In [3]:

```
from math import perm


def Print_values(a,b,c):
    if a > b:
        if b>c:
            print(a,b,c)
        elif a>c:
            print(a,c,b)
        else:
            print(c,a,b)
    elif b>c:
        if a>c:
            print(b,a,c)
        else:
            print(b,c,a)
    elif c>b:
            print(c,b,a)




a,b,c=eval(input("请输入3个整数,中间用逗号分开:"))
Print_values(a,b,c)
```

```
请输入3个整数,中间用逗号分开:2,3,5
5 3 2
```

In [4]:

```
import numpy as np

M1 = np.random.randint(50,size=[5,10])
M2 = np.random.randint(50,size=[10,5])
print(M1,'\n',M2)


M3 = np.zeros([5,5])


def Matrix_multip(M1,M2):
    for a in range(0,5):
        for k in range(0,5):
             for b in range(0,10):
                M3[a][k] +=M1[a][b] * M2[b][a]
    print(M3)


Matrix_multip(M1,M2)
```

```
[[ 0 13 10 41 31 14 47 25 14 10]
 [ 6  2 14  5 20 20 21 11  7 44]
 [ 1 46 22  7 24  7 18  9  7 17]
 [26 17 46 13 25 16 17  7 23 15]
 [ 3 46  5 28 35  6  1 41 21  6]]
 [[33 20 38 22 47]
 [ 9 17 41 41 16]
 [25 30 40 23 35]
 [ 3 15 34 28 33]
 [31 19 31 15 44]
 [25 21 17 29 24]
 [ 7 49 35  0 34]
 [ 6 45 35  1 20]
 [32 19 10  8 15]
 [27  9 13  7 31]]
[[2998. 2998. 2998. 2998. 2998.]
 [3502. 3502. 3502. 3502. 3502.]
 [5141. 5141. 5141. 5141. 5141.]
 [3826. 3826. 3826. 3826. 3826.]
 [5015. 5015. 5015. 5015. 5015.]]
```

In [ ]:

```
def Pascal_triangle(k):
    def triangle():
        L = [1]
        while True:
            yield L
            L = [1] + [L[i-1] + L[i] for i in range(1,len(L))] + [1]

    t= triangle()

    n=0
    for t in triangle():

        n = n + 1
        if n == k:
            print(t)

Pascal_triangle(100)

Pascal_triangle(200)
```

```
[1, 99, 4851, 156849, 3764376, 71523144, 1120529256, 14887031544, 171200862756, 1731
030945644, 15579278510796, 126050526132804, 924370524973896, 6186171974825304, 38000
770702498296, 215337700647490344, 1130522928399324306, 5519611944537877494, 25144898
858450330806, 107196674080761936594, 428786696323047746376, 161305471473908437922 4,
571901217043857188997 6, 191462581358160885012 24, 606298174300842802538 76, 1818894522
90252840761628, 517685364210719623706172, 139966783656972342705 7428, 359914586546500
3098147672, 881170194648328344718 9128, 205606378751276613767746 32, 457640004317357624
19272568, 972485009174384951409 54207, 197443926105102399225573693, 383273503615787010
261407757, 711793649572175876199757263, 126541093257275711324401 2912, 215461861492
1181030658724688, 351543037171350589212 7392912, 549849365832112460050 6947888, 824774
048748168690076042 1832, 118686997258882811149874753368, 163901091452742930164937070 3
2, 217264237507124349288404953 68, 276518120463612808185242668 32, 337966591677748987
8196326128, 396743390230400985657087306 72, 447391482600239409357992069 28, 4846741061
502593601378247417 2, 504456722727820966674062486 28, 504456722727820966674062486 28, 4
846741061502593601378247417 2, 447391482600239409357992069 28, 396743390230400985657087
30672, 337966591677748987781963261 28, 276518120463612808185242668 32, 21726423750712
434928840495368, 163901091452742930164937070 32, 118686997258882811149874753368, 82477
404874816869007604218 32, 549849365832112460050 6947888, 351543037171350589212 7392912,
215461861492118103065872 4688, 126541093257275711324401 2912, 711793649572175876199757
263, 383273503615787010261407757, 197443926105102399225573693, 972485009174384951409
54207, 457640004317357624192725 68, 205606378751276613767746 32, 881170194648328344718
9128, 359914586546500309814767 2, 139966783656972342705 7428, 517685364210719623706172
, 181889452290252840761628, 606298174300842802538 76, 191462581358160885012 24, 57190
121704385718899 76, 161305471473908437922 4, 428786696323047746376, 107196674080761936
594, 251448988584503308 06, 551961194453787749 4, 113052292839932430 6, 215337700647490
344, 380007707024982 96, 6186171974825304, 924370524973896, 126050526132804, 15579278
510796, 1731030945644, 171200862756, 14887031544, 1120529256, 71523144, 3764376, 156
849, 4851, 99, 1]
```

