Assignment 2
Question 2

Q2. Perform the following operations on the specified matrices using any software

Software Used (Python)

```
#libraries import
import numpy as np
import cv2
```

a) Transpose of A and B.

```
     # Question 2a
     A_trans = np.transpose(A)
     B_trans = np.transpose(B)
     print("Transpose of A: " ,A_trans,'\n')
     print("Transpose of B: " ,B_trans)
```

Result:

Transpose of A:

```
[[ 63  99 111  40  58  89 160 145 149 164 192 174 147 107  18  52 161 179
  118  91]
 [134  95 198   8  12  80  81  85 167 197  31 140  53  41 107 115  54 145
  130  82]
 [ 79  61  90 104  43  82 151 146  50 154  74 157  99   8 184 124 136 140
  175  58]
 [ 35  92  96 175 198 159 158 146  93  16 132 127  20  86 139  78 100  25
  171 133]
 [152 131  26 132 190 126 106 106 162 134 141 176  87 195 107   2 193  82
   52  16]
 [ 31 163  88  36  77 101  28  74 169 175  65   9   8  43 173 112 126 173
  144  59]
 [146 177  77 169 171  58 196  27 121 148 138  91 113 155 110  69  13  78
    3  53]
 [ 72  76 138  66  61 136 100 162  75 124  20  20 136  20  45  76 120 122
   64 122]
 [  9  10   5 191 130  97  38  41  66 157 167  83  65 160  60  53  99 151
  172  60]
 [ 81   1 136  64  67   7  34  42 136 191  30 103  40 112  89  74  33 187
  132 174]
 [105   9 185   7  52 106 174  53 117   5  90 179 196 144 183 190  13 183
  153 193]
 [ 26 173  92 121 165  47 160  45  28 151 160 104  29  97   8 142 134 150
  189  64]
 [ 79  61  90 104  43  82 151 146  50 154  74 157  99   8 184 124 136 140
```

```
  175  58]
 [ 35  92  96 175 198 159 158 146  93  16 132 127  20  86 139  78 100  25
  171 133]
 [152 131  26 132 190 126 106 106 162 134 141 176  87 195 107   2 193  82
   52  16]
 [ 31 163  88  36  77 101  28  74 169 175  65   9   8  43 173 112 126 173
  144  59]
 [146 177  77 169 171  58 196  27 121 148 138  91 113 155 110  69  13  78
    3  53]
 [ 72  76 138  66  61 136 100 162  75 124  20  20 136  20  45  76 120 122
   64 122]
 [  8  10   5 191 130  97  38  41  66 157 167  83  65 160  60  53  99 151
  172  60]
 [ 81   1 136  64  67   7  34  42 136 191  30 103  40 112  89  74  33 187
  132 174]]
```

Transpose of B:

```
[[ 84  58 173 147 142 193  80  25 177  75 182   3  40  91  34 164  52 189
  135  62]
 [ 62  70   0 167 125 129  67   6   5 130 144  97 154 130  55  70  27 134
   93   8]
 [175 156  64  49  80  53  77 118  31  67  54  19 186 110 180 164  98  69
   94  13]
 [ 38  44  47 152 168  72 198 132 133 152 145  17  41 164 103 147 180 187
  182 146]
 [159  80 155 129 152 103 140  84   3 162   3 127  79 141 118  95  93  42
  139 115]
 [123 180 194 167   2  92  60  25 163  74  62  36 151 181  16 195  32 199
   46 135]
 [ 90  53   8  25 194 142 161 187 143 108  51 128  13 193  27   9 146  60
  122  29]
 [115  43  32   7 100  23  33  24 128  42  83  32 196  58  95 107  32  26
   52 138]
 [142 122 196 110   6 102 150 189 184  97  16   1  75  56 169 155 132  70
   74  85]
 [163   4 196 106  56  70  15 161 133 124  79  96 166  98  70  55 123  95
  112 133]
 [ 80  21  96 123  50 104 166 167 167 165  82 199 160 145 198  14  60 186
    9 104]
 [ 82  23  71  79 127  23 192 100  94 102 104 189 104 196   8 159 166 132
  188 118]
 [ 62  70   0 167 125 129  67   6   5 130 144  97 154 130  55  70  27 134
   93   8]
 [175 156  64  49  80  53  77 118  31  67  54  19 186 110 180 164  98  69
   94  13]
 [ 38  44  47 152 168  72 198 132 133 152 145  17  41 164 103 147 180 187
  182 146]
 [159  80 155 129 152 103 140  84   3 162   3 127  79 141 118  95  93  42
  139 115]
 [123 180 194 167   2  92  60  25 163  74  62  36 151 181  16 195  32 199
   46 135]
 [ 90  53   8  25 194 142 161 187 143 108  51 128  13 193  27   9 146  60
  122  29]
 [115  43  32   7 100  23  33  24 128  42  83  32 196  58  95 107  32  26
   52 138]
 [142 122 196 110   6 102 150 189 184  97  16   1  75  56 169 155 132  70
   74  85]]
```

b) Inverse of matrix A and B.

```
# Question 2b
A_inv = np.linalg.inv(A)
B_inv = np.linalg.inv(B)
print("Inverse of A: " ,A_inv,'\n')
print("Inverse of B: " ,B_inv)
```

Results:

Inverse of A:

```
[[-8.43223619e-17 -1.07793294e-03 -1.38935116e-02 -1.01739453e-02
   2.24401776e-03 -4.17321945e-03 -5.37644613e-03  6.59605477e-03
   5.97517851e-04 -2.62557619e-03  8.95466151e-03  1.52055653e-03
   2.82831706e-03 -2.96682936e-03 -1.02592586e-02 -3.61284346e-03
   2.49209147e-03  7.75394862e-03 -6.51041667e-03  5.38966468e-03]
 [-3.94574708e-16 -1.38828633e-03  1.14533623e-02  5.55314534e-03
  -3.81778742e-03  9.71800434e-03  1.04121475e-03 -1.66594360e-02
   1.04121475e-02  4.16485900e-03 -1.11062907e-02  2.08242950e-02
  -1.38828633e-03 -1.38828633e-02  1.04121475e-03  6.24728850e-03
   6.94143167e-04 -9.71800434e-03  0.00000000e+00  6.94143167e-03]
 [ 1.91499164e+00 -1.25299748e+13  7.10984006e+12  3.67248362e+12
  -2.48312622e+12  1.79503201e+13  4.91610858e+12 -2.57408166e+13
  -5.30288575e+12  4.21832958e+12  1.42485448e+13  6.73288094e+12
  -1.65835034e+13 -1.68089265e+13  4.68546201e+12  3.83680946e+12
   1.73379014e+13  6.61666883e+12 -2.65684624e+13  1.20825525e+13]
 [-2.64504313e+00 -9.68543090e+12  7.98741100e+12  7.20415473e+12
  -1.72057123e+13 -2.36837612e+13  1.58605257e+13  2.22788254e+13
   1.49909372e+13 -6.79365022e+12 -9.14093805e+12 -2.53297827e+13
  -5.13226655e+12  3.35051565e+13 -3.33426222e+12  1.24568912e+13
   7.60876730e+12 -1.27169881e+13  1.29299129e+13 -1.89894304e+13]
 [ 6.98079093e+00  4.43699262e+13 -9.40478642e+13  4.12789636e+13
```

```
  -2.14369608e+13  4.95830026e+13 -4.64762770e+12 -2.37452981e+13
   3.51638817e+13 -2.14489972e+13 -4.81307893e+13  6.97243629e+13
  -8.73014430e+12 -5.07894771e+13 -4.51138181e+13  2.61227529e+13
  -2.42860417e+13  5.01743728e+13 -1.95521388e+13  4.62354903e+13]
 [-9.86921566e+00 -3.89939441e+13  1.42887781e+14 -5.56000432e+13
   9.39351120e+12 -7.31100867e+13 -1.15125583e+13  2.82789716e+13
  -7.11615115e+13  3.32511600e+13  7.63955572e+13 -9.58345446e+13
   2.09414027e+13  8.38118347e+13  7.45792323e+13 -5.13843121e+13
   4.00933115e+13 -8.78807078e+13  3.32223025e+13 -4.69975033e+13]
 [-3.01612903e+00 -2.02472256e+13  3.53664621e+13  4.80992331e+12
   2.78122392e+12 -3.63193518e+13 -2.18504845e+12  1.96368326e+13
   9.07983796e+12 -7.83616028e+12  1.28376139e+13 -2.76873514e+13
   3.91940225e+12  1.86210722e+13  6.71272408e+12  1.13497974e+13
   1.40106526e+13 -1.49557885e+13  3.79825247e+12 -2.81500757e+13]
 [-1.00000000e+00  8.84188069e+12 -1.69386248e+13  6.14550871e+12
  -4.16255022e+12  2.57358195e+00  2.43041174e+12 -7.80410913e+12
   1.40737488e+13 -2.17982109e+12 -1.45179885e+13  4.69305394e+12
   1.13761572e+13 -5.99672490e+12 -9.05279975e+12  1.92185096e-01
  -8.19407447e+11 -5.07261878e+12  1.31043235e+13  6.59847475e+12]
 [ 1.00000000e+00 -1.89703131e+00  3.83271182e+00 -3.55894669e-01
   6.00672903e-01 -2.58305452e+00 -4.71439658e-01  1.51698090e+00
  -7.56880065e-01 -5.79737922e-01  2.50253536e+00 -2.75053436e+00
  -2.93693621e-01  1.23677423e+00  1.30118956e+00 -1.93245974e-01
   1.05963209e+00 -6.21607883e-01  8.37399221e-02 -2.19012104e+00]
 [ 4.77419355e+00  3.28796467e+13 -8.69541622e+13  7.14197930e+12
  -1.21039027e+13  5.44790278e+13  1.15994514e+12 -3.15652555e+13
   1.45277407e+13  9.63061737e+12 -3.15695287e+13  5.40481649e+13
  -2.42391290e+12 -3.95007231e+13 -2.91804048e+13  1.81596759e+13
  -1.84202826e+13  1.44093384e+13 -1.07004456e+13  5.58429552e+13]
 [-2.36029688e-16 -1.63771513e-03  4.54212470e-03 -8.96055002e-04
  -2.64868483e-04 -1.90875393e-03 -1.32444792e-04 -5.36611123e-04
  -8.66093252e-04 -2.68371588e-03  1.07593528e-03 -2.49385176e-03
   1.52809368e-03  3.50812243e-03  2.02023935e-03  7.62548513e-04
   1.86706669e-03 -2.60620831e-04  5.51594764e-04 -2.42286980e-03]
 [-1.64014811e-16  2.57701524e-03 -2.21358656e-03  4.79243950e-04
   5.19987366e-04 -3.87126477e-04  1.18380360e-03 -2.14885356e-03
  -2.67871522e-04 -7.99013130e-04 -3.74689206e-03  1.62305612e-03
  -3.87253219e-04  3.03317825e-04 -3.77012430e-03  3.10155175e-03
   1.72009721e-03  5.44815174e-04  1.63336337e-03  7.65767668e-04]
 [-1.91499164e+00  1.25299748e+13 -7.10984006e+12 -3.67248362e+12
   2.48312622e+12 -1.79503201e+13 -4.91610858e+12  2.57408166e+13
   5.30288575e+12 -4.21832958e+12 -1.42485448e+13 -6.73288094e+12
   1.65835034e+13  1.68089265e+13 -4.68546201e+12 -3.83680946e+12
  -1.73379014e+13 -6.61666883e+12  2.65684624e+13 -1.20825525e+13]
 [ 2.64504313e+00  9.68543090e+12 -7.98741100e+12 -7.20415473e+12
```

```
   1.72057123e+13  2.36837612e+13 -1.58605257e+13 -2.22788254e+13
  -1.49909372e+13  6.79365022e+12  9.14093805e+12  2.53297827e+13
   5.13226655e+12 -3.35051565e+13  3.33426222e+12 -1.24568912e+13
  -7.60876730e+12  1.27169881e+13 -1.29299129e+13  1.89894304e+13]
 [-6.98079093e+00 -4.43699262e+13  9.40478642e+13 -4.12789636e+13
   2.14369608e+13 -4.95830026e+13  4.64762770e+12  2.37452981e+13
  -3.51638817e+13  2.14489972e+13  4.81307893e+13 -6.97243629e+13
   8.73014430e+12  5.07894771e+13  4.51138181e+13 -2.61227529e+13
   2.42860417e+13 -5.01743728e+13  1.95521388e+13 -4.62354903e+13]
 [ 9.86921566e+00  3.89939441e+13 -1.42887781e+14  5.56000432e+13
  -9.39351120e+12  7.31100867e+13  1.15125583e+13 -2.82789716e+13
   7.11615115e+13 -3.32511600e+13 -7.63955572e+13  9.58345446e+13
  -2.09414027e+13 -8.38118347e+13 -7.45792323e+13  5.13843121e+13
  -4.00933115e+13  8.78807078e+13 -3.32223025e+13  4.69975033e+13]
 [ 3.01612903e+00  2.02472256e+13 -3.53664621e+13 -4.80992331e+12
  -2.78122392e+12  3.63193518e+13  2.18504845e+12 -1.96368326e+13
  -9.07983796e+12  7.83616028e+12 -1.28376139e+13  2.76873514e+13
  -3.91940225e+12 -1.86210722e+13 -6.71272408e+12 -1.13497974e+13
  -1.40106526e+13  1.49557885e+13 -3.79825247e+12  2.81500757e+13]
 [ 1.00000000e+00 -8.84188069e+12  1.69386248e+13 -6.14550871e+12
   4.16255022e+12 -2.58064516e+00 -2.43041174e+12  7.80410913e+12
  -1.40737488e+13  2.17982109e+12  1.45179885e+13 -4.69305394e+12
  -1.13761572e+13  5.99672490e+12  9.05279975e+12 -1.93548387e-01
   8.19407447e+11  5.07261878e+12 -1.31043235e+13 -6.59847475e+12]
 [-1.00000000e+00  1.88690857e+00 -3.80678854e+00  3.56925171e-01
  -6.00520932e-01  2.58064516e+00  4.71782505e-01 -1.51912588e+00
   7.54838710e-01  5.82119515e-01 -2.49083990e+00  2.74337090e+00
   2.92825391e-01 -1.23240587e+00 -1.29287364e+00  1.93548387e-01
  -1.05617097e+00  6.18348361e-01 -8.11888945e-02  2.17954219e+00]
 [-4.77419355e+00 -3.28796467e+13  8.69541622e+13 -7.14197930e+12
   1.21039027e+13 -5.44790278e+13 -1.15994514e+12  3.15652555e+13
  -1.45277407e+13 -9.63061737e+12  3.15695287e+13 -5.40481649e+13
   2.42391290e+12  3.95007231e+13  2.91804048e+13 -1.81596759e+13
   1.84202826e+13 -1.44093384e+13  1.07004456e+13 -5.58429552e+13]]
```

