Report of Assignment 01

Luyu Xiahou (12132226)

1. Flowchart:

```
print("T1")

import random

def Print_values():
    a = float(input("Input a = "))
    b = float(input("Input b = "))
    c = float(input("Input c = "))

    if a>b:
        if b>c:
            print(str(a)+", "+str(b)+", "+str(c))
        else:
            if a>c:
                print(str(a)+", "+str(c)+", "+str(b))
            else:
                print(str(c)+", "+str(a)+", "+str(b))
    else:
        if b>c:
            print(str(c)+", "+str(a)+", "+str(b))
        else:
            print(str(c)+", "+str(b)+", "+str(a))

Print_values()

print ("----------------------")
```

Output:

```
T1
Input a = -1
Input b = 3
Input c = 99
99.0, 3.0, -1.0
----------------------
```

```
T1
Input a = 9
Input b = 8
Input c = 7
9.0, 8.0, 7.0
----------------------
```

```
T1
Input a = 0
Input b = 22
Input c = -3
-3.0, 0.0, 22.0
----------------------
```

2.Matrix multiplication

```
print("T2")

import numpy as np
M1 = np.random.randint(0,50, size=(5, 10))
M2 = np.random.randint(0,50, size=(10, 5))

print("M1:")
print(M1)
print("M2:")
print(M2)

def Matrix_multip():
    a=np.zeros((5,5)) #5*5 0 matrix
    for i in range(0,5):
        for j in range(0,5):
            for m in range(0, 10):
                for n in range(0, 10):
                    a[i][j]=M1[i][m]*M2[n][j]
    print("M1*M2:")
    print(a)
    return a
Matrix_multip()

print ("----------------------")
```

Output:

```
T2
M1:
[[19 42 14 27 20 44 27 49 25 46]
 [26 47 41 20 44 40 15 34 22 47]
 [25 37 33 13 12 14 24 20 10 13]
 [41  4 22 13 35 41 32  0 46 17]
 [49 47 15 15  4 10 49 12  9 16]]
M2:
[[33 24 20 38  8]
 [46 44 36 37 35]
 [ 1 46  8 18 30]
 [40 20 47 14 25]
 [15 38 39 36  6]
 [34 16 19 14 40]
 [20 27 35 48 26]
 [25  4 31 37 18]
 [30 10 15 49 44]
 [ 5 36 31 36  7]]
M1*M2:
[[ 230. 1656. 1426. 1656.  322.]
 [ 235. 1692. 1457. 1692.  329.]
 [  65.  468.  403.  468.   91.]
 [  85.  612.  527.  612.  119.]
 [  80.  576.  496.  576.  112.]]
----------------------
```

```
T2
M1:
[[16 19 44 32 38 40 10 15  7 43]
 [ 4 35 21 46 42  5  5 12 39 19]
 [39 42  0  3 39 37 38 45 38 28]
 [45 34  1 10 23 49 19 12 32 20]
 [29 32 25 47 31 15  4 24 46 30]]
M2:
[[ 9  0 35 43 26]
 [45 14 38  5 35]
 [13 19 32 45  2]
 [ 8 49 18 30 16]
 [47  6 44 44  9]
 [33 23 34 39 20]
 [37 10 35 25  8]
 [32  8  5  7 13]
 [ 5 31  5  0  2]
 [47 15 24  2 23]]
M1*M2:
[[2021.  645. 1032.   86.  989.]
 [ 893.  285.  456.   38.  437.]
 [1316.  420.  672.   56.  644.]
 [ 940.  300.  480.   40.  460.]
 [1410.  450.  720.   60.  690.]]
----------------------
```

```
T2
M1:
[[25 46 36  5 43 12  4 44 44 12]
 [ 1 13 38 27 13  6 23 21 11 47]
 [15 23 45 31 26 26 25 14 29  9]
 [ 3  0 42 15 48  7  6  2 40 48]
 [13  1  1 13 28 36 29 18 17 34]]
M2:
[[24 33 25 24 23]
 [ 4 10 26 17 14]
 [20 44 18 19 23]
 [ 3 29  1 30 17]
 [45 11 43  9  3]
 [31 44  4 30 23]
 [18 10 20 44 31]
 [18 37 38 49 49]
 [18 32 14 46 22]
 [ 3 43 26 36 18]]
M1*M2:
[[  36.  516.  312.  432.  216.]
 [ 141. 2021. 1222. 1692.  846.]
 [  27.  387.  234.  324.  162.]
 [ 144. 2064. 1248. 1728.  864.]
 [ 102. 1462.  884. 1224.  612.]]
----------------------
```

