Frequency of numbers in a string

In [3]:
```
# Frequency of every digit in a string
# s = 123andj48739292
n=input()
for i in range(0,10):
    print(n.count(str(i)),end=' ')
```

```
sjkls72828273hjsje
0 0 3 1 0 0 0 2 2 0
```

In [1]:
```
def digitFrequency2(s):
    for i in range(0,10):
        count = s.count(str(i))
        print(count, end=' ')
    return
digitFrequency2('213abc456def111')
```

```
0 4 1 1 1 1 1 0 0 0
```

Problem Statement

- Given a string, s , consisting of alphabets and digits, find the frequency of each digit in the given string.
- Input Format
  - The first line contains a string, num, which is the given number.
- Constraints
  - 1<=len(num)<=1000
  - All the elements of num are made of english alphabets and digits.
- Output Format
  - Print ten space-separated integers in a single line denoting the frequency of each digit from 0 to 9

In [9]:
```
# s = 123abc456def
# 0 1 1 1 1 1 1 0 0 0 --> frequency of sorted numbers

# s = c
# 0 0 0 0 0 0 0 0 0 0

# s 1234567890
# 1 1 1 1 1 1 1 1 1 1

def uniqueData(allnumbers):
    unique = []
    for n in allnumbers:
        if n not in unique:
            unique.append(n)
    return unique

def digitFrequency1(s):
    allnumbers = []
    for i in s:
        if i.isdigit():
            allnumbers.append(i)
    unique=uniqueData(allnumbers)
    for i in range(0,10):
        if str(i) not in unique:
            print(0,end=' ')
        else:
            count = allnumbers.count(str(i))
            print(count, end=' ')


digitFrequency1('2938729198281919919288829')

```

```
0 4 5 1 0 0 0 1 6 8
```

In [9]:
```
# Generation of marks
from random import randint
def generatemarks(n,lb,ub):
    filepath='DataFiles/marks.txt'
    with open(filepath,'w') as f:
        for i in range(0,n):
            r=randint(lb,ub)
            f.write(str(r)+'\n')
    print(n,'marks stored in file ')
generatemarks(30,1,100)
```

```
30 marks stored in file
```

In [10]:
```
# def classAverage(filepath):
#     sum=0
#     count=0
#     with open(filepath,'r') as f:
#         for i in f:
#             sum=sum+int(i)
#             count=count+1
#         print(sum/count)

# classAverage('DataFiles/marks.txt')
```

```
60.1
```

In [11]:
```
# # % passed percentage
# def passpercentage(filepath):
#     count=0
#     sum=0
#     with open(filepath,'r') as f:
#         for i in f:
#             if(int(i)>=35):
#                 sum=sum+int(i)
#                 count=count+1
#         print(sum/count)
# passpercentage('DataFiles/marks.txt')
```

```
71.43478260869566
```

In [16]:
```
# # % Pass
# def Passpercentage(filepath):
#     count=0
#     mc=0
#     with open(filepath,'r') as f:
#         for i in f:
#             mc=mc+1
#             if(int(i)>=35):
#                 count=count+1
#         return ((count/mc)*100)
# Passpercentage('DataFiles/marks.txt')
```

Out[16]:
```
76.66666666666667
```

In [17]:
```
# # % Fail Percen
# def failper(filepath):
#     fcount=100 - Passpercentage(filepath)
#     return (fcount)

# failper('DataFiles/marks.txt')
```

```
23.33333333333333
```

In [13]:
```
# def failpercentage(filepath):
#     count=0
#     mc=0
#     with open(filepath,'r') as f:
#         for i in f:
#             mc=mc+1
#             if(int(i)<35):
#                 count=count+1
#         return ((count/mc)*100)
# failpercentage('DataFiles/marks.txt')
```

```
23.333333333333332
```

In [22]:
```
# # Frequency of highest marks -->
# def frequencyHighest(filepath):
#     with open(filepath,'r') as f:
#         sp=f.read().split()
#         sp=list(map(int,sp))
#         # sp = list(map(str,sp))  # --> can convert int of list element in
#         #print(sp) # --> printing the list of int values of list
#         return (sp.count(max(sp)))
#     #return
# frequencyHighest('DataFiles/marks.txt')
```

```
1
```