Inverse of B:

```
[[ 5.18134715e-03  1.29533679e-02  1.19818653e-02 -1.55440415e-02
   5.18134715e-03 -1.55440415e-02  6.21761658e-02  2.59067358e-03
  -1.03626943e-02 -5.18134715e-03  0.00000000e+00 -1.03626943e-02
  -1.29533679e-03  7.77202073e-03  0.00000000e+00 -4.14507772e-02
   0.00000000e+00  7.77202073e-03 -5.18134715e-03  5.18134715e-03]
 [-6.65535404e+13 -2.06923736e+13  3.98657375e+12  4.33501279e+13
   6.78128219e+13 -3.74017766e+13 -8.80849999e+13  2.90996108e+13
   3.71157616e+13  2.77088518e+13 -4.22973052e+13  4.11920624e+13
  -1.57699617e+13  1.23964394e+13  2.17430435e+13  9.19138028e+13
   0.00000000e+00 -2.81381203e+13 -4.28238252e+13 -4.26977894e+13]
 [-2.32372607e+13 -2.67491306e+13 -9.19217647e+12  1.30863963e+13
   1.33322429e+13  2.69186537e+13 -2.97786660e+13  3.74571464e+13
  -2.00226779e+13 -1.32944021e+13 -5.90230801e+12  1.24467754e+13
  -6.09620463e+12  1.24556625e+13  1.05635619e+12  5.62176793e+13
   0.00000000e+00 -1.41406009e+13 -3.94981041e+13  1.55624291e+13]
 [ 3.23271878e+13  2.24876733e+13 -2.34264396e+12 -2.89419675e+13
  -3.87337718e+13  1.25745674e+13  2.40052825e+13 -5.01245182e+13
  -6.39205637e+11  7.63519397e+12  2.19636543e+13 -7.74143439e+12
  -4.41028070e+12 -5.78612087e+12  4.58632591e+12 -5.01860157e+13
   3.51843721e+13  1.22442694e+13  1.91616298e+13  1.06331734e+13]
 [ 1.11590691e+13  4.56589608e+12 -1.56386816e+13 -1.73829313e+13
  -2.85447075e+13  1.89949439e+13 -3.39402243e+12 -1.21966729e+13
   6.78838864e+11  9.35572569e+12 -1.73188837e+12  2.10471999e+12
  -7.37040469e+12 -2.21190537e+12  6.23565332e+12 -7.23422425e+12
  -0.00000000e+00  8.63417804e+12  2.65067264e+13  1.24137101e+13]
 [-3.06735552e+13 -3.84532205e+13 -4.88340133e+12  1.63841908e+11
   4.65792436e+12  2.98651776e+13 -2.44242275e+13  1.64099709e+13
  -5.75101248e+12  4.51081693e+12 -2.99741570e+13 -1.79928510e+12
   9.88134747e+12  2.19526424e+13  5.47257357e+12  4.81153806e+13
  -0.00000000e+00 -8.07682408e+12 -1.44812361e+13 -1.64906146e+12]
 [ 1.44346142e+13  1.80922960e+13  8.44802772e+12 -1.44247661e+13
  -1.17626001e+13 -1.52693052e+13  2.06102283e+13 -2.18694864e+13
   7.89846087e+12  1.44346142e+13  1.36366044e+13 -9.33848217e+12
   1.88139406e+12 -4.81153806e+12 -2.72866127e+12 -3.36807664e+13
  -0.00000000e+00  4.69309707e+12  1.86773284e+13 -7.28444228e+12]
 [-2.25540847e+13 -3.30219799e+13 -5.49715945e+12  1.42828629e+13
  -1.89854784e+11  2.93352297e+13 -2.22346489e+13  2.34505365e+13
  -1.45991066e+13 -2.25540847e+13 -5.02750212e+12  3.20350567e+12
  -1.65450139e+12  3.09742763e+13  1.10576828e+13  4.08980735e+13
  -0.00000000e+00 -2.16914349e+13 -2.25377560e+13  1.13961485e+13]
 [-1.35324508e+12 -2.54205645e+13 -1.05629882e+13  1.08184816e+13
   4.07500422e+12  1.10690945e+13 -1.66971568e+13  1.16356514e+13
  -2.23303945e+12 -1.35324508e+12 -1.12905306e+13  3.30437765e+12
  -5.47977872e+12  1.21792057e+13  3.07116481e+12  3.24778819e+13
  -0.00000000e+00 -6.07208567e+12 -1.83649236e+13  6.00933434e+10]
 [-2.76226224e-03 -2.21720896e-03  7.72462436e-04  1.19054337e-03
   1.27568000e-03 -1.00491935e-03 -7.00880489e-03  3.64272450e-03
   1.07155572e-03  2.39954339e-03 -2.66861696e-03  7.24232928e-04
   1.06172070e-03  2.46729529e-04 -2.07992177e-04  2.06487076e-03
   0.00000000e+00  2.59973390e-04  2.01921942e-03 -1.12761263e-03]
 [ 1.56388013e-03 -3.69168851e-03  9.32935932e-04  5.86550168e-04
  -7.87589874e-04  1.85406676e-03  2.39630134e-03  1.11225611e-03
  -3.20053223e-03 -5.58757262e-03  3.45461346e-03  3.01756833e-04
  -1.09125617e-03  3.86189661e-03  2.81234549e-03 -5.35604199e-04
```

```
   0.00000000e+00  3.25206729e-04 -5.81485218e-03  2.15352871e-03]
 [-1.31147574e-03 -3.12626106e-03  2.04165050e-03 -1.59190382e-03
  -3.45817031e-03  1.96223156e-03  5.52318119e-03  4.06778002e-04
  -2.83284841e-03 -3.96898286e-03  2.48150828e-03  2.19679435e-04
   2.98804489e-03  2.17581382e-03 -1.98340865e-03  1.17575982e-03
  -0.00000000e+00 -2.14279800e-03  1.01588043e-03  7.01082226e-05]
 [ 6.65535404e+13  2.06923736e+13 -3.98657375e+12 -4.33501279e+13
  -6.78128219e+13  3.74017766e+13  8.80849999e+13 -2.90996108e+13
  -3.71157616e+13 -2.77088518e+13  4.22973052e+13 -4.11920624e+13
   1.57699617e+13 -1.23964394e+13 -2.17430435e+13 -9.19138028e+13
  -0.00000000e+00  2.81381203e+13  4.28238252e+13  4.26977894e+13]
 [ 2.32372607e+13  2.67491306e+13  9.19217647e+12 -1.30863963e+13
  -1.33322429e+13 -2.69186537e+13  2.97786660e+13 -3.74571464e+13
   2.00226779e+13  1.32944021e+13  5.90230801e+12 -1.24467754e+13
   6.09620463e+12 -1.24556625e+13 -1.05635619e+12 -5.62176793e+13
  -0.00000000e+00  1.41406009e+13  3.94981041e+13 -1.55624291e+13]
 [-3.23271878e+13 -2.24876733e+13  2.34264396e+12  2.89419675e+13
   3.87337718e+13 -1.25745674e+13 -2.40052825e+13  5.01245182e+13
   6.39205637e+11 -7.63519397e+12 -2.19636543e+13  7.74143439e+12
   4.41028070e+12  5.78612087e+12 -4.58632591e+12  5.01860157e+13
  -3.51843721e+13 -1.22442694e+13 -1.91616298e+13 -1.06331734e+13]
 [-1.11590691e+13 -4.56589608e+12  1.56386816e+13  1.73829313e+13
   2.85447075e+13 -1.89949439e+13  3.39402243e+12  1.21966729e+13
  -6.78838864e+11 -9.35572569e+12  1.73188837e+12 -2.10471999e+12
   7.37040469e+12  2.21190537e+12 -6.23565332e+12  7.23422425e+12
  -0.00000000e+00 -8.63417804e+12 -2.65067264e+13 -1.24137101e+13]
 [ 3.06735552e+13  3.84532205e+13  4.88340133e+12 -1.63841908e+11
  -4.65792436e+12 -2.98651776e+13  2.44242275e+13 -1.64099709e+13
   5.75101248e+12 -4.51081693e+12  2.99741570e+13  1.79928510e+12
  -9.88134747e+12 -2.19526424e+13 -5.47257357e+12 -4.81153806e+13
   0.00000000e+00  8.07682408e+12  1.44812361e+13  1.64906146e+12]
 [-1.44346142e+13 -1.80922960e+13 -8.44802772e+12  1.44247661e+13
   1.17626001e+13  1.52693052e+13 -2.06102283e+13  2.18694864e+13
  -7.89846087e+12 -1.44346142e+13 -1.36366044e+13  9.33848217e+12
  -1.88139406e+12  4.81153806e+12  2.72866127e+12  3.36807664e+13
  -0.00000000e+00 -4.69309707e+12 -1.86773284e+13  7.28444228e+12]
 [ 2.25540847e+13  3.30219799e+13  5.49715945e+12 -1.42828629e+13
   1.89854784e+11 -2.93352297e+13  2.22346489e+13 -2.34505365e+13
   1.45991066e+13  2.25540847e+13  5.02750212e+12 -3.20350567e+12
   1.65450139e+12 -3.09742763e+13 -1.10576828e+13 -4.08980735e+13
   0.00000000e+00  2.16914349e+13  2.25377560e+13 -1.13961485e+13]
 [ 1.35324508e+12  2.54205645e+13  1.05629882e+13 -1.08184816e+13
  -4.07500422e+12 -1.10690945e+13  1.66971568e+13 -1.16356514e+13
   2.23303945e+12  1.35324508e+12  1.12905306e+13 -3.30437765e+12
   5.47977872e+12 -1.21792057e+13 -3.07116481e+12 -3.24778819e+13
  -0.00000000e+00  6.07208567e+12  1.83649236e+13 -6.00933434e+10]]
```

