1. In this question, I use if & else to solve it.
   My outputs with some random a, b, and c values:
   The first one:
   ```
   please input the first number: 2
   please input the second number: 1
   please input the third number: 3
   3, 2, 1
   ```
   The second one:
   ```
   please input the first number: 10
   please input the second number: 1
   please input the third number: 5
   10, 5, 1
   ```
   The third one:
   ```
   please input the first number: 2
   please input the second number: 4
   please input the third number: 5
   5, 4, 2
   ```
   The fourth one:
   ```
   please input the first number: 5
   please input the second number: 3
   please input the third number: 1
   5, 3, 1
   ```
   The fifth one:
   ```
   please input the first number: 2
   please input the second number: 10
   please input the third number: 3
   ```
   The sixth one:
   ```
   please input the first number: 3
   please input the second number: 10
   please input the third number: 2
   ```
   (As the flowchart shows, the fifth one and the sixth one exports nothing.)
2. In this question, I use function numpy and function random to creat random matrixes (refer to *https://blog.csdn.net/qq_43287650/article/details/82860938*), and then I use 3 for loops to solve the question of matrix multiplication.
3. In this question, I define a list to contain all of the numbers in k^{th} line (refer to *https://blog.csdn.net/qq_31672701/article/details/100710080*). For each number, I use combinatorial number.
   ```
   Pascal_triangle(100) = [1.0, 99.0, 4851.0, 156849.0, 3764376.0, 71523144.0,
   1120529256.0, 14887031544.0, 171200862756.0, 1731030945644.0, 15579278510796.0,
   126050526132804.0, 924370524973896.0, 6186171974825304.0,
   3.8000770702498296e+16, 2.1533770064749034e+17, 1.1305229283993243e+18,
   5.519611944537878e+18, 2.514489885845033e+19, 1.0719667408076194e+20,
   4.2878669632304775e+20, 1.6130547147390845e+21, 5.719012170438572e+21,
   1.914625813581609e+22, 6.062981743008428e+22, 1.8188945229025285e+23,
   ```

```
5.1768536421071965e+23, 1.3996678365697236e+24, 3.599145865465003e+24,
8.811701946483283e+24, 2.0560637875127662e+25, 4.576400043173577e+25,
9.72485009174385e+25, 1.974439261051024e+26, 3.8327350361578704e+26,
7.117936495721758e+26, 1.265410932572757e+27, 2.154618614921181e+27,
3.515430371713506e+27, 5.498493658321124e+27, 8.247740487481687e+27,
1.1868699725888282e+28, 1.6390109145274292e+28, 2.1726423750712436e+28,
2.765181204636128e+28, 3.37966591677749e+28, 3.96743390230401e+28,
4.473914826002394e+28, 4.846741061502594e+28, 5.04456722727821e+28,
5.04456722727821e+28, 4.846741061502594e+28, 4.473914826002394e+28,
3.96743390230401e+28, 3.37966591677749e+28, 2.765181204636128e+28,
2.1726423750712436e+28, 1.6390109145274292e+28, 1.1868699725888282e+28,
8.247740487481687e+27, 5.498493658321124e+27, 3.515430371713506e+27,
2.154618614921181e+27, 1.265410932572757e+27, 7.117936495721758e+26,
3.8327350361578704e+26, 1.974439261051024e+26, 9.72485009174385e+25,
4.576400043173577e+25, 2.0560637875127662e+25, 8.811701946483283e+24,
3.599145865465003e+24, 1.3996678365697236e+24, 5.1768536421071965e+23,
1.8188945229025285e+23, 6.062981743008428e+22, 1.914625813581609e+22,
5.719012170438572e+21, 1.6130547147390845e+21, 4.2878669632304775e+20,
1.0719667408076194e+20, 2.514489885845033e+19, 5.519611944537878e+18,
1.1305229283993243e+18, 2.1533770064749034e+17, 3.8000770702498296e+16,
6186171974825304.0, 924370524973896.0, 126050526132804.0, 15579278510796.0,
1731030945644.0, 171200862756.0, 14887031544.0, 1120529256.0, 71523144.0,
3764376.0, 156849.0, 4851.0, 99.0, 1.0]
Pascal_triangle(200) = [1.0, 199.0, 19701.0, 1293699.0, 63391251.0, 2472258789.0,
79936367511.0, 2203959847089.0, 52895036330136.0, 1122550215450664.0,
2.1328454093562616e+16, 3.664616203348486e+17, 5.741232051912628e+18,
8.258541490058933e+19, 1.0972062265364012e+21, 1.3532210127282282e+22,
1.5562041646374625e+23, 1.6752080125215036e+24, 1.6938214348828537e+25,
1.61358778796735e+26, 1.452229009170615e+27, 1.2378523459120956e+28,
1.0015350798743319e+29, 7.707465614685076e+29, 5.652141450769056e+30,
3.956499015538339e+31, 2.6478108796295038e+32, 1.6965603043552007e+33,
1.0421727583896233e+34, 6.145225575331917e+34, 3.482294492688086e+35,
1.8984121589170535e+36, 9.96666383431453e+36, 5.043735940395535e+37,
2.462529900310761e+38, 1.1609069530036446e+39, 5.2885761192388254e+39,
2.3298321822592664e+40, 9.932442461210556e+40, 4.100315990397178e+41,
1.6401263961588712e+42, 6.360490170469769e+42, 2.3927558260338655e+43,
8.736341039239928e+43, 3.097430004821429e+44, 1.06689255721627e+45,
3.571770735028382e+45, 1.1627253669347713e+46, 3.6819636619601086e+46,
1.134645944808115e+47, 3.4039378344243454e+47, 9.944837986847597e+47,
2.830453888564316e+48, 7.850504181489708e+48, 2.1225437231435135e+49,
5.595797088287444e+49, 1.4389192512739143e+50, 3.609920226880171e+50,
8.838080555465246e+50, 2.1121514547806774e+51, 4.9283533944882477e+51,
1.123018232514535e+52, 2.4996212272097715e+52, 5.435684255995853e+52,
1.1550829043991188e+53, 2.399018339905862e+53, 4.870734205263416e+53,
9.66877088507514e+53,    1.8768790541616448e+54,    3.563350088335876e+54,
6.617650164052342e+54,    1.2023617903700734e+55,    2.1375320717690194e+55,
3.7187201796529513e+55,    6.33187490049016e+55,    1.0553124834150268e+56,
1.7218256308350438e+56,    2.7504487349702646e+56,    4.301983918799644e+56,
6.589114609807051e+56,    9.883671914710577e+56,    1.4520456269759982e+57,
2.089529072965461e+57,    2.9454807414091436e+57,    4.067568642898341e+57,
5.503181105097756e+57,    7.294914488152839e+57,    9.475003875416906e+57,
1.2059095841439698e+58,    1.503999593707648e+58,    1.8382217256426806e+58,
2.2018260230225515e+58,    2.584752287896039e+58,    2.9738547828481305e+58,
3.3534958189564025e+58,    3.70649537884655e+58,    4.015369993750429e+58,
4.2637433954257135e+58,    4.437773738096151e+58,    4.527425732805164e+58,
4.527425732805164e+58,    4.437773738096151e+58,    4.2637433954257135e+58,
4.015369993750429e+58,    3.70649537884655e+58,    3.3534958189564025e+58,
2.9738547828481305e+58,    2.584752287896039e+58,    2.2018260230225515e+58,
1.8382217256426806e+58,    1.503999593707648e+58,    1.2059095841439698e+58,
9.475003875416906e+57,    7.294914488152839e+57,    5.503181105097756e+57,
4.067568642898341e+57,    2.9454807414091436e+57,    2.089529072965461e+57,
1.4520456269759982e+57,    9.883671914710577e+56,    6.589114609807051e+56,
4.301983918799644e+56,    2.7504487349702646e+56,    1.7218256308350438e+56,
1.0553124834150268e+56,    6.33187490049016e+55,    3.7187201796529513e+55,
2.1375320717690194e+55,    1.2023617903700734e+55,    6.617650164052342e+54,
3.563350088335876e+54,    1.8768790541616448e+54,    9.66877088507514e+53,
4.870734205263416e+53,    2.399018339905862e+53,    1.1550829043991188e+53,
5.435684255995853e+52,    2.4996212272097715e+52,    1.123018232514535e+52,
4.9283533944882477e+51,    2.1121514547806774e+51,    8.838080555465246e+50,
3.609920226880171e+50,    1.4389192512739143e+50,    5.595797088287444e+49,
2.1225437231435135e+49,    7.850504181489708e+48,    2.830453888564316e+48,
9.944837986847597e+47,    3.4039378344243454e+47,    1.134645944808115e+47,
3.6819636619601086e+46,    1.1627253669347713e+46,    3.571770735028382e+45,
1.06689255721627e+45,    3.097430004821429e+44,    8.736341039239928e+43,
2.3927558260338655e+43,    6.360490170469769e+42,    1.6401263961588712e+42,
4.100315990397178e+41,    9.932442461210556e+40,    2.3298321822592664e+40,
5.2885761192388254e+39,    1.1609069530036446e+39,    2.462529900310761e+38,
5.043735940395535e+37,    9.96666383431453e+36,    1.8984121589170535e+36,
3.482294492688086e+35,    6.145225575331917e+34,    1.0421727583896233e+34,
1.6965603043552007e+33,    2.6478108796295038e+32,    3.956499015538339e+31,
5.652141450769056e+30,    7.707465614685076e+29,    1.0015350798743319e+29,
1.2378523459120956e+28,    1.452229009170615e+27,    1.61358778796735e+26,
1.6938214348828537e+25,    1.6752080125215036e+24,    1.5562041646374625e+23,
1.3532210127282282e+22,    1.0972062265364012e+21,    8.258541490058933e+19,
5.741232051912628e+18,    3.664616203348486e+17,    2.1328454093562616e+16,
1122550215450664.0,    52895036330136.0,    2203959847089.0,    79936367511.0,
2472258789.0, 63391251.0, 1293699.0, 19701.0, 199.0, 1.0]
```

4. In this question, I use reverse method, to get the resolution from the number we need. I use

a while loop so that it would run if x unequals to 1. In the while loop, I use if & else so that it could judge what is the shortest path from x to 1.

5. In this question, I creat a matrix including 1, 12, 123, 23… With this matrix, I creat a list whose elements are lists then. Each list include all of the number from 1 to 9 in order, and it shows as [1, 2, 3, 4, 5, 6, 7, 8, 9], [1, 234, 56, 789], etc. In this part, I use 8 loops, all with if & else. Next, I use another 8 loops with if & else to solve the question of judging the addition or subtraction. Finally, I get all of the expressions with specific result.
   In the second question, I creat a list to contain all of the expressions with specific results in range. With max & min function, index function, and count function, I use lists to get the maximum number and minimum number and the numbers yielding them.