In [4]:
```
# # #% Pass
# def Disctionpercentage(filepath):
#     count=0
#     mc=0
#     with open(filepath,'r') as f:
#         for i in f:
#             mc=mc+1
#             if(int(i)>75):
#                 count=count+1
#         return ((count/mc)*100)
# Disctionpercentage('DataFiles/marks.txt')
```

In [26]:
```
# # Frequency of lowest marks -->
# def frequencyLowest(filepath):
#     with open(filepath,'r') as f:
#         sp=f.read().split()
#         sp=list(map(int,sp))
#         # sp = list(map(str,sp))  # --> can convert int of list element in
#         #print(sp) # --> printing the list of int values of list
#         return (sp.count(min(sp)))
#     return
# frequencyLowest('DataFiles/marks.txt')
```

```
2
```

In [1]:

```
# Generation of marks
from random import randint
def generatemarks(n,lb,ub):
    filepath='DataFiles/marks.txt'
    with open(filepath,'w') as f:
        for i in range(0,n):
            r=randint(lb,ub)
            f.write(str(r)+'\n')
    #print(n,'marks stored in file ')
#generatemarks(30,1,100)

def classAverage(filepath):
    sum=0
    count=0
    with open(filepath,'r') as f:
        for i in f:
            sum=sum+int(i)
            count=count+1
        return (sum/count)

#classAverage('DataFiles/marks.txt')


# % Pass
def Passpercentage(filepath):
    count=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if(int(i)>=35):
                count=count+1
        return ((count/mc)*100)
#Passpercentage('DataFiles/marks.txt')

def failpercentage(filepath):
    count=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if(int(i)<35):
                count=count+1
        return ((count/mc)*100)
#failpercentage('DataFiles/marks.txt')

# Frequency of highest marks -->
def frequencyHighest(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        # sp = list(map(str,sp))  # --> can convert int of list element into
        #print(sp) # --> printing the list of int values of list
        return (sp.count(max(sp)))
    #return
#frequencyHighest('DataFiles/marks.txt')
```

```
57
58 # Frequency of lowest marks -->
59 def frequencyLowest(filepath):
60     with open(filepath,'r') as f:
61         sp=f.read().split()
62         sp=list(map(int,sp))
63         # sp = list(map(str,sp))  # --> can convert int of list element into
64         #print(sp) # --> printing the list of int values of list
65         return (sp.count(min(sp)))
66     #return
67 #frequencyLowest('DataFiles/marks.txt')
68
69 # #% disction
70 def Disctionpercentage(filepath):
71     count=0
72     mc=0
73     with open(filepath,'r') as f:
74         for i in f:
75             mc=mc+1
76             if(int(i)>75):
77                 count=count+1
78         return ((count/mc)*100)
79 #Disctionpercentage('DataFiles/marks.txt')
```

In [3]:
```
def marksAnalysis(filepath):
    while True:
        n=int(input("Choose option :\n1).Generation of Marks\n2).Class Avera
        if(n==1):
            st=int(input())
            generatemarks(st,1,100)
        elif(n==2):
            print(classAverage(filepath))
        elif(n==3):
            print(Passpercentage(filepath))
        elif(n==4):
            print(failpercentage(filepath))
        elif(n==5):
            print(Disctionpercentage(filepath))
        elif(n==6):
            print(frequencyHighest(filepath))
        elif(n==7):
            print(frequencyLowest(filepath))
        else:
            break
    return
marksAnalysis('DataFiles/marks.txt')
```

```
Choose option :
1).Generation of Marks
2).Class Average
3). % of pass
 4). % of fail
5).% Disction
 6).Highest Frequency
7). Lowest Frequency
1
50
Choose option :
1).Generation of Marks
2).Class Average
3). % of pass
 4). % of fail
5).% Disction
 6).Highest Frequency
7). Lowest Frequency
5
24.0
Choose option :
1).Generation of Marks
2).Class Average
3). % of pass
 4). % of fail
5).% Disction
 6).Highest Frequency
7). Lowest Frequency
8
```