c) Addition (A+B) and (B+A). Comment on the result.

```
# Question 2c
add_AB = np.array(A + B)
add_BA = np.array(B + A)
print("A+B : " ,add_AB,'\n')
print("B+A: " ,add_BA,'\n')
if np.array_equal(add_AB,add_BA):
    print("Both addition are same")
else:
    print("Both addition are different")
```

Results:

A+B :

```
[[147 196 254  73 311 154 236 187 151 244 185 108 141 210 190 190 269 162
  123 223]
 [157 165 217 136 211 343 230 119 132   5  30 196 131 248 175 243 357 129
   53 123]
 [284 198 154 143 181 282  85 170 201 332 281 163  90 160  73 243 271 146
   37 332]
 [187 175 153 327 261 203 194  73 301 170 130 200 271 224 284 165 336  91
  198 174]
 [200 137 123 366 342  79 365 161 136 123 102 292 168 278 358 229 173 255
  230  73]
 [282 209 135 231 229 193 200 159 199  77 210  70 211 212 198 204 150 278
  120 109]
 [240 148 228 356 246  88 357 133 188  49 340 352 218 235 304 168 256 261
   71 184]
 [170  91 264 278 190  99 214 186 230 203 220 145 152 264 238 158  52 349
   65 231]
 [326 172  81 226 165 332 264 203 250 269 284 122  55 124 295 172 284 218
  194 320]
 [239 327 221 168 296 249 256 166 254 315 170 253 284  83 286 337 222 232
  199 288]
 [374 175 128 277 144 127 189 103 183 109 172 264 218 186 286  68 200  71
  250  46]
 [177 237 176 144 303  45 219  52  84 199 378 293 254 146 193 136 127 148
  115 104]
 [187 207 285  61 166 159 126 332 140 206 356 133 253 206 128  87 264 149
  261 115]
 [198 171 118 250 336 224 348  78 216 210 289 293 138 196 359 184 336 213
  218 168]
 [ 52 162 364 242 225 189 137 140 229 159 381  16 239 319 210 291 126  72
  155 258]
 [216 185 288 225  97 307  78 183 208 129 204 301 194 242 149 207 264  85
  160 229]
 [213  81 234 280 286 158 159 152 231 156  73 300 163 198 373 219  45 266
  131 165]
 [368 279 209 212 124 372 138 148 221 282 369 282 274  94 269 215 277 182
  177 257]
 [253 223 269 353 191 190 125 116 246 244 162 377 268 265 234 283  49 186
  224 206]
 [153  90  71 279 131 194  82 260 145 307 297 182  66 146 162 174 188 151
  198 259]]
```

B+A:

```
[[147 196 254  73 311 154 236 187 151 244 185 108 141 210 190 190 269 162
  123 223]
 [157 165 217 136 211 343 230 119 132   5  30 196 131 248 175 243 357 129
   53 123]
 [284 198 154 143 181 282  85 170 201 332 281 163  90 160  73 243 271 146
   37 332]
 [187 175 153 327 261 203 194  73 301 170 130 200 271 224 284 165 336  91
  198 174]
 [200 137 123 366 342  79 365 161 136 123 102 292 168 278 358 229 173 255
  230  73]
 [282 209 135 231 229 193 200 159 199  77 210  70 211 212 198 204 150 278
  120 109]
 [240 148 228 356 246  88 357 133 188  49 340 352 218 235 304 168 256 261
   71 184]
 [170  91 264 278 190  99 214 186 230 203 220 145 152 264 238 158  52 349
   65 231]
 [326 172  81 226 165 332 264 203 250 269 284 122  55 124 295 172 284 218
  194 320]
 [239 327 221 168 296 249 256 166 254 315 170 253 284  83 286 337 222 232
  199 288]
 [374 175 128 277 144 127 189 103 183 109 172 264 218 186 286  68 200  71
  250  46]
 [177 237 176 144 303  45 219  52  84 199 378 293 254 146 193 136 127 148
  115 104]
 [187 207 285  61 166 159 126 332 140 206 356 133 253 206 128  87 264 149
  261 115]
 [198 171 118 250 336 224 348  78 216 210 289 293 138 196 359 184 336 213
  218 168]
 [ 52 162 364 242 225 189 137 140 229 159 381  16 239 319 210 291 126  72
  155 258]
 [216 185 288 225  97 307  78 183 208 129 204 301 194 242 149 207 264  85
```

```
  160 229]
 [213  81 234 280 286 158 159 152 231 156  73 300 163 198 373 219  45 266
  131 165]
 [368 279 209 212 124 372 138 148 221 282 369 282 274  94 269 215 277 182
  177 257]
 [253 223 269 353 191 190 125 116 246 244 162 377 268 265 234 283  49 186
  224 206]
 [153  90  71 279 131 194  82 260 145 307 297 182  66 146 162 174 188 151
  198 259]]
```

```
Both addition are same
```

Comment: Order of Addition doesn’t matter in matrix addition.

d) Subtraction (A-B) and (B-A). Comment on the result.

```
# Question 2d
sub_AB = np.array(A - B)
sub_BA = np.array(B - A)
print("A-B : " ,sub_AB,'\n')
print("B-A: " ,sub_BA,'\n')
if np.array_equal(sub_AB,sub_BA):
    print("Both Subtraction are same")
else:
    print("Both Subtraction are different")
```

Results:

A-B :

```
[[ -21   72  -96   -3   -7  -92   56  -43 -133  -82   25  -56   17
-140
   114 -128   23  -18 -107  -61]
 [  41   25  -95   48   51  -17  124   33 -112   -3  -12  150   -9
-64
    87   83   -3   23  -33 -121]
 [ -62  198   26   49 -129 -106   69  106 -191  -60   89   21   90
32
   -21  -67 -117  130  -27  -60]
 [-107 -159   55   23    3 -131  144   59   81  -42 -116   42  -63
126
   -20  -93    2   41  184  -46]
 [ -84 -113  -37   30   38   75  -23  -39  124   11    2   38  -82
118
    22  -75  169 -133   30   61]
 [-104  -49   29   87   23    9  -84  113   -5  -63    2   24  -47
106
    54   -2  -34   -6   74  -95]
 [  80   14   74  -40  -34  -32   35   67 -112   19    8  -32   84
81
   -92 -112  136  -61    5 -116]
 [ 120   79   28   14   22   49 -160  138 -148 -119 -114  -55  140
28
   -26  -10    2  -25   17 -147]
 [ -28  162   19  -40  159    6  -22  -53 -118    3  -50  -66   45
62
    29  166  -42  -68  -62  -48]
 [  89   67   87 -136  -28  101   40   82   60   67 -160   49   24
-51
   -18   13   74   16  115   94]
 [  10 -113   20  -13  138    3   87  -63  151  -49    8   56  -70
78
    -4   62   76  -31   84   14]
 [ 171   43  138  110   49  -27  -37  -12   82    7  -20  -85   60
108
   159 -118   55 -108   51  102]
 [ 107 -101  -87  -21    8 -143  100  -60  -10 -126   36  -75  -55
-166
    46  -71  -38  123 -131  -35]
 [  16  -89 -102  -78   54 -138  -38  -38  104   14   -1  -99 -122
-24
    31  -98  -26 -173  102   56]
 [ -16   52    4   36  -11  157   83  -50 -109   19  -15    0  129
-41
     4   55   94   18  -35  -80]
 [-112   45  -40  -69  -93  -83   60  -31 -102   19  176  -17   54
-86
  -145   17 -126   67  -54  -81]
 [ 109   27   38  -80  100   94 -133   88  -33  -90  -47  -32  109
2
    13   33  -19  -26   67  -99]
 [ -10   11   71 -162   40  -26   18   96   81   92   -3   18    6
-44
  -105  131 -121   62  125  117]
 [ -17   37   81  -11  -87   98 -119   12   98   20  144    1   82
77
  -130    5  -43  -58  120   58]
 [  29   74   45  -13  -99  -76   24  -16  -25   41   89  -54   50
120
  -130  -56  -82   93  -78   89]]
```

B-A:

```
[[  21  -72   96    3    7   92  -56   43  133   82  -25   56  -17
140
  -114  128  -23   18  107   61]
 [ -41  -25   95  -48  -51   17 -124  -33  112    3   12 -150    9
64
   -87  -83    3  -23   33  121]
 [  62 -198  -26  -49  129  106  -69 -106  191   60  -89  -21  -90
-32
    21   67  117 -130   27   60]
 [ 107  159  -55  -23   -3  131 -144  -59  -81   42  116  -42   63
-126
    20   93   -2  -41 -184   46]
 [  84  113   37  -30  -38  -75   23   39 -124  -11   -2  -38   82
-118
   -22   75 -169  133  -30  -61]
 [ 104   49  -29  -87  -23   -9   84 -113    5   63   -2  -24   47
-106
   -54    2   34    6  -74   95]
 [ -80  -14  -74   40   34   32  -35  -67  112  -19   -8   32  -84
-81
    92  112 -136   61   -5  116]
 [-120  -79  -28  -14  -22  -49  160 -138  148  119  114   55 -140
-28
    26   10   -2   25  -17  147]
 [  28 -162  -19   40 -159   -6   22   53  118   -3   50   66  -45
-62
   -29 -166   42   68   62   48]
 [ -89  -67  -87  136   28 -101  -40  -82  -60  -67  160  -49  -24
51
    18  -13  -74  -16 -115  -94]
 [ -10  113  -20   13 -138   -3  -87   63 -151   49   -8  -56   70
-78
     4  -62  -76   31  -84  -14]
 [-171  -43 -138 -110  -49   27   37   12  -82   -7   20   85  -60
-108
  -159  118  -55  108  -51 -102]
 [-107  101   87   21   -8  143 -100   60   10  126  -36   75   55
166
   -46   71   38 -123  131   35]
 [ -16   89  102   78  -54  138   38   38 -104  -14    1   99  122
24
   -31   98   26  173 -102  -56]
 [  16  -52   -4  -36   11 -157  -83   50  109  -19   15    0 -129
41
    -4  -55  -94  -18   35   80]
 [ 112  -45   40   69   93   83  -60   31  102  -19 -176   17  -54
86
   145  -17  126  -67   54   81]
 [-109  -27  -38   80 -100  -94  133  -88   33   90   47   32 -109
-2
   -13  -33   19   26  -67   99]
 [  10  -11  -71  162  -40   26  -18  -96  -81  -92    3  -18   -6
44
   105 -131  121  -62 -125 -117]
 [  17  -37  -81   11   87  -98  119  -12  -98  -20 -144   -1  -82
-77
   130   -5   43   58 -120  -58]
 [ -29  -74  -45   13   99   76  -24   16   25  -41  -89   54  -50
-120
   130   56   82  -93   78  -89]]

Both Subtraction are different
```

Comment: Order of subtraction matter in Matrix Subtraction

e) Multiplication (A*B) and (B*A). Comment on the result.

```
# Question 2e
mul_AB = A.dot(B)
mul_BA = B.dot(A)
print("A*B : \n" , mul_AB,'\n')
print("B*A : \n" , mul_BA,'\n')
if np.array_equal(mul_AB,mul_BA):
    print("Both multiplication are equal")
else:
    print("Both multiplication are different")
```

Results:

A*B :

```
 [[143678 122789 156180 194897 166134 138707 144983 105749 165596 141851
```

```
  174372 162836 122789 156180 194897 166134 138707 144983 105749 165596]
 [175004 151996 181744 212424 204611 173870 187347 105661 194944 158156
  205933 211300 151996 181744 212424 204611 173870 187347 105661 194944]
 [192652 166686 176457 221124 190863 221722 161229 120133 193960 190903
  215696 202872 166686 176457 221124 190863 221722 161229 120133 193960]
 [210139 167376 177794 277225 229285 204265 222379 135089 222153 216662
  250853 261514 167376 177794 277225 229285 204265 222379 135089 222153]
 [218104 189480 189248 292795 245909 198285 238505 136141 218012 207256
  263351 270034 189480 189248 292795 245909 198285 238505 136141 218012]
 [213073 168535 183397 234745 189337 206024 181215 119998 198644 186718
  213686 203722 168535 183397 234745 189337 206024 181215 119998 198644]
 [212491 191838 210050 263758 238002 223311 212916 141153 225605 229956
  266631 264164 191838 210050 263758 238002 223311 212916 141153 225605]
 [194589 161621 189188 208931 196744 218484 160434 124032 201017 203761
  220012 189392 161621 189188 208931 196744 218484 160434 124032 201017]
 [236455 185714 219387 276133 239657 227832 206251 153903 239530 202515
  230456 229722 185714 219387 276133 239657 227832 206251 153903 239530]
 [270457 199754 260423 313312 297523 286837 251381 196070 309488 279902
  301541 296374 199754 260423 313312 297523 286837 251381 196070 309488]
 [208695 167747 187754 245189 224192 191544 210970 142830 208634 206286
  227899 243910 167747 187754 245189 224192 191544 210970 142830 208634]
 [209781 185522 209368 241727 236981 217282 185742 164151 212500 223574
  241113 226842 185522 209368 241727 236981 217282 185742 164151 212500]
 [164976 123851 153946 195685 149299 148599 147609 111302 169075 170936
  183686 174300 123851 153946 195685 149299 148599 147609 111302 169075]
 [199712 158586 170270 267287 211679 157088 210336 140188 194771 183172
  219440 232117 158586 170270 267287 211679 157088 210336 140188 194771]
 [248978 193842 205623 260420 221190 247878 174562 152678 229023 212931
  226837 227157 193842 205623 260420 221190 247878 174562 152678 229023]
 [183402 149850 147056 186703 162541 196007 139741 110676 166566 173611
  188429 190871 149850 147056 186703 162541 196007 139741 110676 166566]
 [213947 171368 206999 220335 224206 212173 185889 148332 215699 215300
  237693 209081 171368 206999 220335 224206 212173 185889 148332 215699]
 [278300 205338 234812 294289 262351 278136 224677 190844 273673 273401
  283084 279276 205338 234812 294289 262351 278136 224677 190844 273673]
 [270238 216848 213043 265732 253047 294499 204087 175897 242994 265633
  271893 268643 216848 213043 265732 253047 294499 204087 175897 242994]
 [191267 160390 146474 230232 179659 204848 159371 116382 175529 192035
  211396 203617 160390 146474 230232 179659 204848 159371 116382 175529]]

B*A :
 [[260333 229796 220354 238006 251726 212044 223783 201727 202178 211798
  261973 237576 220354 238006 251726 212044 223783 201727 202094 211798]
 [178480 151016 144525 177437 187867 149495 151839 152482 133068 132246
  192530 152020 144525 177437 187867 149495 151839 152482 133010 132246]
 [214225 198584 183855 206216 237063 191354 190166 173948 176024 182416
  201284 185696 183855 206216 237063 191354 190166 173948 175851 182416]
 [217243 182666 204091 218971 254785 192478 232536 177013 180240 151947
  215970 199665 204091 218971 254785 192478 232536 177013 180093 151947]
 [218488 192740 234727 220935 226734 176906 242981 166682 180634 171094
  255028 226516 234727 220935 226734 176906 242981 166682 180492 171094]
 [208070 173979 180882 179811 212081 166413 213375 163309 149966 149148
  214580 187042 180882 179811 212081 166413 213375 163309 149773 149148]
 [247190 203441 236452 261030 258515 194384 255880 170418 207899 196201
  286266 229774 236452 261030 258515 194384 255880 170418 207819 196201]
 [242854 207015 207378 215993 222148 192239 232347 163751 199284 214729
  255088 213910 207378 215993 222148 192239 232347 163751 199259 214729]
 [264466 219577 242331 241643 256978 209343 210206 194238 207010 200532
  242182 206841 242331 241643 256978 209343 210206 194238 206833 200532]
 [237149 194687 222959 232703 247624 198839 248870 174849 200851 177465
  244464 231573 222959 232703 247624 198839 248870 174849 200776 177465]
 [172656 149990 176261 159298 192001 134370 178157 131290 134507 118060
  168706 147443 176261 159298 192001 134370 178157 131290 134325 118060]
 [191035 129805 158383 149742 169217 119265 165830 107793 140127 108819
  169518 184213 158383 149742 169217 119265 165830 107793 140124 108819]
 [279765 223563 232478 244732 263764 212544 213376 206767 212013 186579
  243954 242451 232478 244732 263764 212544 213376 206767 211973 186579]
 [304102 240018 286429 287017 313978 234017 285669 221307 242918 205555
  306049 283556 286429 287017 313978 234017 285669 221307 242827 205555]
 [203452 174347 165276 210686 198189 171361 198933 147800 176066 177707
  225296 187653 165276 210686 198189 171361 198933 147800 176032 177707]
 [238796 223795 229114 259828 275526 203922 217137 198807 194480 196471
  262952 201992 229114 259828 275526 203922 217137 198807 194316 196471]
 [210118 193260 208837 210820 216690 174865 218492 151285 181285 192701
  239114 193220 208837 210820 216690 174865 218492 151285 181233 192701]
 [243367 194695 229869 233910 279463 193597 239984 181499 199394 157213
  228710 207016 229869 233910 279463 193597 239984 181499 199205 157213]
 [214823 204908 228030 222065 240117 181760 238682 166551 181984 187321
  250086 209239 228030 222065 240117 181760 238682 166551 181849 187321]
 [192413 165418 206756 217086 209720 174277 163957 150038 181082 149975
  180340 182356 206756 217086 209720 174277 163957 150038 181020 149975]]

Both multiplication are different
```

Comment: Order of multiplication gives different results.

f) Multiply with a scalar to both matrices A and B. Comment on the result.

1. **C > 1**
2. **C < 1**

```
c1 = 2
c2 = 0.5

# Question 2f
c1_A = c1*A
c2_A = c2*A
c1_B = c1*B
c2_B = c2*B
print("c1*A with c1 = 2 : \n" , c1_A,'\n')
print("c2*A with c2 = 0.5 : \n" , c2_A,'\n')
print("c1*B with c1 = 2 : \n" , c1_B,'\n')
print("c2*B with c2 = 0.5 : \n" , c2_B,'\n')
```

Results:

c1*A with c1 = 2 :

```
 [[126 268 158  70 304  62 292 144  18 162 210  52 158  70 304  62 292 144
   16 162]
 [198 190 122 184 262 326 354 152  20   2  18 346 122 184 262 326 354 152
   20   2]
 [222 396 180 192  52 176 154 276  10 272 370 184 180 192  52 176 154 276
   10 272]
 [ 80  16 208 350 264  72 338 132 382 128  14 242 208 350 264  72 338 132
  382 128]
 [116  24  86 396 380 154 342 122 260 134 104 330  86 396 380 154 342 122
  260 134]
 [178 160 164 318 252 202 116 272 194  14 212  94 164 318 252 202 116 272
  194  14]
 [320 162 302 316 212  56 392 200  76  68 348 320 302 316 212  56 392 200
   76  68]
 [290 170 292 292 212 148  54 324  82  84 106  90 292 292 212 148  54 324
   82  84]
 [298 334 100 186 324 338 242 150 132 272 234  56 100 186 324 338 242 150
  132 272]
 [328 394 308  32 268 350 296 248 314 382  10 302 308  32 268 350 296 248
  314 382]
 [384  62 148 264 282 130 276  40 334  60 180 320 148 264 282 130 276  40
  334  60]
 [348 280 314 254 352  18 182  40 166 206 358 208 314 254 352  18 182  40
  166 206]
 [294 106 198  40 174  16 226 272 130  80 392  58 198  40 174  16 226 272
  130  80]
 [214  82  16 172 390  86 310  40 320 224 288 194  16 172 390  86 310  40
  320 224]
 [ 36 214 368 278 214 346 220  90 120 178 366  16 368 278 214 346 220  90
```

```
  120 178]
 [104 230 248 156   4 224 138 152 106 148 380 284 248 156   4 224 138 152
  106 148]
 [322 108 272 200 386 252  26 240 198  66  26 268 272 200 386 252  26 240
  198  66]
 [358 290 280  50 164 346 156 244 302 374 366 300 280  50 164 346 156 244
  302 374]
 [236 260 350 342 104 288   6 128 344 264 306 378 350 342 104 288   6 128
  344 264]
 [182 164 116 266  32 118 106 244 120 348 386 128 116 266  32 118 106 244
  120 348]]
```

c2*A with c2 = 0.5 :

```
 [[31.5 67.  39.5 17.5 76.  15.5 73.  36.   4.5 40.5 52.5 13.  39.5 17.5
  76.  15.5 73.  36.   4.  40.5]
 [49.5 47.5 30.5 46.  65.5 81.5 88.5 38.   5.   0.5  4.5 86.5 30.5 46.
  65.5 81.5 88.5 38.   5.   0.5]
 [55.5 99.  45.  48.  13.  44.  38.5 69.   2.5 68.  92.5 46.  45.  48.
  13.  44.  38.5 69.   2.5 68. ]
 [20.   4.  52.  87.5 66.  18.  84.5 33.  95.5 32.   3.5 60.5 52.  87.5
  66.  18.  84.5 33.  95.5 32. ]
 [29.   6.  21.5 99.  95.  38.5 85.5 30.5 65.  33.5 26.  82.5 21.5 99.
  95.  38.5 85.5 30.5 65.  33.5]
 [44.5 40.  41.  79.5 63.  50.5 29.  68.  48.5  3.5 53.  23.5 41.  79.5
  63.  50.5 29.  68.  48.5  3.5]
 [80.  40.5 75.5 79.  53.  14.  98.  50.  19.  17.  87.  80.  75.5 79.
  53.  14.  98.  50.  19.  17. ]
 [72.5 42.5 73.  73.  53.  37.  13.5 81.  20.5 21.  26.5 22.5 73.  73.
  53.  37.  13.5 81.  20.5 21. ]
 [74.5 83.5 25.  46.5 81.  84.5 60.5 37.5 33.  68.  58.5 14.  25.  46.5
  81.  84.5 60.5 37.5 33.  68. ]
 [82.  98.5 77.   8.  67.  87.5 74.  62.  78.5 95.5  2.5 75.5 77.   8.
  67.  87.5 74.  62.  78.5 95.5]
 [96.  15.5 37.  66.  70.5 32.5 69.  10.  83.5 15.  45.  80.  37.  66.
  70.5 32.5 69.  10.  83.5 15. ]
 [87.  70.  78.5 63.5 88.   4.5 45.5 10.  41.5 51.5 89.5 52.  78.5 63.5
  88.   4.5 45.5 10.  41.5 51.5]
 [73.5 26.5 49.5 10.  43.5  4.  56.5 68.  32.5 20.  98.  14.5 49.5 10.
  43.5  4.  56.5 68.  32.5 20. ]
 [53.5 20.5  4.  43.  97.5 21.5 77.5 10.  80.  56.  72.  48.5  4.  43.
  97.5 21.5 77.5 10.  80.  56. ]
 [ 9.  53.5 92.  69.5 53.5 86.5 55.  22.5 30.  44.5 91.5  4.  92.  69.5
  53.5 86.5 55.  22.5 30.  44.5]
 [26.  57.5 62.  39.   1.  56.  34.5 38.  26.5 37.  95.  71.  62.  39.
   1.  56.  34.5 38.  26.5 37. ]
 [80.5 27.  68.  50.  96.5 63.   6.5 60.  49.5 16.5  6.5 67.  68.  50.
  96.5 63.   6.5 60.  49.5 16.5]
 [89.5 72.5 70.  12.5 41.  86.5 39.  61.  75.5 93.5 91.5 75.  70.  12.5
  41.  86.5 39.  61.  75.5 93.5]
 [59.  65.  87.5 85.5 26.  72.   1.5 32.  86.  66.  76.5 94.5 87.5 85.5
  26.  72.   1.5 32.  86.  66. ]
 [45.5 41.  29.  66.5  8.  29.5 26.5 61.  30.  87.  96.5 32.  29.  66.5
   8.  29.5 26.5 61.  30.  87. ]]
```

c1*B with c1 = 2 :

```
 [[168 124 350  76 318 246 180 230 284 326 160 164 124 350  76 318 246 180
  230 284]
 [116 140 312  88 160 360 106  86 244   8  42  46 140 312  88 160 360 106
   86 244]
 [346   0 128  94 310 388  16  64 392 392 192 142   0 128  94 310 388  16
   64 392]
 [294 334  98 304 258 334  50  14 220 212 246 158 334  98 304 258 334  50
   14 220]
 [284 250 160 336 304   4 388 200  12 112 100 254 250 160 336 304   4 388
  200  12]
 [386 258 106 144 206 184 284  46 204 140 208  46 258 106 144 206 184 284
   46 204]
 [160 134 154 396 280 120 322  66 300  30 332 384 134 154 396 280 120 322
   66 300]
 [ 50  12 236 264 168  50 374  48 378 322 334 200  12 236 264 168  50 374
   48 378]
 [354  10  62 266   6 326 286 256 368 266 334 188  10  62 266   6 326 286
  256 368]
 [150 260 134 304 324 148 216  84 194 248 330 204 260 134 304 324 148 216
   84 194]
 [364 288 108 290   6 124 102 166  32 158 164 208 288 108 290   6 124 102
  166  32]
 [  6 194  38  34 254  72 256  64   2 192 398 378 194  38  34 254  72 256
   64   2]
 [ 80 308 372  82 158 302  26 392 150 332 320 208 308 372  82 158 302  26
  392 150]
 [182 260 220 328 282 362 386 116 112 196 290 392 260 220 328 282 362 386
  116 112]
 [ 68 110 360 206 236  32  54 190 338 140 396  16 110 360 206 236  32  54
  190 338]
 [328 140 328 294 190 390  18 214 310 110  28 318 140 328 294 190 390  18
  214 310]
 [104  54 196 360 186  64 292  64 264 246 120 332  54 196 360 186  64 292
   64 264]
 [378 268 138 374  84 398 120  52 140 190 372 264 268 138 374  84 398 120
```

```
   52 140]
 [270 186 188 364 278  92 244 104 148 224  18 376 186 188 364 278  92 244
  104 148]
 [124  16  26 292 230 270  58 276 170 266 208 236  16  26 292 230 270  58
  276 170]]
```

c2*B with c2 = 0.5 :

```
 [[42.  31.  87.5 19.  79.5 61.5 45.  57.5 71.  81.5 40.  41.  31.  87.5
  19.  79.5 61.5 45.  57.5 71. ]
 [29.  35.  78.  22.  40.  90.  26.5 21.5 61.   2.  10.5 11.5 35.  78.
  22.  40.  90.  26.5 21.5 61. ]
 [86.5  0.  32.  23.5 77.5 97.   4.  16.  98.  98.  48.  35.5  0.  32.
  23.5 77.5 97.   4.  16.  98. ]
 [73.5 83.5 24.5 76.  64.5 83.5 12.5  3.5 55.  53.  61.5 39.5 83.5 24.5
  76.  64.5 83.5 12.5  3.5 55. ]
 [71.  62.5 40.  84.  76.   1.  97.  50.   3.  28.  25.  63.5 62.5 40.
  84.  76.   1.  97.  50.   3. ]
 [96.5 64.5 26.5 36.  51.5 46.  71.  11.5 51.  35.  52.  11.5 64.5 26.5
  36.  51.5 46.  71.  11.5 51. ]
 [40.  33.5 38.5 99.  70.  30.  80.5 16.5 75.   7.5 83.  96.  33.5 38.5
  99.  70.  30.  80.5 16.5 75. ]
 [12.5  3.  59.  66.  42.  12.5 93.5 12.  94.5 80.5 83.5 50.   3.  59.
  66.  42.  12.5 93.5 12.  94.5]
 [88.5  2.5 15.5 66.5  1.5 81.5 71.5 64.  92.  66.5 83.5 47.   2.5 15.5
  66.5  1.5 81.5 71.5 64.  92. ]
 [37.5 65.  33.5 76.  81.  37.  54.  21.  48.5 62.  82.5 51.  65.  33.5
  76.  81.  37.  54.  21.  48.5]
 [91.  72.  27.  72.5  1.5 31.  25.5 41.5  8.  39.5 41.  52.  72.  27.
  72.5  1.5 31.  25.5 41.5  8. ]
 [ 1.5 48.5  9.5  8.5 63.5 18.  64.  16.   0.5 48.  99.5 94.5 48.5  9.5
   8.5 63.5 18.  64.  16.   0.5]
 [20.  77.  93.  20.5 39.5 75.5  6.5 98.  37.5 83.  80.  52.  77.  93.
  20.5 39.5 75.5  6.5 98.  37.5]
 [45.5 65.  55.  82.  70.5 90.5 96.5 29.  28.  49.  72.5 98.  65.  55.
  82.  70.5 90.5 96.5 29.  28. ]
 [17.  27.5 90.  51.5 59.   8.  13.5 47.5 84.5 35.  99.   4.  27.5 90.
  51.5 59.   8.  13.5 47.5 84.5]
 [82.  35.  82.  73.5 47.5 97.5  4.5 53.5 77.5 27.5  7.  79.5 35.  82.
  73.5 47.5 97.5  4.5 53.5 77.5]
 [26.  13.5 49.  90.  46.5 16.  73.  16.  66.  61.5 30.  83.  13.5 49.
  90.  46.5 16.  73.  16.  66. ]
 [94.5 67.  34.5 93.5 21.  99.5 30.  13.  35.  47.5 93.  66.  67.  34.5
  93.5 21.  99.5 30.  13.  35. ]
 [67.5 46.5 47.  91.  69.5 23.  61.  26.  37.  56.   4.5 94.  46.5 47.
  91.  69.5 23.  61.  26.  37. ]
 [31.   4.   6.5 73.  57.5 67.5 14.5 69.  42.5 66.5 52.  59.   4.   6.5
  73.  57.5 67.5 14.5 69.  42.5]]
```

Comment: Multiplication by C>1 increase the value while C<1 decrease the value of the matrices

g) Divide with a scalar to both matrices A and B. Comment on the result.