In [ ]:

```
def Least_moves(n):
    if n==1:
        return 0

    elif n%2!=0:
        return 1 + Least_moves(n-1)

    else:
        return 1 + min(Least_moves(n-1), Least_moves(int(n/2)))

Least_moves(90)
```

9

In [ ]:

```
from functools import reduce

operators = {
    1: '+',
    2: '-',
    0: ''
}

bases = ['1', '2', '3', '4', '5', '6', '7', '8', '9']


def find_expression(num):
    arr = []
    for i in range(8):
        i = 7-i
        arr.append(num//(3**i))
        num -= (num//(3**i))*(3**i)

    arr = map(lambda x: operators[x], arr)

    formula = reduce(lambda x, y: x+y, zip(bases, arr))
    formula = list(formula)
    formula.append('9')
    formula = ''.join(formula)

    result = eval(formula)
    return result, formula


if __name__ == '__main__':
    total = 3**8
    n = 0
    Total_solutions = []
    for j in range(total):
        result, formula = find_expression(j)
        if result == 50:
            print(formula+" = 50")

    for k in range(1, 100):
        for l in range(total):
            result, formula = find_expression(l)
            if result == k:
                n += 1
        Total_solutions.append(n)
        n = 0

print(Total_solutions)
print(Total_solutions.index(max(Total_solutions))+1)
print(Total_solutions.index(min(Total_solutions))+1)
```

12+3+4-56+78+9 = 50

12-3+45+6+7-8-9 = 50

12-3-4-5+67-8-9 = 50

1+2+34-56+78-9 = 50

1+2+34-5-6+7+8+9 = 50

1+2+3+4-56+7+89 = 50

1+2+3-4+56-7+8-9 = 50

1+2-34+5-6-7+89 = 50

1+2-3+4+56+7-8-9 = 50

1-23+4+5-6+78-9 = 50

1-23-4-5-6+78+9 = 50

1-2+34+5+6+7+8-9 = 50

1-2+34-5-67+89 = 50

1-2+3-45+6+78+9 = 50

1-2-34-5-6+7+89 = 50

1-2-3+4+56-7-8+9 = 50

1-2-3-4-5-6+78-9 = 50

[26, 11, 18, 8, 21, 12, 17, 8, 22, 12, 21, 11, 16, 15, 20, 8, 17, 11, 20, 15, 16, 11, 23, 18, 13, 14, 21, 15, 19, 17, 14, 19, 19, 7, 14, 19, 19, 17, 18, 16, 17, 18, 10, 15, 26, 18, 15, 16, 12, 17, 19, 9, 17, 21, 16, 13, 14, 16, 17, 17, 11, 13, 22, 14, 13, 15, 15, 15, 17, 7, 14, 17, 15, 12, 13, 14, 14, 14, 10, 9, 19, 12, 13, 13, 12, 11, 12, 6, 12, 14, 16, 13, 11, 11, 10, 11, 7, 9, 17]

1

88