1. Flowchart

Write a function Print_values with arguments a, b, and c to reflect the following flowchart. Here the purple parallelogram operator on a list [x, y, z] is to compute and print x+y-10z. Try your output with some random a, b, and c values. Report your output when a = 10, b = 5, c = 1.

In [1]:
```
def Print_values(a,b,c):

    if(a>b and b>c):
        print('The result is',a,b,c,',The final result is',a+b-10*c)
    if(a<=b and b<=c):
        print('The result is',c,b,a,',The final result is',c+b-10*a)
    if(a<=b and b>c and a>c):
        print('The result is',b,a,c,',The final result is',b+a-10*c)
    if(a<=b and b>c and a<=c):
        print('The result is',b,c,a,',The final result is',b+c-10*a)
    if(a>b and b<=c and a>c):
        print('The result is',a,c,b,',The final result is',a+c-10*b)
    if(a>b and b<=c and a<=c):
        print('The result is',c,a,b,',The final result is',c+a-10*b)
```

随机生成三个数字来进行计算

In [2]:
```
import random
a = random.randint(0,10)
b = random.randint(0,10)
c = random.randint(0,10)
Print_values(a,b,c)
```

```
The result is 8 6 1 ,The final result is 4
```

In [3]:
```
Print_values(10,5,1)    #最终结果
```

```
The result is 10 5 1 ,The final result is 5
```

2. Continuous celing function

Given a list with N positive integers. For every element x of the list, find the value of continuous ceiling function defined as F(x) = F(ceil(x/3)) + 2x, where F(1) = 1.

In [4]:
```
import math
import random
import numpy as np
N = input('请输入所定义数组的长度')
list1 = np.random.randint(1,20,size = int(N))
m = list1.max()
list2 = np.zeros(shape = m)
list2[0] = 1
for i in range (1,m):
    list2[i] = list2[math.ceil((i+1)/3)-1] + 2*(i+1)


print('生成的随机数组为：',list1[:])
print('对应的函数值分别为:',end=' ')
for i in range(int(N)):
    print(list2[list1[i]-1],end=' , ')
```

```
请输入所定义数组的长度10
生成的随机数组为： [ 3 11 13  5 18  7 14 18 14  4]
对应的函数值分别为: 7.0 , 35.0 , 41.0 , 15.0 , 53.0 , 21.0 , 43.0 , 53.0 , 43.0 , 13.0 ,
```

3 Dice rolling

3.1 Given 10 dice each with 6 faces, numbered from 1 to 6. Write a function Find_number_of_ways to find the number of ways to get sum x, defined as the sum of values on each face when all the dice are thrown.

3.2 Count the number of ways for any x from 10 to 60, assign the number of ways to a list called Number_of_ways, so which x yields the maximum of Number_of_ways?

In [5]:
```
def Find_number_of_ways(value): #10-60
    import numpy as np
    n=10
    dice = [[0 for i in range(6*n)] for i in range(n)]#Python 的深拷贝和浅拷贝
    for i in range(6):
        dice[0][i] = 1
    # print (dice)
    for i in range(1,n):
        for j in range(i,6*(i+1)):
            dice[i][j] = dice[i-1][j-6] + dice[i-1][j-5] + dice[i-1][j-4] + dice[i-1][j-3] + dice[i-1][j-2] + dice[i-1][j-1]

    count = np.array(dice[n-1])
    return count[value -1]
```

In [6]:
```
import numpy as np
Number_of_ways = np.zeros(shape = 51)
for i in range(10,61):
    Number_of_ways[i-10] = Find_number_of_ways(i)

print('此时10个骰子的点数之和为：     ', np.argmax(Number_of_ways)+10 )
print('10个骰子出现最多的路径数目为：  ', int(Number_of_ways[np.argmax(Number_of_ways)]),'条')
```

```
此时10个骰子的点数之和为：     35
10个骰子出现最多的路径数目为：  4395456 条
```

来源： https://blog.csdn.net/weixin_43494312/article/details/107288704?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-107288704-blog-122987002.pc_relevant_3mothn_strategy_recovery&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-107288704-blog-122987002.pc_relevant_3mothn_strategy_recovery&utm_relevant_index=6 (https://blog.csdn.net/weixin_43494312/article/details/107288704?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-107288704-blog-122987002.pc_relevant_3mothn_strategy_recovery&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-107288704-blog-122987002.pc_relevant_3mothn_strategy_recovery&utm_relevant_index=6)

4. Dynamic programming

4.1 [5 points] Write a function Random_integer to fill an array of N elements by randomly selecting integers from 0 to 10.

