In [1]:

```
import random

def Print_values(a,b,c):
    #输出a,b,c，实际上是从大到小按顺序输出
    if a>b:
        if b>c:
            print(a,b,c)
        else:
            if a>c:
                print(a,c,b)
            else:
                print(c,a,b)
    else:
        if b>c:
            if a>c:
                print(c,a,b)
        else:
            print(c,b,a)

a=random.random()#获取随机数
b=random.random()
c=random.random()
Print_values(a,b,c)
```

```
0.7695844821297868 0.7426451121643899 0.49804348032252466
```

In [2]:

```
import random
#2.1
n = 10
m = 5
M1 = [[random.randint(0,50) for i in range(n)] for j in range(m)]#五行十列
M2 = [[random.randint(0,50) for i in range(m)] for j in range(n)]#五列十行
#2.2
def Matrix_multip(M1,M2):#两个矩阵相乘
    R=[[0 for i in range(5)] for j in range(5)]  # 五行五列
    for i in range(5):
        for j in range(5):
            sum=0
            for k in range(10):
                 sum += M1[i][k]*M2[k][j]#行*列
            R[i][j] = sum
        print(R[i])


Matrix_multip(M1,M2)
```

```
[4677, 6016, 5213, 4394, 6365]
[3792, 6514, 6522, 3838, 6362]
[5241, 6254, 7172, 3381, 6432]
[3857, 6010, 5785, 3717, 5949]
[5100, 6801, 7168, 4086, 6480]
```

In [3]:

```
def Pascal_triangle(k):#帕斯卡三角函数
    n=0;
    p = [1]
    while n<k:
        n+=1
        p = [1] + [p[i] + p[i+1] for i in range(len(p)-1)] + [1]#调整每行数据
    print(p)#输出第K列的数值

Pascal_triangle(50)
Pascal_triangle(200)
```

```
[1, 50, 1225, 19600, 230300, 2118760, 15890700, 99884400, 536878650, 2505433700, 1
0272278170, 37353738800, 121399651100, 354860518600, 937845656300, 2250829575120,
4923689695575, 9847379391150, 18053528883775, 30405943383200, 47129212243960, 6732
7446062800, 88749815264600, 108043253365600, 121548660036300, 126410606437752, 121
548660036300, 108043253365600, 88749815264600, 67327446062800, 47129212243960, 304
05943383200, 18053528883775, 9847379391150, 4923689695575, 2250829575120, 93784565
6300, 354860518600, 121399651100, 37353738800, 10272278170, 2505433700, 536878650,
99884400, 15890700, 2118760, 230300, 19600, 1225, 50, 1]
[1, 200, 19900, 1313400, 64684950, 2535650040, 82408626300, 2283896214600, 5509899
6177225, 1175445251780800, 22451004309013280, 387790074428411200, 6107693672247476
400, 88326646952501966400, 1179791641436990551200, 14629416353818682834880, 169152
626591028520278300, 183082842898524986654160 0, 186134223613500403098396 00, 1782969
9314556354402056880 0, 161358778796735007338614764 0, 13830752468291572057595551200,
11253203144655415446861834840 0, 87090006945594084762669852240 0, 64228880122375637 5
1246901602700, 45217131606152448808778187283008, 30434607811833379005908395286640
0, 19613413923181510914918743629168 00, 12118287888251433529574795170878800, 718739
83337215398865064302392798400, 40968170502212777353086652363895088 0, 2246641608185
86198387894545221360160 0, 11865075993231583602360680669503083450, 6040402323826988
015747255613565206120 0, 29669034943503147018523284925452630060 0, 14071599430347206
87164247227892896168560, 64494830722424698161694664611757741059 00, 285868979418314
87833832229719806133874800, 12262274643469822412933298537706315319980 0, 5093560236
51823392537229323873954636368400, 20501579951985891549623480285926674113828 10, 800
06165666286406047311142579226045322256 00, 30288048430808425146482075404992717157711
200, 11129096865273793425916669567415928630042720 0, 39710641087454217451566298229
1886544299251600, 13766355576984128716542983386118733535707388 80, 4638663292244652
06753078788010522543051009840 0, 15199024404376094008505134756089462048905428800, 4
84468902889487996521101170350351602808860543 00, 15028423110041260300246403651684376
2503973066400, 45385837792324606106744139028086816276199866052 8, 1334877582127194
29725718055964961224341764311920 0, 38249376872490759671407673728421581590236312454
00, 10680958070054023455411954173219611462933913666400, 29075941412924841628621430
80487560898243120942520 0, 77183408114309579595976889045669798389726483201440, 1998
49896010265875739583016278966442259113215432300, 50488394781540852818420972533633 6
275180917596881600, 12448000782345417160403791503982084026012278681736 00, 29959595
10327202096232776938246535477447023004756800, 70405048492689249261470258048793583 7
2000504061178480, 16158535719633598191157108404641150361968369976475200, 362263945
97243066912432872068469675811509732689194400, 79353054832056241808186291197600242 2
53783223985854400, 16986513299987039262064877959486301857450471384471957 5, 3554101
24430498052252434369613867238863579093582797880, 72697525451692783415270666511927 3
89767550269141935930 0, 14539505090338556683054133302385477953510053828387186 00, 28
43756142669158880656176072378336129142407587022787850, 54402291424975213369074672 6
88976865079246058186522898 00, 10181000252388218501926831603222813321973190889192 14
2340, 18641268067753076130288564907309376505021335430915190200, 33398938621390928 0
66767012125595966238163225980389715775, 58562522514219709486933939069538132581984 8
34595751830400, 10050595080143112303838662515988301132313613504946597920 0, 1688499
97346404286704489530268603459022868706883102845056, 27771381142500705050080514846 8
09779444550774158405073200 0, 44722743658053083457272517415641722741873973969379598
4000, 70524326537699093144160508232358101246801266644021674400 0, 10891098528606695
```

