PS1_1:

CODE:

```
def Print_values(a,b,c):
    if a > b:
        if b > c:
            print(a,b,c)
        else:
            if a > c:
                print(a,c,b)
            else:
                print(c,a,b)
    else:
        if b > c:
            '''
            if a > c:
                print(b,a,c)
            else:
                print(b,c,a)
                '''
        else:
            print(c,b,a)
    return
```

RESULTS:

```
In [203]: Print_values(1,2,3)
3 2 1

In [204]: Print_values(3,2,1)
3 2 1

In [205]: Print_values(2,1,3)
3 2 1
```

PS1_2:

2.1

CODE:

```
import numpy as np

M1 = np.random.randint(0,50,(5,10))
M2 = np.random.randint(0,50,(10,5))
```

RESULT:

```
In [222]: import numpy as np
     ...:
     ...: M1 = np.random.randint(0,50,(5,10))
     ...: M2 = np.random.randint(0,50,(10,5))

In [223]: M1
Out[223]:
array([[48, 26, 42,  7,  3, 16, 14, 44,  8, 38],
       [ 9, 11,  8, 13, 15, 43, 14, 15, 14,  0],
       [32, 48,  7, 19, 10, 24, 17, 17, 25, 10],
       [21, 31, 42, 43, 35, 17, 44, 17,  2, 14],
       [ 9, 14,  5, 23, 29, 26, 11, 48, 29, 10]])

In [224]: M2
Out[224]:
array([[49, 29, 10, 25, 38],
       [ 9, 25, 13, 10, 17],
       [ 4, 31,  1,  7, 13],
       [16, 25, 12, 16, 47],
       [43, 19,  9,  5, 10],
       [24, 39,  4, 40, 27],
       [ 7, 22,  8, 48, 12],
       [10, 27,  8, 44, 10],
       [10, 40, 26, 29, 39],
       [41, 34, 15, 29, 27]])
```

2.2

CODE:

```
import numpy as np

M1 = np.random.randint(0,50,(5,10))
M2 = np.random.randint(0,50,(10,5))
```

```
def Matrix_multip(A,B):
    m=np.zeros([5,5],dtype=int)
    for i in range(5):
        for j in range(5):
            for k in range(10):
                m[i,j]+= M1[i,k] * M2[k,j]
    return m

Matrix_multip(M1,M2)
```

RESULTS:

```
Out[225]:
array([[5555, 7308, 2277, 6463, 5549],
       [2845, 4344, 1300, 4132, 3419],
       [4287, 6119, 2437, 5222, 5383],
       [5149, 7072, 2304, 5996, 5855],
       [4083, 5944, 2294, 5724, 4731]])
```

PS1_3:

CODE:

```
def Pascal_triangle(k):
    x = []
    for i in range(k):
        if i == 0:
            x.append([1])
        elif i == 1:
            x.append([1,1])
        else:
            y = []
            for j in range(i+1):
                if j == 0 or j == i:
                    y.append(1)
                else:
                    y.append(x[i-1][j-1]+x[i-1][j])
            x.append(y)
    return x[k-1]
```

RESULT:

```
In [211]:Pascal_triangle(100)
Out[211]:
[1,
 99,
 4851,
 156849,
 3764376,
 71523144,
 1120529256,
 14887031544,
 171200862756,
 1731030945644,
 15579278510796,
 126050526132804,
 924370524973896,
 6186171974825304,
 38000770702498296,
 215337700647490344,
 1130522928399324306,
 5519611944537877494,
 25144898858450330806,
 107196674080761936594,
```

42878669632304774637 6,
1613054714739084379224,
5719012170438571889976,
19146258135816088501224,
60629817430084280253876,
18188945229025284076162 8,
51768536421071962370617 2,
13996678365697234270574 28,
35991458654650030981476 72,
88117019464832834471891 28,
20560637875127661376774632,
45764000431735762419272568,
97248500917438495140954207,
19744392610510239922557 3693,
38327350361578701026140 7757,
71179364957217587619975 7263,
12654109325727571132440 12912,
21546186149211810306587 24688,
35154303717135058921273 92912,
54984936583211246005069 47888,
82477404874816869007604 21832,
11868699725888281149874753368,
16390109145274293016493707032,
21726423750712434928840495368,
27651812046361280818524266832,
33796659167774898778196326128,
39674339023040098565708730672,
44739148260023940935799206928,
48467410615025936013782474172,
50445672272782096667406248628,
50445672272782096667406248628,
48467410615025936013782474172,
44739148260023940935799206928,
39674339023040098565708730672,
33796659167774898778196326128,
27651812046361280818524266832,
21726423750712434928840495368,
16390109145274293016493707032,
11868699725888281149874753368,
82477404874816869007604 21832,
54984936583211246005069 47888,
35154303717135058921273 92912,
21546186149211810306587 24688,
12654109325727571132440 12912,