4.2 [15 points] Wri ave =[]te a function Sum_averages to compute the sum of the average of all subsets of the array. For example, given an array of [1, 2, 3], you Sum_averages function should compute the sum of: average of [1], average of [2], average of [3], average of [1, 2], average of [1, 3], average of [2, 3], and average of [1, 2, 3].

4.3 [5 points] Call Sum_averages with N increasing from 1 to 100, assign the output to a list called Total_sum_averages. Plot Total_sum_averages, describe what do you see.

In [7]:
```
def Random_integer(N):
    import math
    import random
    import numpy as np
#     N = input('请输入所定义数组的长度')
    array = np.random.randint(1,10,size = int(N))
#     print('生成的数组为',array)
    return array
```

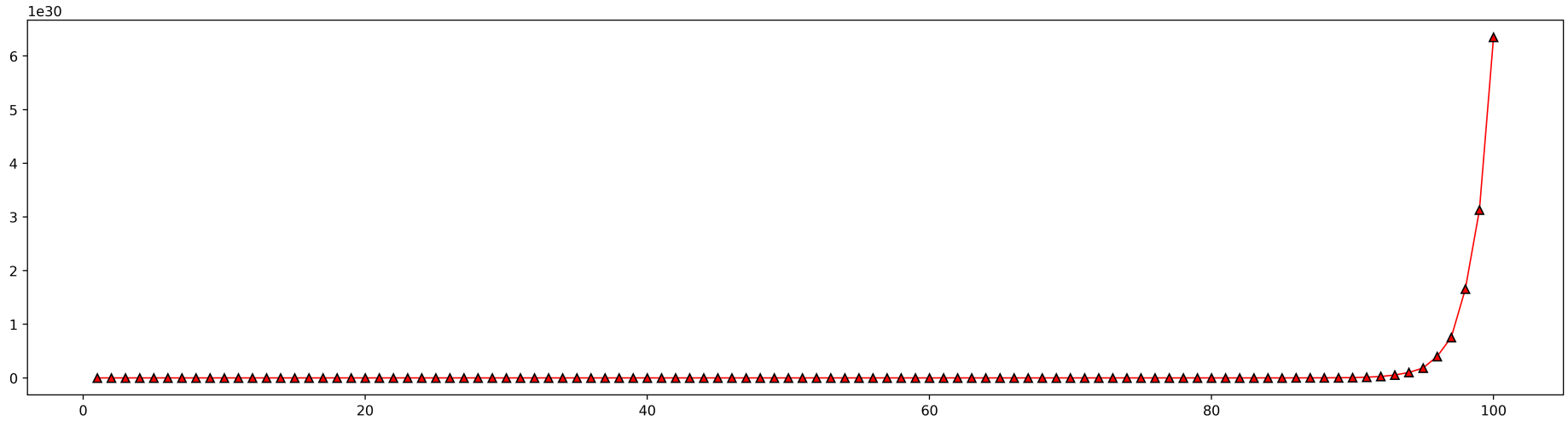
In [8]:
```
def CCC(n, k):
    C = [[0 for i in range(k + 1)]
            for j in range(n + 1)]
    for i in range(n + 1):
        for j in range(min(i, k) + 1):
            if (j == 0 or j == i):
                C[i][j] = 1
            else:
                C[i][j] = C[i-1][j-1] + C[i-1][j]
    return C[n][k]

def Sum_averages(N):
    N =int(N)
    array = Random_integer(N)
    result = 0.0
    sum = 0
    for i in range(N):
        sum += array[i]
    for n in range(1, N + 1):
        result += (sum * (CCC(N - 1, n - 1))) / n

    return result
```

In [9]:
```
Total_sum_averages = []
from tqdm import tqdm
for i in tqdm(range(1,101)):
    Total_sum_averages.append(Sum_averages(i))
import matplotlib.pyplot as plt
import numpy as np
x= list(range(1,101))
fig=plt.figure(figsize=(20,5),dpi=500)
Total_sum_averages= np.array(Total_sum_averages)
plt.plot(x,Total_sum_averages,color='r',linewidth=1,marker='^',markeredgecolor='k')
plt.show()
```

```
  0%|          | 0/100 [00:00<?, ?it/s]C:\Users\nicol\AppData\Local\Temp\ipykernel_16444\1555966595.py:20: RuntimeWarning: overflow
encountered in long_scalars
  result += (sum * (CCC(N - 1, n - 1))) / n
100%|██████████| 100/100 [00:01<00:00, 60.73it/s]
```



