Given two matrices find the multiplication of the matrices.

In [1]:
```
f = int(input("enter number of rows in A:")) #get no. of columns in matrix A
g = int(input("enter number of columns in A:")) #get no. of columns in matrix A
h = int(input("enter number of rows in B:")) #get no. of rows in matrix B
o = int(input("enter number of columns in B:")) #get no. of columns in matrix B

if g!=h:
    print("Not Possible!!")

result=[[0 for i in range(o)] for j in range(f)]

print("enter elements in A:")
A=[[int(input()) for i in range(g)] for j in range(f)]
print("A:")
for i in range(f):
    for j in range(g):
        print(format(A[i][j],"<3"),end="")
    print()

print("enter elements in B:")
B=[[int(input()) for i in range(o)] for j in range(h)]
print("B:")
for i in range(h):
    for j in range(o):
        print(format(B[i][j],"<3"),end="")
    print()

for i in range(f):
    for j in range(o):
        for l in range(g):
            result[i][j] = result[i][j] + A[i][l] * B[l][j]

print("A*B:")
for i in range(f):
    for j in range(o):
        print(format(result[i][j],"<3"),end="")
    print()
```

```
enter number of rows in A:2
enter number of columns in A:2
enter number of rows in B:2
enter number of columns in B:5
enter elements in A:
1
2
3
4
A:
1  2
3  4
enter elements in B:
1
2
3
4
5
5
6
7
8
9
B:
1  2  3  4  5
5  6  7  8  9
A*B:
11 14 17 20 23
23 30 37 44 51
```

selecting a number randomly with probability proportional to its magnitude.

In [2]:
```
import random as rd
first=int(input("enter the starting number of the range:"))
last=int(input("enter the ending number of the range:"))
rg=int(input("enter the range:"))
Rn = [rd.randint(first,last) for i in range(rg)]
print(Rn)
Rn.sort(reverse=True)
for i in range(len(Rn)):
    print("f({})".format(Rn[i]),end="+")
```

```
enter the starting number of the range:100
enter the ending number of the range:200
enter the range:15
[142, 178, 145, 155, 137, 119, 106, 196, 132, 182, 151, 185, 162, 160, 166]
f(196)+f(185)+f(182)+f(178)+f(166)+f(162)+f(160)+f(155)+f(151)+f(145)+f(142)+f(137)+f(132)+f(119)+f(106)+
```

Replace the digits in the string with

In [80]:
```
import re
str=input("enter the string in which the digit is changed:")
str1 = re.sub(r'[0-9]',"#",str)
print(str1)
```

```
enter the string in which the digit is changed:a@2$56$ghT
a@#$##$ghT
```

Students marks dashboard

In [3]:
```
lst=['stud1','stud2','stud3','stud4','stud5','stud6','stud7','stud8','stud9','stud10']
mark=['34','69','72','99','23','13','54','63','90','12']

import math
dash=list(zip(lst,mark))
dash.sort(key=lambda x:x[1],reverse=True)
print("top 5:",dash[:5])
dash.sort(key=lambda x:x[1])
print("last 5:",dash[:5])

tfp_25 = math.ceil(len(dash) // 4)
sfp_75= math.floor(3 * len(dash) // 4)
students_within_25_and_75 = dash[tfp_25:sfp_75]

print("between 25 to 75:",students_within_25_and_75)
```

```
top 5: [('stud4', '99'), ('stud9', '90'), ('stud3', '72'), ('stud2', '69'), ('stud8', '63')]
last 5: [('stud10', '12'), ('stud6', '13'), ('stud5', '23'), ('stud1', '34'), ('stud7', '54')]
between 25 to 75: [('stud5', '23'), ('stud1', '34'), ('stud7', '54'), ('stud8', '63'), ('stud2', '69')]
```

Find the closest points:

In [5]:
```
import math
P = (3,-4)
S = [(1,2), (3,4),(-1,1),(6,-7),(0,6),(-5,-8),(-1,-1),(6,0),(1,-1)]
distance = {}
for x in range(len(S)):
    distance[S[x]] = math.acos((S[x][0]*P[0])+(S[x][1]*P[1]))/((math.sqrt((S[x][0]**2)+(S[x][1]**2)))*(math.sqrt((P[0]**2)+(P[1]**2))))
dist = sorted(distance.values())[:5]
print(dist)
print("points closer to P :")
for x in dist:
    print(format(list(distance.keys())[list(distance.values()).index(x)]))
```

```
[0.06512516333438509, 0.14189705460416438, 0.9272952180016123, 1.2021090421368467, 1.4288992721907328]
points closer to P :
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

Find Which line separates oranges and apples:

In [40]:
```
red=[(1,1),(2,1),(4,2),(2,4),(-1,4)]
blue=[(-2,-1),(-1,-2),(-3,-2),(-3,-1),(1,-3)]
lines=["1x+1y+0","1x-1y+0","1x+0y-3","0x+1y-0.5"]
import re
for i in lines:
    BL=[]
    RD=[]
    l=re.findall\('[0-9\+\-\.]+',i)
    for x in blue:
        linebelue=float(l[0])*(x[0])+float(l[1])*(x[1])+float(l[2])
        if linebelue > 0:  ##https://stackoverflow.com/questions/57188227/to-find-whether-a-given-line-equation-is-able-to-separate-the-two-lists-of-poi
            BL.append(0)
        else:
            BL.append(1)
    lr=len(set(BL))
    print(lr)
    for x in red:
        linered=float(l[0])*(x[0])+float(l[1])*(x[1])+float(l[2])
        if linered > 0:
            RD.append(0)
        else:
            RD.append(1)
    lb=len(set(RD))
    print(lb)
    if lr == 1 and lb == 1:
        print("YES")
    else:
        print("NO")
```

```
1
1
YES
2
2
NO
1
2
NO
1
1
YES
```

Filling the missing values in the specified formate

In [60]:
```
input_list = ['-','-','30','-','-','-','50','-','-']
cnt = 0
output_list = input_list
c = 0
prev = 0
prev_val = 0
prev_start_range = 0
prev_end_range = 0
#output_list = []

for i in input_list:
    cnt = cnt+1
    if i == '-' and cnt != len(input_list):
        c = c+1
    else:
        c = c+1
        if i == '-': val = (prev_val)/c
        else: val = (int(i)-prev_val)/c
        if prev_end_range == 0:
            start_range = 0
            end_range = c
        else:
            start_range = prev_end_range - 1
            end_range = c+start_range
        for j in range(start_range,end_range):
            output_list[j] = int(val)
        prev = c
        prev_start_range = start_range
        prev_end_range = end_range
        prev_val = int(val)
        c = 1
print(output_list)
```

```
[10, 10, 12, 12, 12, 12, 4, 4, 4]
```

Filling the conditional probability

In [3]:
```
A = [['F1','S1'],['F2','S2'],['F3','S3'],['F1','S2'],['F2','S3'],['F3','S2'],['F2','S1'],['F4','S1'],['F4','S3'],['F5','S1']]
l=len(A)
F=['F1','F2','F3','F4','F5']
S=['S1','S2','S3']
for f in F :
    for s in S:
        n=0
        d=0
        for i in range(0,l):
            if(A[i]==s):  # https://stackoverflow.com/questions/57160252/find-conditional-probabilities-using-python
                #print(A[i][1])
                d=d+1
                #print(d)
                if(A[i][0]==f):
                    #print(A[i][0])
                    n=n+1
                    #print(n)
        print('P(F=()|S==()=()/()'.format(f, s, n, d))
```

```
P(F=F1|S==S1)=1/4
P(F=F1|S==S2)=1/3
P(F=F1|S==S3)=0/3
P(F=F2|S==S1)=1/4
P(F=F2|S==S2)=1/3
P(F=F2|S==S3)=1/3
P(F=F3|S==S1)=0/4
P(F=F3|S==S2)=1/3
P(F=F3|S==S3)=1/3
P(F=F4|S==S1)=1/4
P(F=F4|S==S2)=0/3
P(F=F4|S==S3)=1/3
P(F=F5|S==S1)=1/4
P(F=F5|S==S2)=0/3
P(F=F5|S==S3)=0/3
```

Given two sentences S1, S2

In [1]:
```
S1= "the first column F will contain only 5 uniques values"
S2= "the second column S will contain only 3 uniques values"
A1=S1.split()
A2=S2.split()
#O2 = []
#O3 = []
O1=len(set(A1)&set(A2))
print(O1)
O2= list(set(A1)-set(A2))
print(O2)
O3= list(set(A2)-set(A1))
print(O3)
```

```
7
['first', 'F', '5']
['3', 'second', 'S']
```

Given two sentences S1, S2

In [2]:
```
from math import log
lst = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
print(lst)
fx=0
for x in lst:
    f = f + (x[0]*log((x[1]),10)+((1-x[0])*log(1-(x[1]),10)))
    fx = -1 * (f/len(lst))
print(fx)
```

```
[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
0.42430993457031635
```