Contacts application

- addContact(name,phone,email)
- searchContact(name)
- listAllContacts()
- editContact(name,newphone,newemail)
- deleteContact(name)
- contactsApp()

In [10]:

```
filename='DataFiles/contacts.txt'
import re
def phnovalidator(no):
    pattern='[9876][0-9]{9}$|^[0][6-9][0-9]{9}$|^[+][9][1][6-9][0-9]{9}$'
    if re.match(pattern,str(no)):
        return True
        #print("valid number")
    else:
        return False
        #print("invalid number")
    return
def emailval(email):
    pattern='^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z][@][0-9a-z]{3,18}[.][a-z]{2,4}
    if re.match(pattern,email):
        #print("valid")
        return True
    else:
        #print("invalid")
        return False
def csvtolist(filename):
    li=[]
    with open(filename,'r') as f:
        for line in f:
            li.append(line.split(','))
    return li


def listofile(li):
    s=''
    for i in li:
        s+=','.join(i)
    return(s)


def addcontact(name,no,email):
    with open(filename,'a') as f:
        if not searchcontact(name):
            if phnovalidator(int(no)):
                if emailval(email):
                    f.write(name+','+str(no)+','+email+'\n')
                    print("contact added successfully")
        else:
            print("invalid,")

def searchcontact(name):
    fh=csvtolist(filename)
    for line in fh:

        if line[0]==name:

            #print(line[0],":",line[1],",",line[2])
            print("contact exists")
            return True
    print("contact does not exists" )
    return False

```

```
def searchwithreturn(name):
    fh=csvtolist(filename)
    for line in range(len(fh)):
        if fh[line][0]==name:
            #print(line[0],":",line[1],",",line[2])
            #print("contact exists")
            return line
    return -1

def delcon(name):
    a=searchwithreturn(name)
    print(a)
    if a!=-1:
        fh=csvtolist(filename)
        fh.pop(a)
        fh=listofile(fh)
        with open(filename,'w') as f:
            f.write(fh)
        print("deleted contact  successfully")
    else:
        print("contact not found")

def listallcontacts():
    with open(filename,'r') as f:
        f=csvtolist(filename)
        for line in f:
            print("Name : ",line[0])
            print("Phone no : ",line[1],end=" ")
            print()
            print("Email-id : ",line[2],end=" ")
#listallcontacts()
def modifycontact(name,no,email):
    with open(filename,'r') as f:
        i=searchwithreturn(name)
        if i!=-1:
            fh=csvtolist(filename)

            for i in range(len(fh)):
                if fh[i][0]==name:
                    fh[i][1]=str(no)
                    fh[i][2]=email
                fa=listofile(fh)
                with open(filename,'w') as f:
                    f.write(fa)
            print("modified successfully")
        else:
            print("contact does not exists to modify")


while 1:
    print("1.addcontact\n2.searchcontact\n3.deletecontact\n4.listallcontacts

    n=int(input("enter your choice"))

    if n==1:

        name,no,email=input("enter the name,no,email").split(',')
```

```
        addcontact(name,no,email)
    elif n==2:
        name=input("enter the name")
        searchcontact(name)
    elif n==3:
        name=input("enter the name")
        delcon(name)
    elif n==4:
        listallcontacts()
    elif n==5:
        name,no,email=input("enter the name,no,email").split(',')
        modifycontact(name,no,email)

    print("\n press 1 to continue\n0 to exit")
    p=int(input())
    if p==0:
        break
    elif p==1:
        continue
```

```
1.addcontact
2.searchcontact
3.deletecontact
4.listallcontacts
5.modifycontact
enter your choice2
enter the namesiri
contact exists

 press 1 to continue
0 to exit
0
```

Find and Replace application

- Check If word is existing
- wordExists(filepath,word)
- Count the total number of occurances of a word
- countWordOccurances(filepath,word)
- Replace all occurances of a word in a file with another word
- replaceWord(filepath,word)