711793649572175876199757263,
383273503615787010261407757,
19744392610510239922557369 3,
97248500917438495140954207,
45764000431735762419272568,
20560637875127661376774632,
88117019464832834471891 28,
35991458654650030981476 72,
13996678365697234270574 28,
51768536421071962370617 2,
18188945229025284076162 8,
60629817430084280253876,
19146258135816088501224,
571901217043857188997 6,
161305471473908437922 4,
428786696323047746376,
107196674080761936594,
25144898858450330806,
5519611944537877494,
1130522928399324306,
215337700647490344,
38000770702498296,
6186171974825304,
924370524973896,
126050526132804,
15579278510796,
1731030945644,
171200862756,
14887031544,
1120529256,
71523144,
3764376,
156849,
4851,
99,
1]

Removing all variables...

```
def Pascal_triangle(k):
    x = []
```

```
    for i in range(k):
        if i == 0:
            x.append([1])
        elif i == 1:
            x.append([1,1])
        else:
            y = []
            for j in range(i+1):
                if j == 0 or j == i:
                    y.append(1)
                else:
                    y.append(x[i-1][j-1]+x[i-1][j])
            x.append(y)
    return x[k-1]

Pascal_triangle(100)
Out[211]:
[1,
 99,
 4851,
 156849,
 3764376,
 71523144,
 1120529256,
 14887031544,
 171200862756,
 1731030945644,
 15579278510796,
 126050526132804,
 924370524973896,
 6186171974825304,
 38000770702498296,
 215337700647490344,
 1130522928399324306,
 5519611944537877494,
 25144898858450330806,
 107196674080761936594,
 428786696323047746376,
 1613054714739084379224,
 5719012170438571889976,
 19146258135816088501224,
 606298174300842802538 76,
 1818894522902528407616 28,
 5176853642107196237061 72,
```

139966783656972342705742 8,
359914586546500309814767 2,
881170194648328344718912 8,
205606378751276613767746 32,
457640004317357624192725 68,
972485009174384951409542 07,
197443926105102399225573 693,
383273503615787010261407 757,
711793649572175876199757 263,
126541093257275711324401 2912,
215461861492118103065872 4688,
351543037171350589212739 2912,
549849365832112460050694 7888,
824774048748168690076042 1832,
118686997258882811149874 753368,
163901091452742930164937 07032,
217264237507124349288404 95368,
276518120463612808185242 66832,
337966591677748987781963 26128,
396743390230400985657087 30672,
447391482600239409357992 06928,
484674106150259360137824 74172,
504456722727820966674062 48628,
504456722727820966674062 48628,
484674106150259360137824 74172,
447391482600239409357992 06928,
396743390230400985657087 30672,
337966591677748987781963 26128,
276518120463612808185242 66832,
217264237507124349288404 95368,
163901091452742930164937 07032,
118686997258882811149874 753368,
824774048748168690076042 1832,
549849365832112460050694 7888,
351543037171350589212739 2912,
215461861492118103065872 4688,
126541093257275711324401 2912,
711793649572175876199757 263,
383273503615787010261407 757,
197443926105102399225573 693,
972485009174384951409542 07,
457640004317357624192725 68,
205606378751276613767746 32,
881170194648328344718912 8,

```
 359914586546500309814767 2,
 139966783656972342705742 8,
 517685364210719623706172,
 181889452290252840761628,
 60629817430084280253876,
 191462581358160885012 24,
 5719012170438571889976,
 161305471473908437922 4,
 428786696323047746376,
 107196674080761936594,
 25144898858450330806,
 5519611944537877494,
 1130522928399324306,
 215337700647490344,
 38000770702498296,
 6186171974825304,
 924370524973896,
 126050526132804,
 15579278510796,
 1731030945644,
 171200862756,
 14887031544,
 1120529256,
 71523144,
 3764376,
 156849,
 4851,
 99,
 1]
In [211]:Pascal_triangle(200)
Out[213]:
[1,
 199,
 19701,
 1293699,
 63391251,
 2472258789,
 79936367511,
 2203959847089,
 52895036330136,
 1122550215450664,
 21328454093562616,
 366461620334848584,
 5741232051912627816,
```