396946306334617327027987031051355245920000, 164727865245176267878812883311087071298303844651748094540 0, 24404128184470558204268575305346232784933902911370088080000, 3541574699941459056473122513824636221228212739576878636000, 503500981437460444173287297146153101331239883457917685600 0, 729714615310133123988345791768560000, 701304938430748475812793021024998962568512694816385347800 0, 957074974799609684638635181634116231269970265867067062880 0, 127980955932505946201677960334794612320984396017107804920 00, 167699183635697446747026292852489492006807139608624020240 00, 124780955932505946201677960334794612320984396017107804920 00, 957074974799609684638635181634116231269970265867067062880 0, 701304938430748475812793021024998962568512694816385347800 0, 503500981437460444173287297146153101331239883457917685600 0, 354157469994145905647312251382463622122821273957687863600 0, 244041281844705582042685753053462327849339029113700880800 0, 164727865245176267878812883311087071298303844651748094540 0, 108910985286066953969463063346173270279870310513552459200 0, 705243265376990931441605082323581012468012666440216744000, 44722743658053083457272517415641722741873973969379598400 00, 27771381142500705050080514846809779444550774158405073200 0, 168849997346404286704489530268603459022868706883102845056, 100505950801431123038386625159883011323136135049465979200, 58562522514219709486933939069538132581984834595751830400, 3339893862139092806676701212559596623816322598038971577 5, 18641268067753076130288564907309376505021335430915190200, 10181000252388218501926831603222813321973190889192142340, 54402291424975213369074672688976865079246058186522898 00, 28437561426691588806561760723783361291424075870227878 50, 14539505090338556683054133302385477953510053828387186 00, 72697525451692783415270666511927389767550269141935930 0, 3554101244304980522524343696138672388635790935827978 80, 16986513299987039262064877959486301857450471384471957 5, 79353054832056241808186291197600242253783223985854400, 3622639459724306691243287206846967581150973268919440 0, 16158535719633598191157108404641150361968369976475200, 70405048492689249261470258048793583720005040611784 80, 2995959510327202096232776938246535477447023004756800, 12448000782345417160403791503982084026012278681736 00, 50488394781540852818420972533633627518091759688160 0, 1998498960102658757395830162789664422591132154323 00, 77183408114309579595976889045669798389726483201440, 29075941412924841628621430804875608982431209425200, 10680958070054023455411954173219611462933913666400, 38249376872490759671407673728421581590236312454 00, 1334877582127194297257180559649612243417643119200, 45385837792324606106744139028086816276199866052 8, 15028423110041260300246403651684376250397306640 0, 48446890288948799652110117035035160280886054300, 15199024404376094008505134756089462048905428800, 46386632922446520675307878801052254305100984 00, 13766355576984128716542983386118733535707388 80, 39710641087454217451566298229188654429925160 0, 111290968652737934259166695674159286300427200, 30288048430808425146482075404992717157711200, 8000616566628640604731114257922604532225600, 20501579951985891549623480285926674113828 10, 5093560236518233925372293238739546363684 00, 12262274643469822412933298537706315319980 00, 28586897941831487833832229719806133874800, 64494830722424698161694664611757741 05900, 14071599430347206871642472278928961685 60, 29669034943503147018523284925452 6300600, 60404023238269880157472556135652061200, 11865075993231583602360680669503 08

```
3450, 22466416081858619838789454522136016000, 40968170502212777353086652363895088
0, 71873983337215398865064302392798400, 12118287888251433529574795170878800, 19613413
92318151091491874362916800, 30434607811833379005908395286640
0, 45217131606152448808778187283008, 64228880122375637512469016027
00, 87090006945594084762669852240
0, 11253203144655415446861834840
0, 13830752468291572057595551200, 16135877879673500733386147640, 17829699314556354402056880
0, 18613422361350040309839600, 18308284289852498665416
00, 16915262659102852027830
0, 14629416353818682834880, 11797916414369905512
00, 88326646952501966400, 6107693672247476400, 387790074428411200, 22451004309013280, 1175445251780800, 55098996177225, 2283896214600, 82408626300, 2535650040, 64684950, 1313400, 19900, 200, 1]
```