3. Pascal triangle

```
print("T3")

import numpy as np
def Pascal_triangle(k):
    K = np.zeros((k, k))
    for x in range (0,k):
        K[x][0]=1
        K[x][x]=1
        for i in range(2,k):
            for j in range(1,i):
                K[i][j]=K[i-1][j-1]+K[i-1][j]
    #test print(K)
    print("The "+str(k)+"th line of the Pascal_triangle is:")
    print(K[k-1])
    #for i in range(1,k+1):
      #    if i%2==0:

Pascal_triangle(100)
Pascal_triangle(200)
print ("----------------------")
```

Output:

```
T3
The 100th line of the Pascal_triangle is:
[1.00000000e+00 9.90000000e+01 4.85100000e+03 1.56849000e+05
 3.76437600e+06 7.15231440e+07 1.12052926e+09 1.48870315e+10
 1.71200863e+11 1.73103095e+12 1.55792785e+13 1.26050526e+14
 9.24370525e+14 6.18617197e+15 3.80007707e+16 2.15337701e+17
 1.13052293e+18 5.51961194e+18 2.51448989e+19 1.07196674e+20
 4.28786696e+20 1.61305471e+21 5.71901217e+21 1.91462581e+22
 6.06298174e+22 1.81889452e+23 5.17685364e+23 1.39966784e+24
 3.59914587e+24 8.81170195e+24 2.05606379e+25 4.57640004e+25
 9.72485009e+25 1.97443926e+26 3.83273504e+26 7.11793650e+26
 1.26541093e+27 2.15461861e+27 3.51543037e+27 5.49849366e+27
 8.24774049e+27 1.18686997e+28 1.63901091e+28 2.17264238e+28
 2.76518120e+28 3.37966592e+28 3.96743390e+28 4.47391483e+28
 4.84674106e+28 5.04456723e+28 5.04456723e+28 4.84674106e+28
 4.47391483e+28 3.96743390e+28 3.37966592e+28 2.76518120e+28
 2.17264238e+28 1.63901091e+28 1.18686997e+28 8.24774049e+27
 5.49849366e+27 3.51543037e+27 2.15461861e+27 1.26541093e+27
 7.11793650e+26 3.83273504e+26 1.97443926e+26 9.72485009e+25
 4.57640004e+25 2.05606379e+25 8.81170195e+24 3.59914587e+24
 1.39966784e+24 5.17685364e+23 1.81889452e+23 6.06298174e+22
 1.91462581e+22 5.71901217e+21 1.61305471e+21 4.28786696e+20
 1.07196674e+20 2.51448989e+19 5.51961194e+18 1.13052293e+18
 2.15337701e+17 3.80007707e+16 6.18617197e+15 9.24370525e+14
 1.26050526e+14 1.55792785e+13 1.73103095e+12 1.71200863e+11
 1.48870315e+10 1.12052926e+09 7.15231440e+07 3.76437600e+06
 1.56849000e+05 4.85100000e+03 9.90000000e+01 1.00000000e+00]
```

```
The 200th line of the Pascal_triangle is:
[1.00000000e+00 1.99000000e+02 1.97010000e+04 1.29369900e+06
 6.33912510e+07 2.47225879e+09 7.99363675e+10 2.20395985e+12
 5.28950363e+13 1.12255022e+15 2.13284541e+16 3.66461620e+17
 5.74123205e+18 8.25854149e+19 1.09720623e+21 1.35322101e+22
 1.55620416e+23 1.67520801e+24 1.69382143e+25 1.61358779e+26
 1.45222901e+27 1.23785235e+28 1.00153508e+29 7.70746561e+29
 5.65214145e+30 3.95649902e+31 2.64781088e+32 1.69656030e+33
 1.04217276e+34 6.14522558e+34 3.48229449e+35 1.89841216e+36
 9.96666383e+36 5.04373594e+37 2.46252990e+38 1.16090695e+39
 5.28857612e+39 2.32983218e+40 9.93244246e+40 4.10031599e+41
 1.64012640e+42 6.36049017e+42 2.39275583e+43 8.73634104e+43
 3.09743000e+44 1.06689256e+45 3.57177074e+45 1.16272537e+46
 3.68196366e+46 1.13464594e+47 3.40393783e+47 9.94483799e+47
 2.83045389e+48 7.85050418e+48 2.12254372e+49 5.59579709e+49
 1.43891925e+50 3.60992023e+50 8.83808056e+50 2.11215145e+51
 4.92835339e+51 1.12301823e+52 2.49962123e+52 5.43568426e+52
 1.15508290e+53 2.39901834e+53 4.87073421e+53 9.66877089e+53
 1.87687905e+54 3.56335009e+54 6.61765016e+54 1.20236179e+55
 2.13753207e+55 3.71872018e+55 6.33187490e+55 1.05531248e+56
 1.72182563e+56 2.75044873e+56 4.30198392e+56 6.58911461e+56
 9.88367191e+56 1.45204563e+57 2.08952907e+57 2.94548074e+57
 4.06756864e+57 5.50318111e+57 7.29491449e+57 9.47500388e+57
 1.20590958e+58 1.50399959e+58 1.83822173e+58 2.20182602e+58
 2.58475229e+58 2.97385478e+58 3.35349582e+58 3.70649538e+58
 4.01536999e+58 4.26374340e+58 4.43777374e+58 4.52742573e+58
 4.52742573e+58 4.43777374e+58 4.26374340e+58 4.01536999e+58
 3.70649538e+58 3.35349582e+58 2.97385478e+58 2.58475229e+58
 2.20182602e+58 1.83822173e+58 1.50399959e+58 1.20590958e+58
 9.47500388e+57 7.29491449e+57 5.50318111e+57 4.06756864e+57
 2.94548074e+57 2.08952907e+57 1.45204563e+57 9.88367191e+56
 6.58911461e+56 4.30198392e+56 2.75044873e+56 1.72182563e+56
 1.05531248e+56 6.33187490e+55 3.71872018e+55 2.13753207e+55
 1.20236179e+55 6.61765016e+54 3.56335009e+54 1.87687905e+54
 9.66877089e+53 4.87073421e+53 2.39901834e+53 1.15508290e+53
 5.43568426e+52 2.49962123e+52 1.12301823e+52 4.92835339e+51
 2.11215145e+51 8.83808056e+50 3.60992023e+50 1.43891925e+50
 5.59579709e+49 2.12254372e+49 7.85050418e+48 2.83045389e+48
 9.94483799e+47 3.40393783e+47 1.13464594e+47 3.68196366e+46
 1.16272537e+46 3.57177074e+45 1.06689256e+45 3.09743000e+44
 8.73634104e+43 2.39275583e+43 6.36049017e+42 1.64012640e+42
 4.10031599e+41 9.93244246e+40 2.32983218e+40 5.28857612e+39
 1.16090695e+39 2.46252990e+38 5.04373594e+37 9.96666383e+36
 1.89841216e+36 3.48229449e+35 6.14522558e+34 1.04217276e+34
 1.69656030e+33 2.64781088e+32 3.95649902e+31 5.65214145e+30
 7.70746561e+29 1.00153508e+29 1.23785235e+28 1.45222901e+27
 1.61358779e+26 1.69382143e+25 1.67520801e+24 1.55620416e+23
 1.35322101e+22 1.09720623e+21 8.25854149e+19 5.74123205e+18
 3.66461620e+17 2.13284541e+16 1.12255022e+15 5.28950363e+13
 2.20395985e+12 7.99363675e+10 2.47225879e+09 6.33912510e+07
 1.29369900e+06 1.97010000e+04 1.99000000e+02 1.00000000e+00]
--------------------
```

4. Add or double

```
print("T4")

def Least_moves():
    x=random.randint(1,100)
    # test x=2 and x=5
    print ("When x="+str(x))
    for i in range(1,x):
        if pow(2,i-1)<x<pow(2,i):
            if x-pow(2,i-1)>pow(2,i):
                b=i+(pow(2,i)-x)
                break
            else:
                b=(i-1)+(x-pow(2,i-1))
                break
    # test print(i)
    print("The smallest number of moves is "+str(b))

Least_moves()
print ("----------------------")
```

Output:

```
T4
When x=62
The smallest number of moves is 35
----------------------
```

```
T4
When x=75
The smallest number of moves is 17
----------------------
```

```
T4
When x=67
The smallest number of moves is 9
----------------------
```

5. Dynamic programming

```
print ("T5")

import random
import numpy as np
from collections import Counter

def sjz(r):
    yushu = []
    R=int(r)
    while R != 0:
        s=R%3
        yushu.append(s)
        R=R//3
    yushu.reverse()
    T = ''.join(str(s) for s in yushu)
    t = T.zfill(9)     #3^9（三进制 1000000000）
    # print(t)    #test
    return t

# Xingyu Nan and Tianci Jiang explained to me how to use ternary in T5.1
# 十进制转三进制方法参考：https://blog.51cto.com/u_16213459/7819614

def Find_expression(x):

    K1 = []
    for u in range(0, pow(3, 9)):
        K1.append(sjz(u))
    # print(K1)    # test
    K2 = []

    for j in range(0,pow(3, 9)):
        A = ""    # 储存式子
        sum = 0    # 除了最后一项的和
        b = 0    # 待加的最后一项
        c = 0    # 运算符号
        for i in range(1,10):
            M1="1"+K1[j]            #读取 1 个 str，前面加 1，变成 int 时保留前面的 0
            M2=int(M1)
            a = (M2 // pow(10, 9 - i))%10        #依次取出符号
            #print("M1 : " + str(M1)) #test
            #print("M2 : " + str(M2)) #test
            #print("a : " + str(a)) #test
```

```
            #a=random.randint(0,2) #test
            if a==1:    #+
                A=A+"+"+str(i)
                sum=sum+b
                b=i
                c=a
            elif a==2: #-
                A=A+"-"+str(i)
                sum=sum+b
                b=-i
                c=a
            else:
                A = A+str(i)
                if i==1:
                    c=1
                    break    #首个是+1、1 重复，去掉 1 开头的
                if c==1:
                    b = b*10+i
                elif c==2:
                    b = b*10-i
        if i==9:    # test
          sum = sum + b
          K2.append(sum)
          if x==sum:
              print(str(x)+"="+A)

    Total_solutions = []    #T5.2

    for d in range(1,101):
        Total_solutions.append(K2.count(d)) #结果为 i 的个数

    suoying_max=[]       #检查有没有多个最大值、最小值
    suoying_min = []
    for e1 in range(0,100):
        if Total_solutions[e1]==max(Total_solutions):
            suoying_max.append(e1+1)
        if Total_solutions[e1]==min(Total_solutions):
            suoying_min.append(e1+1)

    #print("Total_solutions: " + str(Total_solutions))      #test
    print("The maximum number(s): " + str(suoying_max))
    print("The minimum number(s) : " + str(suoying_min))

#    print("i : "+str(i))      #test
```

```
#    print("A : "+str(A))      #test
#    print("sum : "+str(sum))      #test
#    print("b : "+str(b))      #test
#    print("c : " + str(c))      #test
```

Find_expression(50)

Output:

```
T5
50=+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=-12+3+45+6+7-8+9
50=-12+3+4+5+67-8-9
50=-12+3-4-5+67-8+9
50=-12-3+4-5+67+8-9
50=-1+23-4+56-7-8-9
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=-1-2+34+5+6+7-8+9
50=-1-2+3+4+56+7-8-9
```

```
The maximum number(s): [9]
The minimum number(s) : [100]
```