In [7]:
```
# s = 123abc456def
# 0 1 1 1 1 1 1 0 0 0   (String) - Frequency of sorted numbers
# count(1) --> 1 4 1 1 1 1 4 4 4

# 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:
            c=allnumbers.count(str(i))
            print(c, end = ' ')
digitFrequency1('212abc456def111')


```

```
0 4 2 0 1 1 1 0 0 0
```

In [4]:
```
#Second solution:

def digitFrequency2(s):
    for i in range(0,10):
        count = s.count(str(i))

digitFrequency2('212abc456def111')
```

In [ ]:
```

```

In [ ]:
```

```

Contacts Application :

- Addcontact(name,phone,email)
- searchcontact(name)
- listcontact()
- modify/editcontact(name, newphone, newemail)
- deletecontact(name)
- contactsapp()

In [10]:

```
from Packages.validators import phoneNumValidator as pnv, emailValidator as
#from Packages.validators import emailValidator as ev


def addContact(name, phone, email):
    # store data as name,phone,email in the contacts file
    filename = 'DataFiles/contacts.txt'
    if not checkContactExists(name):
        if pnv(phone) and ev(email):
            with open(filename, 'a') as f:
                line = name + ',' + str(phone) + ',' + email + '\n'
                f.write(line)
            print(name, 'added to contacts')
        else:
            print('Invalid Phone number or Email')
            return
    else:
        print(name, 'already exists')
    return

import re
# Function to check if contact already exists
def checkContactExists(name):
    filename = 'DataFiles/contacts.txt'
    with open(filename, 'r') as f:
        filedata = f.read()
        pattern = name+','
    return re.search(pattern, filedata)

#addContact('name5', 8765432109, 'name5_64@gmail.com')
#checkContactExists('name4')

# files to list

def contactsToList(name):
    ls = []
    with open(name,'r') as f:
        for line in f:
            ls.append(line.split(','))
    return ls
ls = contactsToList('DataFiles/contacts.txt')
ls
def saveToFile(s):
    with open("DataFiles/contacts.txt",'w') as f:
        f.write(s)
        print("\n \nData updated...")


def listToContacts(ls):
    s = ''
    for i in range(0,len(ls)):
        lls = ls[i]
        for j in range(0,len(lls)):
            s+=str(lls[j])+','
    return s
listToContacts(ls)
```

```
57
58 def editContact(name,number,email):
59     ls = contactsToList('DataFiles/contacts.txt')
60     for i in range(0,len(ls)):
61         lss = ls[i]
62         for j in range(0,len(lss)):
63             if(name == lss[j]):
64                 lss[0] = name
65                 lss[1] = number
66                 lss[2] = email
67                 print(name," Updated..")
68         else:
69             print("not updated")
70     s=listToContacts(ls)
71     saveToFile(s)
72     return s
73
74 editContact('name2',9876543219,'name3_n@gmail.com')
75
```

```
not updated
name2  Updated..
not updated
not updated
not updated
not updated
not updated


Data updated...
```

Out[10]:
```
'name1,9876543210, name1_123@gmail.com,\n,name2,9876543219,name3_n@gmail.com,na
me3_n@gmail.com,name3_n@gmail.com,name3_n@gmail.com,name3_n@gmail.com,name3_n@g
mail.com,name3_n@gmail.com,9876543219,name3_n@gmail.com,\n,,,,,,,,,name3,823456
7890, name3_123@gmail.com,\n,,,,,,,,,,,,,,name4,8765432109,name4_34@gmail.com
\n,,name5,8765432109,name5_64@gmail.com\n,,,'
```

In [13]:

```
# Function to check if two strings are anagrams

# abc cba --> True

# {a:1,b:1,c:!} {c:1,a:1,b:1}
# aabbcc ccbbaaa ---> False
# {a:2,b:2,c:2} {a:3,b:2,c:2}

# abccc --> {a:1,b:2,c:3}
# aabcb --> {a:2,b:2,c:1}

# uncommon [e,e,d,d,d]
def checkAnagrams(s1,s2):
    if len(s1) != len(s2):
        return False
    if sorted(s1) == sorted(s2):
        return True
    return False

checkAnagrams('abc','bca')



def charDeletionsAnagrams(s1,s2):
    # to select all uncommon characters - characters occuring
    uncommon = []
    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  -> Frequency of common charcters in s1
    # freqs2  -> Frequency of common charcters in s2

    freqs1 = {}
    freqs2 = {}

    # unique characters in s1 and s2
    uniqs1 = []
    uniqs2 = []

    # Frequency of 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 of 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)
    # Difference in frequencies for common characters and add the difference
    for key in freqs1.keys():
```

```
57          count += abs(freqs1[key] - freqs2[key])
58      return count
59
60  charDeletionsAnagrams('aabbcc','ccbbaaa')
61
```

Out[13]: 1

In [15]:

```
1   #Function to give average range
2
3   def averageRange(lb,ub):
4       sum = 0
5       for i in range(lb,ub+1):
6           sum += i
7       count = ub-lb+1
8       return sum//count
9
10  averageRange(1000,123456)
11
```

Out[15]: 62228

In [22]:

```
# Frequency of a word exam problem 2

#{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):
    # Construct the frequency dictionary for all unique characters
    #unique = []
    freq = {}
    for i in s:
        if i not in freq.keys():
            freq[i] = s.count(i)

    #Extract  unique frequencies in descending order
    values = sorted(freq.values(),reverse = True)
    uniquevalues = list(set(values))
    uniquevalues = sorted(uniquevalues, reverse = True)

    #identify kth largest frequency
    if k < len(freq.keys()):
        kvalue = uniquevalues[k-1]
    else:
        return -1
    #get all elements with kth largest frequency
    li = []
    for item in freq.items():
        if item[1] == kvalue:
            li.append(item[0])
    #Minimum of kth largest frequency
    return min(li)

kLargestFrequency('aabcdcc',3)

```

Out[22]: 'b'

In [ ]:

```

```