In [7]:
```
# Digit Frequency
# s=123abc456def
# 0 1 1 1 1 1 1 1 0 0 0

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('213abc456def111')



```

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

In [2]:
```
def digitcount(s):
    li=[]
    for i in range(10):
        count=0
        for j in range(len(s)):
            if s[j]==str(i):
                count=count+1
        li.append(count)
    for i in li:
        print(i,end=' ')
    return
s=input()
digitcount(s)

```

```
213abc456def111
0 4 1 1 1 1 1 0 0 0
```

In [3]:
```
from random import randint
def generatemarks(n,lb,ub):
    with open('DataFiles/marks.txt','w') as f:
        for i in range(0,n):
            r=randint(lb,ub)
            f.write(str(r)+'\n')
    return
generatemarks(10,0,100)
```

In [11]:
```
## Generation of marks
from random import randint
def generatemarks(n,lb,ub):
    filepath='Data/marks.txt'
    with open (filepath,'w') as f:

        for i in range(0,n):

            r=randint(lb,ub)
            f.write(str(r)+'\n')
            #print(i+1,'--->',r)
    print(n,"Marks stored in file successfully")

generatemarks(30,1,100)
```

```
30 Marks stored in file successfully
```

In [9]:
```
# Class average
#
def classavg(filepath):
    s=0
    c=0
    with open(filepath,'r') as f:
        for i in f:
            s=s+int(i)
            c=c+1
    print(s/c)
classavg('Data/marks.txt')
```

```
55.7
```

In [12]:
```
# Passed count
# passed count/total student count) * 100

def passedcount(filepath):
    c=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if(int(i)>=35):
                c=c+1
        print((c/mc)*100)
passedcount('Data/marks.txt')

```

```
66.66666666666666
```

In [13]:
```
# Fail Count

def failedcount(filepath):
    c=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if(int(i)<35):
                c=c+1
        print((c/mc)*100)
failedcount('Data/marks.txt')


```

```
33.33333333333333
```

In [14]:
```
# Distinction

def distinction(filepath):
    c=0
    mc=0
    with open(filepath,'r') as f:
        for i in f:
            mc=mc+1
            if(int(i)>=75):
                c=c+1
        return (c/mc)*100
distinction('Data/marks.txt')

```

Out[14]:
```
30.0
```

In [15]:
```
# frequency of highest marks

def frequencyhigh(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        print(max(sp))
        print(sp.count(max(sp)))

frequencyhigh('Data/marks.txt')

```

```
97
1
```

In [16]:
```
def frequencyhigh(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        print(max(sp))
        print(sp.count(max(sp)))

frequencyhigh('Data/marks.txt')

```

```
97
1
```

In [17]:
```
def frequencylow(filepath):
    with open(filepath,'r') as f:
        sp=f.read().split()
        sp=list(map(int,sp))
        print(min(sp))
        print(sp.count(min(sp)))

frequencylow('Data/marks.txt')
```

```
6
1
```

In [18]:
```
def marksanalysis(filepath):
        while True:

            n=int(input("Choose option :\n1).generatemarks\n2).classavg\n3).
            if(n==1):
                st=int(input("Enter No of Students marks"))
                generatemarks(st,1,100)
            elif(n==2):
                print(classavg(filepath))
            elif(n==3):
                print(passedcount(filepath))
            elif(n==4):
                print(failedcount(filepath))
            elif(n==5):
                print(distinction(filepath))
            elif(n==6):
                print(frequencyhigh(filepath))
            elif(n==7):
                print(frequencylow(filepath))
            else:
                break
marksanalysis('Data/marks.txt')

```

```
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
1
Enter No of Students marks70
70 Marks stored in file successfully
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
2
46.714285714285715
None
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
3
62.857142857142854
```

```
None
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
4
37.142857142857146
None
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
5
22.857142857142858
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
6
100
1
None
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
7
1
3
None
Choose option :
1).generatemarks
2).classavg
3).passedcount
4).failedcount
5).distinction
6).frequencyhigh
7).frequencylow
8
```

In [37]:
```
## Check anagram

def checkanagram(s1,s2):
    if(len(s1)!=len(s2)):
        return False
    if sorted(s1)==sorted(s2):
        return True
    return False
checkanagram('abc','bcc')

```

Out[37]: False

In [12]:
```
def chardeletionsanagram(s1,s2):
    # to collect all uncommon characters-characters
    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)
    c=len(uncommon)
    # freqs1 -> frequency of common characters in s1(repeating characters)
    # freqs2 -> frequency of common characters in s2
    freqs1={}
    freqs2={}
    # unique characters in s1 and s2
    uniqs1=[]
    uniqs2=[]
     # frequency of unique characters of s1
    for i in s1:
        if i not in uncommon and i not in uniqs1:
            freqs1[i]=s1.count(i)
            uniqs1.append(i)
    # frequency od 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
    for key in freqs1.keys():
        c=c+abs(freqs1[key]-freqs2[key])
    return c

chardeletionsanagram('cde','abc')



```

Out[12]: 4

In [ ]:
```
def anagram(a,b):
    c=0
    for i in set(a):
        if a.count(i)==b.count(i):
            c += a.count(i)
        else:
            c += min([a.count(i),b.count(i)])
    print(len(a)  + len(b) - 2  * c)
t = int(input())
for i in range(t):
    a=input()
    b=input()
    anagram(a,b)

```

```
2
cde
abc
4
```

In [14]:
```
def largestfrequency(s,k):
    # construct the frequency dictionary
    #unique=[]
    fre={}
    for i in s:
        if i not in fre.keys():
            #unique.append(i)
            fre[i]=s.count(i)
            values=sorted(fre.values(),reverse=True)
            uniquevalues=list(set(values))
            uniquevalues=sorted(uniquevalues,reverse=True)
            if k <= len(uniquevalues):

                kvalue=uniquevalues[k-1]
            else:
                return -1
            li=[]
            for items in fre.items():
                if items[1]==kvalue:
                    li.append(items[0])
            return min(li)

largestfrequency('aabcd',2)




```

Out[14]: -1

In [ ]:
```

```