 is said to be a shuffle of str1 and str2 if it can be formed by interleaving the characters of str1 and str2 in a way that maintains the left to right ordering of the characters from each string. For example, given str1="abc" and str2="def", str3="dabecf" is a valid shuffle since it preserves the character ordering of the two strings. So, given these 3 strings write a function that detects whether str3 is a valid shuffle of str1 and str2.

public class Main{

static boolean isInterleaved (String A, String B, String C)

{

int i = 0, j = 0, k = 0; while (k != C.length())

{

if (i<A.length()&&A.charAt(i) == C.charAt(k)) i++;

else if (j<B.length()&&B.charAt(j) == C.charAt(k)) j++;

else

return false;

k++;

}

if (i < A.length() || j < B.length()) return false;

return true;

}

public static void main(String []args){

String A = "abc"; String B = "def"; String C = "dabecf";

if (isInterleaved(A, B, C) == true)

System.out.printf("%s is interleaved of %s and %s", C, A, B);

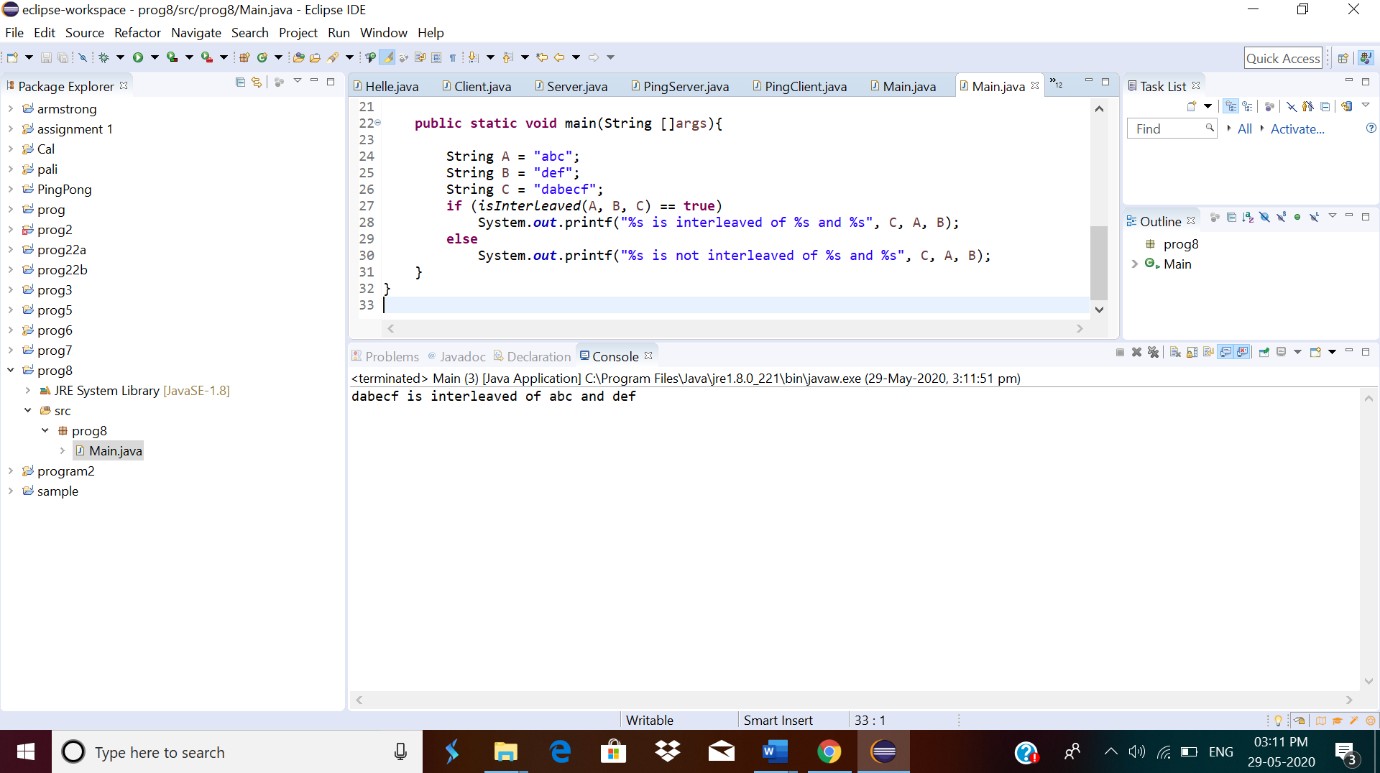
else

System.out.printf("%s is not interleaved of %s and %s", C, A, B);

}

}

# Output:



## 3.Write a c program to solve a system of linear congruences by applying the Chinese Remainder Theorem.

#include <stdio.h>

int mul\_inv(int a, int b)

{

int b0 = b, t, q; int x0 = 0, x1 = 1;

if (b == 1) return 1;

while (a > 1) {

q = a / b;

t = b, b = a % b, a = t;

t = x0, x0 = x1 - q \* x0, x1 = t;