In [ ]:
```
def isWordExist(filepath,word):
    filedata = readFile1(filepath)
    if word in filedata:
        return True
    else:
        return False
    return

def countWordOccurances(filepath,word):
    filedata = readFile1(filepath)
    count = 0
    for i in filedata:
        if word == i:
            count+=1
    return count

def replaceWord(filepath,word):
    filedata = readFile1(filepath)
    with open(filepath,'w') as f:
        filedata = [w.replace(word, 'siri') for w in filedata]
        s=''
        for i in filedata:
            s+=''.join(i)
            s+='\n'
        f.write(s)
    return filedata
import re
def readFile1(filepath):
    with open(filepath,'r') as f:
        filedata=f.read()
        pattern='[\n]'
        filedata = re.split(pattern,filedata)
    return filedata

filepath = 'DataFiles/replace.txt'
word=input()
replaceWord(filepath,word)
```

In [7]:
```
# Function to check if two strings are anagrams
# abc cbc --> True
# {'ea':1,'b':1,'c':1} {'c':1, 'a':1, 'b':1}   --> both dictionaries are sam
# aabbcc ccbbaaa --> False
# By Sorting the strings we will get as --> aabbcc  aaabbcc
def checkAnagrams(s1,s2):
    if len(s1) != len(s2):
        return False
    if sorted(s1) == sorted(s2):
        return True
    return False
checkAnagrams('abc','bcc')

```

Out[7]: False

In [9]:

```
t=int(input())
for i in range(0,t):
    s1=sorted(input())
    s2=sorted(input())

    count=0
    for i in s1:
        if i not in s2:
            count+=1
    for i in s2:
        if i not in s1:
            count+=1
    print(count)
```

```
2
cde
abc
4
dfdskc
acjdd
5
```

In [15]:
```
def CharDeletionsAnagrams(s1,s2):
    # characters occuring  only  string
    uncommon = [] # to collect all uncommon characters i.e., characters occu
    for i in s1:
        if i not in s2:
            uncommon.append(i)
    for i in s2:
        if i not in s1:
            uncommon.append(i)
    count = len(uncommon)
    # freqs1 -> Freqency of common charactes which are only in s1
    # freqs2 -> Freqency of common charactes which are only in s2
    freqs1 = {}
    freqs2 = {}
    # uniqs1 ,uniqs2 --> Unique characters in s1 and s2
    uniqs1 = []
    uniqs2 = []
    # Frequency of common unique characters in s1
    for i in s1:
        if i not in uncommon and i not in uniqs1:
            freqs1[i] = s1.count(i)
            uniqs1.append(i)
    # Frequency ofcommon unique characters in s2
    for i in s2:
        if i not in uncommon and i not in uniqs2:
            freqs2[i] = s2.count(i)
            uniqs2.append(i)
    # Diffence in frequencies for common characters
    for key in freqs1.keys():
        count +=abs(freqs1[key] - freqs2[key])
    return count
t=int(input())
for i in range(0,t):
    s1=input()
    s2=input()
    print(CharDeletionsAnagrams(s1,s2))
```

```
2
cde
abd
4
jskjdksaa
aajdjskjx
4
```

In [2]:
```
n= int(input())
for i in range(n):
    s1=input()
    s2=input()
    x1=[]
    x2=[]
    x=[]
    temp= ord('a')
    for i in range(0,26):
        x1.append(s1.count(chr(temp)))
        x2.append(s2.count(chr(temp)))
        temp = temp+1
    for j in range(0,26):
        x.append(abs(x2[j]-x1[j]))
    print(sum(x))

```

```
2
cde
abc
4
abc
cds
4
```

In [9]:
```
# Function to kLarest Frequency
# {a:4, g:9, i:6, p:213, c:6}
# [4,6,6,9,213]
# [213,9,6,6,4]
# [a,c,g,i,p]
# k = 3
# li = []
# for item in d.items():
#     if item[1] == 6
#         li.append(item[0])
# li = [i, c]

def KLargestFrequency(s, k):
    # Consturct the frequency dictionary
    unique = []
    freq = {}
    for i in s:
        if i not in freq.keys():
            freq[i] = s.count(i)
            values = sorted(freq.values(), reverse=True)
            uniqueValues = list(set(values))
            uniqueValues = sorted(uniqueValues, reverse = True)
    kvalue = uniqueValues[k-1]
    return kvalue

KLargestFrequency([2,3,4,3,2,4,2,4,9,6,5,4],2)
```

Out[9]: 3

In [ ]:
```

```