8258541490058933858 4,
109720622653640121261 6,
135322101272822816222 64,
155620416463746238656036,
167520801252150362788556 4,
169382143488285366819540 36,
161358778796735007338614764,
145222900917061506604753287 6,
123785234591209569915480183 24,
100153507987433197477070330076,
770746561468507650149628192324,
565214145076905610109727341037 6,
395649901553833927076809138726 32,
264781087962950397351403038993768,
169656030435520069414047132392303 2,
104217275838962328354343238469557 68,
614522557533191660296299785458426 32,
348229449268808607501236545093108248,
189841215891705337637770890712049335 2,
996666383431453022598297176238259009 8,
504373594039553499314895843732694711 02,
246252990031076120253743264881256829498,
116090695300364456691050396301163933906 2,
528857611923882524925896249816413476683 8,
232983218225926625845732672216419991079 62,
993244246121055615447597181554211540918 38,
410031599039717830992469605718533482276562,
164012639615887132396987842287413392910624 8,
636049017046976928076123583504847060311935 2,
239275582603386558657208395699442465545918 48,
873634103923992783934458561042150397458353 52,
309743000482142896122217126187671504553416248,
106689255721626997553208121242420184901732263 2,
357177073502838209199870666768102358149277576 8,
116272536693477119165064280884084384674126530 32,
368196366196010877356036889466262721813473401268,
113464594480811515266860347570217040690499665132,
340393783442434545800581042710651122071498995396,
994483798684759751456599516938961121346144123804,
283045388856431621568416785590319703767748712159 6,
785050418148970723972778631731641442525642654480 4,
212254372314351343888936444875591945571747828803 96,
559579708828744452070832445581106038325517003210 44,
143891925127391430532499771720855838426561515111256,

360992022688017097651709953615480436754356081770344,
883808055546524618388669196782727965846871786403256,
211215145478067747784410774146380751160015121835354 4,
492835339448824744830291806341555086040035284282493 6,
112301823251453507428541903412255995015680171336502 64,
249962122720977161695786817272440763099417155555441 36,
543568425599585256386076094703561659438415084303102 64,
115508290439911866982041170124506852630663205414409 311,
239901833990586185270393199489360386232915888168388 569,
487073420526341648882313465629913511442586803250970 731,
966877088507514019423099864608634283908418579587747 869,
187687905416164486123307620776970184523398900743503 9981,
356335008833587647567439106112798466269061681121724 9819,
661765016405234202625244054209482865928257407797489 2521,
120236179037007341040361243652145478457387613529402 97679,
213753207176901939627308877603814183924244646274494 18096,
371872017965295155242030513091567141895603699683024 12304,
633187490049016075141835738507262971335757650811635 66896,
105531248341502679190305956417877161889292941801939 278160,
172182563083504371310499192050220632556214799782111 453840,
275044873497026463262225982106196594862524939911684 530160,
430198391879964468179379100217384417605487726528532 213840,
658911460980705071515251533244348285193215378606992 378160,
988367191471057607272877299866522427789823067910488 567240,
145204562697599821315398023066810085070356722322652 0240760,
208952907296546084331914228315653537052464551635035 8395240,
294548074140914359841373068830499564278775331822881 8460760,
406756864289834115971419952194499398289737362993503 5017240,
550318110509775568667215229439616832980232902873563 5611560,
729491448815283893349564373908329290229611057297514 4880440,
947500387541690574120698554616565629838460338788725 7143560,
120590958414396982160816179678471989252167679482201 45455440,
150399959370764775503939280273150458505512499129487 20736560,
183822172564268058949259120333850560395626387824928 80900240,
220182602302255147532629056224062759155200838163925 71627760,
258475228789603868842651500784769325964800983931564 97128240,
297385478284813053399609791225487289013265648179327 44007760,
335349581895640251705942956062783538674533603260500 9200240,
370649537884655015043410635648339700640273982557213 25958160,
401536999375042932963694855285701342360296814436981 03121340,
426374339542571361806810000973476683124851256567103 56922660,
443777373809615090860149184686679813048314573161679 22511340,
452742573280516405827020885387420819372522948377066 68420660,
452742573280516405827020885387420819372522948377066 68420660,