}

if (x1 < 0) x1 += b0; return x1;

}

int chinese\_remainder(int \*n, int \*a, int len)

{

int p, i, prod = 1, sum = 0;

for (i = 0; i < len; i++) prod \*= n[i];

for (i = 0; i < len; i++) {

p = prod / n[i];

sum += a[i] \* mul\_inv(p, n[i]) \* p;

}

return sum % prod;

}

int main(void)

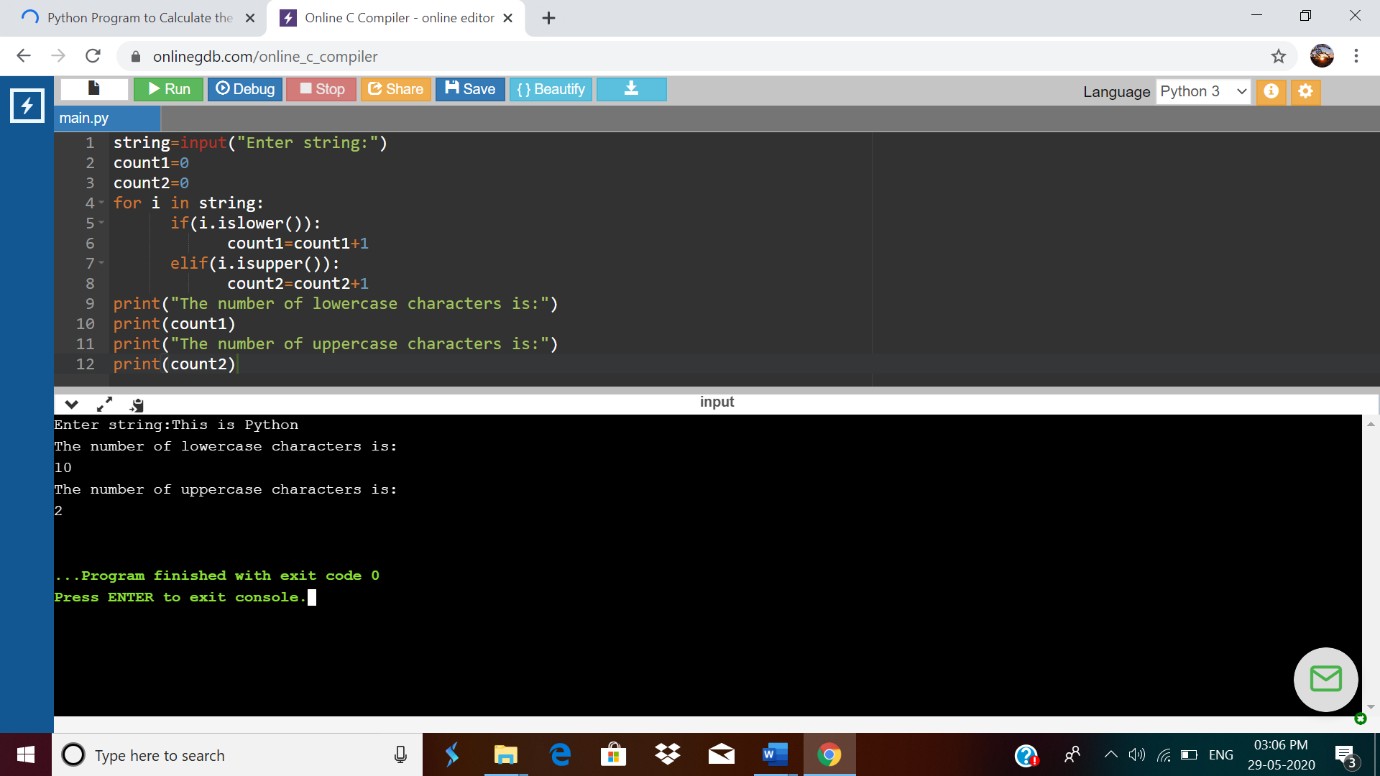
{

elif(i.isupper()): count2=count2+1

print("The number of lowercase characters is:") print(count1)

print("The number of uppercase characters is:") print(count2)

**output:**



4.We are given 3 strings: str1, str2, and str3. Str3 is said to be a shuffle of str1 and str2 if it can be formed by interleaving the characters of str1 and str2 in a way that maintains the left to right ordering of the characters from each string. For example, given str1="abc" and str2="def", str3="dabecf" is a valid shuffle since it preserves the character ordering of the two strings. So, given these 3 strings write a function that detects whether str3 is a valid shuffle of str1 and str2.

public class Main{

static boolean isInterleaved (String A, String B, String C)

{

int i = 0, j = 0, k = 0; while (k != C.length())

{

if (i<A.length()&&A.charAt(i) == C.charAt(k)) i++;

else if (j<B.length()&&B.charAt(j) == C.charAt(k)) j++;

else

return false;

k++;