来源： [https://www.geeksforgeeks.org/sum-average-subsets/ (https://www.geeksforgeeks.org/sum-average-subsets/)](https://www.geeksforgeeks.org/sum-average-subsets/)

5. Path counting

5.1 [5 points] Create a matrix with N rows and M columns, fill the right-bottom corner and top-left corner cells with 1, and randomly fill the rest of matrix with integer 0 or 1.

5.2 [25 points] Consider a cell marked with 0 as a blockage or dead-end, and a cell marked with 1 is good to go. Write a function Count_path to count total number of paths to reach the right-bottom corner cell from the top-left corner cell.

Notice: for a given cell, you are only allowed to move either rightward or downward.

5.3 [5 points] Let N = 10, M = 8, run Count_path for 1000 times, each time the matrix (except the right-bottom corner and top-left corner cells, which remain being 1) is re-filled with integer 0 or 1 randomly, report the mean of total number of paths from the 1000 runs.

In [10]:
```
M= int(input('请输入矩阵的行：'))
N= int(input('请输入矩阵的列：'))
import numpy as np
matrix = np.random.randint(0,2, (N,M))
matrix[0,0] =1
matrix[N-1,M-1] =1
print('生成的矩阵为:')

print(matrix)

res = [0] * M  # 初始化一个长度为列长度的一维数组，
res[0] = 1
for line_idx in matrix:
    for col_idx, each in enumerate(line_idx):
        if each == 1:  # 如果值为1，不是障碍
            if col_idx != 0:  # 并且索引不为0，不是第一列
                res[col_idx] += res[col_idx - 1]  # 则等于列表当前值加上前一项的值，相当于这个列表再一直循环累加
        else:
            res[col_idx] = 0  # 是障碍时候则设置值为0
print('可行的路径数为：',res[-1])
```

```
请输入矩阵的行：10
请输入矩阵的列：10
生成的矩阵为:
[[1 1 1 0 0 0 0 0 0 1]
 [1 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 1 1 0 1 1 0]
 [1 0 0 0 0 1 1 0 1 1]
 [0 0 0 1 0 1 1 0 0 1]
 [0 1 1 1 0 1 1 0 0 0]
 [1 0 0 0 1 1 1 1 0 0]
 [0 0 0 0 1 0 1 0 0 0]
 [0 0 1 1 1 0 0 1 1 1]
 [1 0 0 1 1 1 0 1 1 1]]
可行的路径数为： 0
```

In [11]:
```
def Count_path(N,M,matrix):
    matrix = np.random.randint(0,2, (N,M))
    matrix[0,0] =1
    matrix[N-1,M-1] =1
#     print('生成的矩阵为:')
#     print(matrix)
    res = [0] * M  # 初始化一个长度为列长度的一维数组，
    res[0] = 1
    for line_idx in matrix:
        for col_idx, each in enumerate(line_idx):
            if each == 1:  # 如果值为1，不是障碍
                if col_idx != 0:  # 并且索引不为0，不是第一列
                    res[col_idx] += res[col_idx - 1]  # 则等于列表当前值加上前一项的值，相当于这个列表再一直循环累加
            else:
                res[col_idx] = 0  # 是障碍时候则设置值为0
#     print('可行的路径数为：',res[-1])
    return res[-1]
```

In [12]:
```
from tqdm import tqdm
path= np.zeros(shape = 1000)
for i in tqdm(range(1000)):
    matrix = np.random.randint(0,2,(10,8))
    matrix[0,0] =1
    matrix[10-1,8-1] =1
    path[i] = Count_path(10,8,matrix)

print ('1000次运行的路径总数的平均值为：',path.mean())
```

```
100%|██████████| 1000/1000 [00:00<00:00, 25493.10it/s]
```

1000次运行的路径总数的平均值为： 0.345

来源：https://zhuanlan.zhihu.com/p/43358393 (https://zhuanlan.zhihu.com/p/43358393)