443777373809615090860149184686679813048314573161679225113 40,
426374339542571361806810000973476683124851256567103569226 60,
401536999375042932963694855285701342360296814436981031213 40,
370649537884655015043410635648339700640273982557213259581 60,
335349581895640251705942956062783538674533603260500920024 0,
297385478284813053399609791225487289013265648179327440077 60,
258475228789603868842651500784769325964800983931564971282 40,
220182602302255147532629056224062759155200838163925716277 60,
183822172564268058949259120333850560395626387824928809002 40,
150399959370764775503939280273150458505512499129487207365 60,
120590958414396982160816179678471989252167679482201454554 40,
947500387541690574120698554616565629838460338788725714356 0,
729491448815283893349564373908329290229611057297514488044 0,
550318110509775568667215229439616832980232902873563561156 0,
406756864289834115971419952194499398289737362993503501724 0,
294548074140914359841373068830499564278775331822881846076 0,
208952907296546084331914228315653537052464551635035839524 0,
145204562697599821315398023066810085070356722322652024076 0,
988367191471057607272877299866522427789823067910488567240,
658911460980705071515251533244348285193215378606992378160,
430198391879964468179379100217384417605487726528532213840,
275044873497026463262225982106196594862524939911684530160,
172182563083504371310499192050220632556214799782111453840,
105531248341502679190305956417877161889292941801939278160,
633187490049016075141835738507262971335757650811635668 96,
371872017965295155242030513091567141895603699683024123 04,
213753207176901939627308877603814183924244646274494180 96,
120236179037007341040361243652145478457387613529402976 79,
661765016405234202625244054209482865928257407797489252 1,
356335008833587647567439106112798466269061681121724981 9,
187687905416164486123307620776970184523398900743503998 1,
966877088507514019423099864608634283908418579587747869,
487073420526341648882313465629913511442586803250970731,
239901833990586185270393199489360386232915888168388569,
115508290439911866982041170124506852630663205414409311,
543568425599585256386076094703561659438415084303102 64,
249962122720977161695786817272440763099417155555441 36,
112301823251453507428541903412255995015680171336502 64,
492835339448824744830291806341555086040035284282493 6,
211215145478067747784410774146380751160015121835354 4,
883808055546524618388669196782727965846871786403256,
360992022688017097651709953615480436754356081770344,
143891925127391430532499771720855838426561515111256,
559579708828744452070832445581106038325517003210 44,

212254372314351343888936444875591945571747828803 96,
785050418148970723972778631731641442525642654480 4,
283045388856431621568416785590319703767748712159 6,
994483798684759751456599516938961121346144123804,
340393783442434545800581042710651122071498995396,
113464594480811515266860347570217040690499665132,
368196366196010877356036889466262721813473401268,
116272536693477119165064280884084384674126530 32,
357177073502838209199870666768102358149277576 8,
106689255721626997553208121242420184901732263 2,
309743000482142896122217126187671504553416248,
873634103923992783934458561042150397458353 52,
239275582603386558657208395699442465545918 48,
636049017046976928076123583504847060311935 2,
164012639615887132396987842287413392910624 8,
410031599039717830992469605718533482276562,
993244246121055615447597181554211540918 38,
232983218225926625845732672221641999107962,
528857611923882524925896249816413476683 8,
116090695300364456691050396301163933906 2,
246252990031076120253743264881256829498,
504373594039553499314895843732694711 02,
996666383431453022598297176238259009 8,
189841215891705337637770890712049335 2,
348229449268808607501236545093108248,
614522557533191660296299785458426 32,
104217275838962328354343238469557 68,
169656030435520069414047132392303 2,
264781087962950397351403038993768,
395649901553833927076809138726 32,
565214145076905610109727341037 6,
770746561468507650149628192324,
100153507987433197477070330076,
123785234591209569915480183 24,
145222900917061506606475328 76,
161358778796735007338614764,
169382143488285366819540 36,
167520801252150362788556 4,
155620416463746238656036,
13532210127282281622264,
1097206226536401212616,
82585414900589338584,
5741232051912627816,
366461620334848584,

21328454093562616,
1122550215450664,
52895036330136,
2203959847089,
79936367511,
2472258789,
63391251,
1293699,
19701,
199,
1]

PS1_4:

CODE:

```
import numpy as np

y = np.random.randint(1,101)
def Least_moves(x):
    m = x
    n = 0
    while x > 1:
        if x % 2:
            x = (x-1) / 2
            n += 2
        else:
            x /= 2
            n += 1
    print(" The smallest number of moves you have to make to get to exactly",m, "RMB is", n)
    return
Least_moves(y)
```

RESULTS:

```
In [217]: runfile('E:/Program/ESE5023_Assignments_12132601/PS1_4.py', wdir='E:/
Program/ESE5023_Assignments_12132601')
 The smallest number of moves you have to make to get to exactly 65 RMB is 7

In [218]: runfile('E:/Program/ESE5023_Assignments_12132601/PS1_4.py', wdir='E:/
Program/ESE5023_Assignments_12132601')
 The smallest number of moves you have to make to get to exactly 24 RMB is 5
```

PS1_5:

5.1

COODE:

```
import matplotlib.pyplot as plt

def Find_expression(x):
    a = ['','+','-']
    m = 0
    for i0 in range(3):
        for i1 in range(3):
            for i2 in range(3):
                for i3 in range(3):
                    for i4 in range(3):
                        for i5 in range(3):
                            for i6 in range(3):
                                for i7 in range(3):
                                    n = '1'+a[i0]+'2'+a[i1]+'3'+a[i2]+'4'+a[i3]+'5'+a[i4]+'6'+a[i5]+'7'+a[i6]+'8'+a[i7]+'9'
                                    #m.append(eval(n))
                                    if eval(n) == x:
                                        print(n, '=', x)
                                        m+=1
    return m
```

RESULT:

```
In [220]: Find_expression(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
```

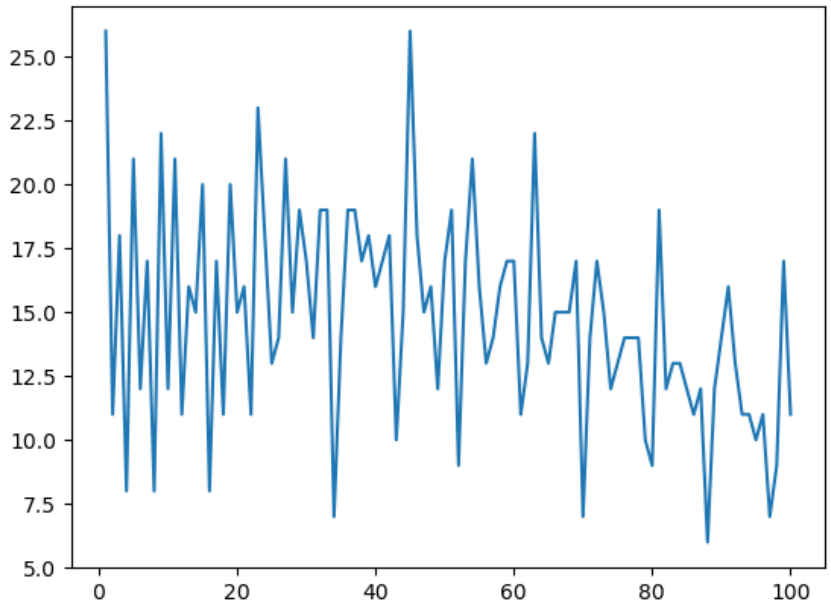
5.2

CODE:

```
Total_solutions = [0]*100
for i in range(1,101):
    Total_solutions[i-1]=Find_expression(i)
num = range(1,101)
plt.plot(num, Total_solutions)
plt.show()

Total_solutions_max = max(Total_solutions)
Total_solutions_min = min(Total_solutions)
for i in range(100):
    if Total_solutions[i] == Total_solutions_max:
        print(i,'yields the maximum of Total_solutions')
    elif Total_solutions[i] == Total_solutions_min:
        print(i,'yields the minimum of Total_solutions')
```

RESULT:




```
0 yields the maximum of Total_solutions
44 yields the maximum of Total_solutions
87 yields the minimum of Total_solutions
```