}

if (i < A.length() || j < B.length()) return false;

return true;

}

public static void main(String []args){

String A = "abc"; String B = "def"; String C = "dabecf";

if (isInterleaved(A, B, C) == true)

System.out.printf("%s is interleaved of %s and %s", C, A, B);

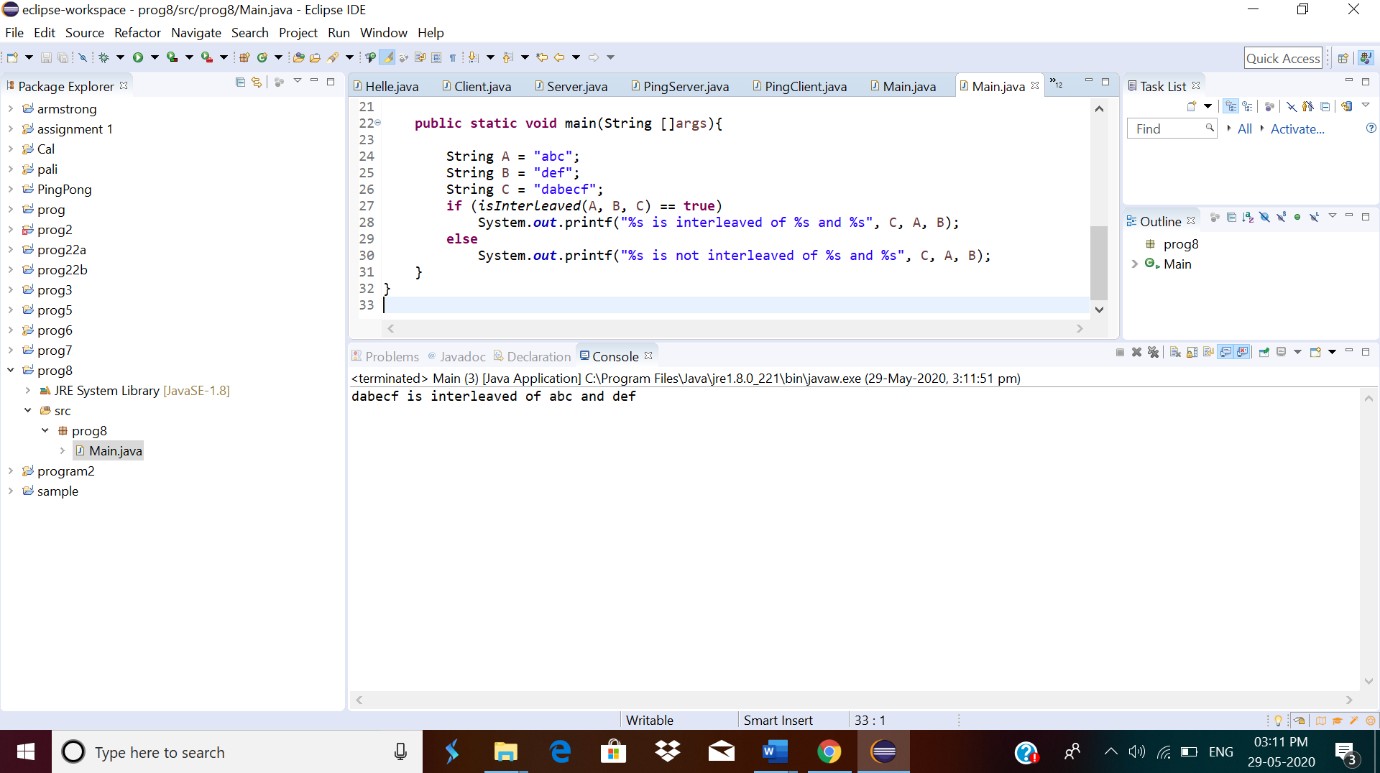
else

System.out.printf("%s is not interleaved of %s and %s", C, A, B);

}

}

# Output:



## 5.Write a c program to solve a system of linear congruences by applying the Chinese Remainder Theorem.

#include <stdio.h>

int mul\_inv(int a, int b)

{

int b0 = b, t, q; int x0 = 0, x1 = 1;

if (b == 1) return 1;

while (a > 1) {

q = a / b;

t = b, b = a % b, a = t;

t = x0, x0 = x1 - q \* x0, x1 = t;

}

if (x1 < 0) x1 += b0; return x1;

}

int chinese\_remainder(int \*n, int \*a, int len)

{

int p, i, prod = 1, sum = 0;

for (i = 0; i < len; i++) prod \*= n[i];

for (i = 0; i < len; i++) {

p = prod / n[i];

sum += a[i] \* mul\_inv(p, n[i]) \* p;

}

return sum % prod;

}

int main(void)

{

int n[] = { 3, 5, 7 };

int a[] = { 2, 3, 2 };

printf("%d\n", chinese\_remainder(n, a, sizeof(n)/sizeof(n[0]))); return 0;

}

Output:

