```
#Question_1_Print_values
#输入a、b、c的值并将字符串转化为浮点数
a = float(input("请输入a的值："))
b = float(input("请输入b的值："))
c = float(input("请输入c的值："))
#流程判断
if a > b:
    if b > c:
        print(a + b - 10*c)
    else:
        if a > c:
            print(a + c - 10*b)
        else:
            print(c + a - 10*b)
else:
    if b > c:
        print("无结果")
    else:
        print(c + b - 10*a)
```

```
请输入a的值：  5
请输入b的值：  15
请输入c的值：  10
```

```
无结果
```

```
#Question_2
#定义ceil函数
def ceil(x):
    if x % 3 == 0:
        y = x / 3
    else:
        y = x // 3 + 1
    return y
#定义连续天花板函数
def F(x):
    if x == 1:
        return 1
    return F(ceil(x)) + 2 * x
#给定一个正整数列表，计算列表中每个元素连续天花板函数值
N = [1,2,3,4,5,6,7,8,9,10]
for i in N:
    print(F(i))
```

```
1
5
7
13
15
17.0
21
23
25.0
33
```

```
#Question_3.1
def Find_number_of_ways(x):
    if x < 10 or x > 60:
        return 0
    def dice(dice_remain,dice_sum):
        if dice_remain == 0:
            return 1 if dice_sum == x else 0
        if dice_sum > x:
            return 0
        if dice_sum + dice_remain > x:
            return 0
        if dice_sum + 6 * dice_remain < x:
            return 0
        count = 0
        for dice_value in range(1,7):
            count += dice(dice_remain - 1,dice_sum + dice_value)
        return count
    return dice(10,0)
#结果测试
test_values = [10,20,30,35,60]
for value in test_values:
    ways = Find_number_of_ways(value)
    print(f"10个骰子总和为{value}的方法数为：{ways}")
```

```
10个骰子总和为10的方法数为：1
10个骰子总和为20的方法数为：85228
10个骰子总和为30的方法数为：2930455
10个骰子总和为35的方法数为：4395456
10个骰子总和为60的方法数为：1
```

```
#Question_3.2
Number_of_ways = []
for i in range(10,61):
    ways = Find_number_of_ways(i)
    Number_of_ways += [ways]
Max_number_of_ways = max(Number_of_ways)
number = Number_of_ways.index(Max_number_of_ways) + 10
print(f"10个骰子总和为{number}时为最大方法数:{Max_number_of_ways}")
```

```
10个骰子总和为35时为最大方法数:4395456
```

```
#Question_4.1
import random
def random_integer(N):
    return [random.randint(0,10) for i in range(N)]
N = int(input("请输入数组的大小:"))
result = random_integer(N)
print("生成的随机数组:", result)
```

```
请输入数组的大小: 5
```

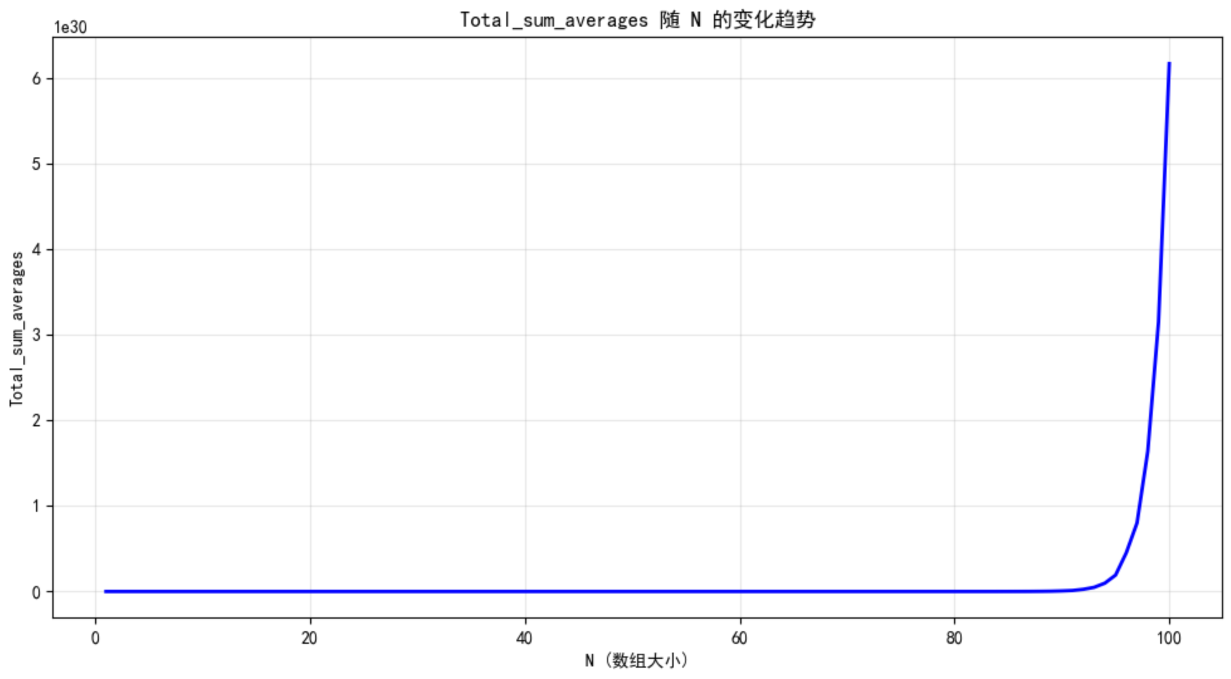
```
生成的随机数组: [4, 7, 8, 6, 8]
```

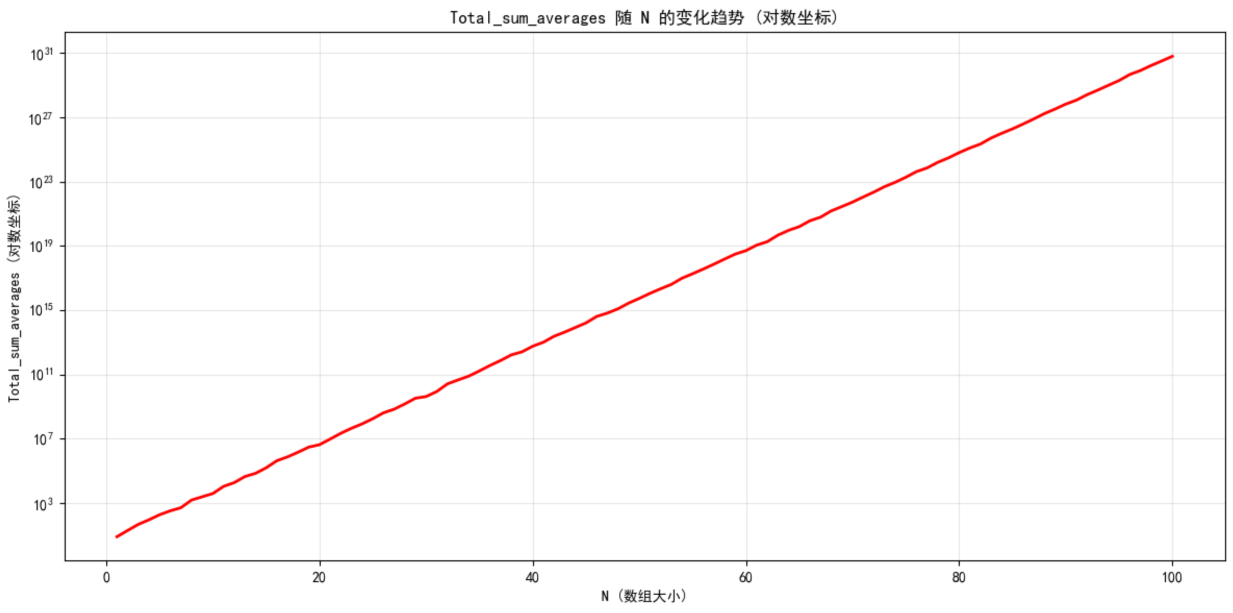
```
#Question_4.2
def sum_averages(arr):
    n = len(arr)
    total_sum = 0
    for i in range(1, n + 1):
        from math import comb
        combination_count = comb(n - 1, i - 1)
        total_sum += (sum(arr) * combination_count) / i
    return total_sum
N = int(input("请输入数组的大小: "))
arr = random_integer(N)
result = sum_averages(arr)
print(f"随机数组:{arr}")
print(f"所有子集平均值之和:{result}")
```

```
请输入数组的大小:  5
```

```
随机数组:[2, 1, 8, 5, 3]
所有子集平均值之和:117.8
```

```python
#Question_4.3
import matplotlib.pyplot as plt
import matplotlib
# 设置中文字体
matplotlib.rcParams['font.sans-serif'] = ['SimHei']  # 用黑体显示中文
matplotlib.rcParams['axes.unicode_minus'] = False  # 正常显示负号

Total_sum_averages = []
N_values = list(range(1, 101))

for N in N_values:
    arr = random_integer(N)
    sum_average = sum_averages(arr)
    Total_sum_averages.append(sum_average)

# 绘制普通坐标图（参考deepseek）
plt.figure(figsize=(12, 6))
plt.plot(N_values, Total_sum_averages, 'b-', linewidth=2)
plt.xlabel('N (数组大小)')
plt.ylabel('Total_sum_averages')
plt.title('Total_sum_averages 随 N 的变化趋势')
plt.grid(True, alpha=0.3)

# 添加对数坐标图以更好地观察趋势
plt.figure(figsize=(12, 6))
plt.semilogy(N_values, Total_sum_averages, 'r-', linewidth=2)
plt.xlabel('N (数组大小)')
plt.ylabel('Total_sum_averages (对数坐标)')
plt.title('Total_sum_averages 随 N 的变化趋势 (对数坐标)')
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# 打印一些关键点的值
print("N=1 时的 Total_sum_averages:", Total_sum_averages[0])
print("N=10 时的 Total_sum_averages:", Total_sum_averages[9])
print("N=50 时的 Total_sum_averages:", Total_sum_averages[49])
print("N=100 时的 Total_sum_averages:", Total_sum_averages[99])
"""
现象解释：
1. Total_sum_averages随N的增加呈指数级增长
2. 在对数坐标下，曲线呈近似线性趋势，这证实了指数增长的特性
"""
```





```
N=1 时的 Total_sum_averages: 8.0
N=10 时的 Total_sum_averages: 3887.4
N=50 时的 Total_sum_averages: 5359283556570886.0
N=100 时的 Total_sum_averages: 6.173458423111478e+30
```

```
'\n现象解释：\n1. Total_sum_averages随N的增加呈指数级增长\n2. 在对数坐标下，曲线呈近似线性趋
势，这证实了指数增长的特性\n'
```

```
#Question_5.1
import random
def arr(N, M):
    arr1 = [[random.randint(0, 1) for _ in range(M)] for _ in range(N)]
    arr1[0][0] = 1
    arr1[-1][-1] = 1
    return arr1
arr1 = (arr(5,5))
print(arr1)
```

```
[[1, 0, 1, 1, 1], [1, 0, 0, 1, 0], [0, 1, 1, 0, 1], [0, 0, 1, 1, 0], [0, 1, 0, 0,
1]]
```

```
#Question_5.2(参考deepseek)
def count_path(arr1):
    N, M = len(arr1), len(arr1[0])
    dp = [[0]*M for _ in range(N)]
    dp[0][0] = 1 if arr1[0][0] == 1 else 0   # 起点
    # 首行
    for j in range(1, M):
        dp[0][j] = dp[0][j-1] if arr1[0][j] == 1 else 0
    # 首列
    for i in range(1, N):
        dp[i][0] = dp[i-1][0] if arr1[i][0] == 1 else 0
    # 其余
    for i in range(1, N):
        for j in range(1, M):
            dp[i][j] = 0 if arr1[i][j] == 0 else dp[i-1][j] + dp[i][j-1]
    return dp[-1][-1]
test_arr = arr(6,6)
print("测试矩阵：",test_arr)
print("测试矩阵的总路径数：",count_path(test_arr))
```

```
测试矩阵： [[1, 0, 1, 1, 0, 1], [1, 1, 0, 1, 0, 0], [0, 1, 1, 1, 0, 1], [1, 1, 0, 1,
1, 0], [0, 1, 0, 1, 1, 0], [1, 1, 0, 1, 1, 1]]
测试矩阵的总路径数： 3
```

```python
#Question_5.3
def run_experiment(N=10, M=8, trials=1000):
    # 重复实验并取平均
    total = 0
    for _ in range(trials):
        g = arr(N, M)
        total += count_path(g)
    return total / trials
mean_paths = run_experiment(N=10, M=8, trials=1000)
print("1000次运行中总路径数的平均值：",mean_paths)
```

```
1000次运行中总路径数的平均值： 0.267
```