In [4]:

```
import random
def Least_moves(n):
    sum=n#令总值为金币的计算和
    count=0
    while sum>1:#循环直到一个硬币为止
        count+=1
        if sum%2== 1:#判断当前金钱是否是奇数，是奇数则减去一
            sum=sum-1
        else:#偶数就除以2
            sum=sum/2
    return count

n=random.randint(1,100);
print('n:{0}'.format(n))
print(Least_moves(n))
```

```
n:6
3
```

In [5]:

```
#5.1
import random
from functools import reduce
#暂存操作数
operator = {1: '+', 2: '-', 0: ''}
#暂存1-9数字
base = ['1', '2', '3', '4', '5', '6', '7', '8', '9']

def Find_expression(n):
    total = 3 ** 8
    for i in range(total):#循环遍历获取所有运算式
        num=i
        # 转化为8位3进制数，得到运算符数组
        arr = []
        for index in range(8):
            index = 7 - index
            arr.append(num // (3 ** index))
            num -= (num // (3 ** index)) * (3 ** index)
        arr = map(lambda x: operator[x], arr)

        # 生成表达式
        expression = reduce(lambda x, y: x + y, zip(base, arr))
        expression = list(expression)
        expression.append('9')
        expression = ''.join(expression)
        # 计算表达式结果
        res = eval(expression)
        if res == n:#若是符合条件就输出结果
            print('{0} = {1}'.format(expression,n))

n=random.randint(1,100)
Find_expression(n)
```

```
123+4-56+7-8+9 = 79
123-4+56-7-89 = 79
12+3-4+5-6+78-9 = 79
12-3+4+56-7+8+9 = 79
1+2-3-4-5+6-7+89 = 79
1-23+4-5+6+7+89 = 79
1-2+34+56+7-8-9 = 79
1-2+3+4+5+67-8+9 = 79
1-2+3-4+5-6-7+89 = 79
1-2-3+4-5+67+8+9 = 79
```

In [ ]:

```

```