1. **C > 1**
2. **C < 1**

```
c1 = 2
c2 = 0.5
# Question 2g
div_c1_A = A/c1
div_c2_A = A/c2
div_c1_B = B/c1
div_c2_B = B/c2
print("A/c1 with c1 = 2 : \n" , div_c1_A,'\n')
print("A/c2 with c2 = 0.5 : \n" , div_c2_A,'\n')
print("B/c1 with c1 = 2 : \n" , div_c1_B,'\n')
print("B/c2 with c2 = 0.5 : \n" , div_c2_B,'\n')
```

Results:

A/c1 with c1 = 2 :

```
 [[31.5 67.  39.5 17.5 76.  15.5 73.  36.   4.5 40.5 52.5 13.  39.5 17.5
  76.  15.5 73.  36.   4.  40.5]
 [49.5 47.5 30.5 46.  65.5 81.5 88.5 38.   5.   0.5  4.5 86.5 30.5 46.
  65.5 81.5 88.5 38.   5.   0.5]
 [55.5 99.  45.  48.  13.  44.  38.5 69.   2.5 68.  92.5 46.  45.  48.
  13.  44.  38.5 69.   2.5 68. ]
 [20.   4.  52.  87.5 66.  18.  84.5 33.  95.5 32.   3.5 60.5 52.  87.5
  66.  18.  84.5 33.  95.5 32. ]
 [29.   6.  21.5 99.  95.  38.5 85.5 30.5 65.  33.5 26.  82.5 21.5 99.
  95.  38.5 85.5 30.5 65.  33.5]
 [44.5 40.  41.  79.5 63.  50.5 29.  68.  48.5  3.5 53.  23.5 41.  79.5
  63.  50.5 29.  68.  48.5  3.5]
 [80.  40.5 75.5 79.  53.  14.  98.  50.  19.  17.  87.  80.  75.5 79.
  53.  14.  98.  50.  19.  17. ]
 [72.5 42.5 73.  73.  53.  37.  13.5 81.  20.5 21.  26.5 22.5 73.  73.
  53.  37.  13.5 81.  20.5 21. ]
 [74.5 83.5 25.  46.5 81.  84.5 60.5 37.5 33.  68.  58.5 14.  25.  46.5
  81.  84.5 60.5 37.5 33.  68. ]
 [82.  98.5 77.   8.  67.  87.5 74.  62.  78.5 95.5  2.5 75.5 77.   8.
  67.  87.5 74.  62.  78.5 95.5]
 [96.  15.5 37.  66.  70.5 32.5 69.  10.  83.5 15.  45.  80.  37.  66.
  70.5 32.5 69.  10.  83.5 15. ]
 [87.  70.  78.5 63.5 88.   4.5 45.5 10.  41.5 51.5 89.5 52.  78.5 63.5
  88.   4.5 45.5 10.  41.5 51.5]
 [73.5 26.5 49.5 10.  43.5  4.  56.5 68.  32.5 20.  98.  14.5 49.5 10.
  43.5  4.  56.5 68.  32.5 20. ]
 [53.5 20.5  4.  43.  97.5 21.5 77.5 10.  80.  56.  72.  48.5  4.  43.
  97.5 21.5 77.5 10.  80.  56. ]
 [ 9.  53.5 92.  69.5 53.5 86.5 55.  22.5 30.  44.5 91.5  4.  92.  69.5
  53.5 86.5 55.  22.5 30.  44.5]
 [26.  57.5 62.  39.   1.  56.  34.5 38.  26.5 37.  95.  71.  62.  39.
   1.  56.  34.5 38.  26.5 37. ]
 [80.5 27.  68.  50.  96.5 63.   6.5 60.  49.5 16.5  6.5 67.  68.  50.
  96.5 63.   6.5 60.  49.5 16.5]
 [89.5 72.5 70.  12.5 41.  86.5 39.  61.  75.5 93.5 91.5 75.  70.  12.5
  41.  86.5 39.  61.  75.5 93.5]
 [59.  65.  87.5 85.5 26.  72.   1.5 32.  86.  66.  76.5 94.5 87.5 85.5
  26.  72.   1.5 32.  86.  66. ]
 [45.5 41.  29.  66.5  8.  29.5 26.5 61.  30.  87.  96.5 32.  29.  66.5
   8.  29.5 26.5 61.  30.  87. ]]
```

A/c2 with c2 = 0.5 :

```
 [[126. 268. 158.  70. 304.  62. 292. 144.  18. 162. 210.  52. 158.  70.
  304.  62. 292. 144.  16. 162.]
 [198. 190. 122. 184. 262. 326. 354. 152.  20.   2.  18. 346. 122. 184.
  262. 326. 354. 152.  20.   2.]
 [222. 396. 180. 192.  52. 176. 154. 276.  10. 272. 370. 184. 180. 192.
   52. 176. 154. 276.  10. 272.]
 [ 80.  16. 208. 350. 264.  72. 338. 132. 382. 128.  14. 242. 208. 350.
  264.  72. 338. 132. 382. 128.]
 [116.  24.  86. 396. 380. 154. 342. 122. 260. 134. 104. 330.  86. 396.
  380. 154. 342. 122. 260. 134.]
 [178. 160. 164. 318. 252. 202. 116. 272. 194.  14. 212.  94. 164. 318.
  252. 202. 116. 272. 194.  14.]
 [320. 162. 302. 316. 212.  56. 392. 200.  76.  68. 348. 320. 302. 316.
  212.  56. 392. 200.  76.  68.]
 [290. 170. 292. 292. 212. 148.  54. 324.  82.  84. 106.  90. 292. 292.
  212. 148.  54. 324.  82.  84.]
 [298. 334. 100. 186. 324. 338. 242. 150. 132. 272. 234.  56. 100. 186.
  324. 338. 242. 150. 132. 272.]
 [328. 394. 308.  32. 268. 350. 296. 248. 314. 382.  10. 302. 308.  32.
  268. 350. 296. 248. 314. 382.]
 [384.  62. 148. 264. 282. 130. 276.  40. 334.  60. 180. 320. 148. 264.
  282. 130. 276.  40. 334.  60.]
 [348. 280. 314. 254. 352.  18. 182.  40. 166. 206. 358. 208. 314. 254.
```

```
  352.  18. 182.  40. 166. 206.]
 [294. 106. 198.  40. 174.  16. 226. 272. 130.  80. 392.  58. 198.  40.
  174.  16. 226. 272. 130.  80.]
 [214.  82.  16. 172. 390.  86. 310.  40. 320. 224. 288. 194.  16. 172.
  390.  86. 310.  40. 320. 224.]
 [ 36. 214. 368. 278. 214. 346. 220.  90. 120. 178. 366.  16. 368. 278.
  214. 346. 220.  90. 120. 178.]
 [104. 230. 248. 156.   4. 224. 138. 152. 106. 148. 380. 284. 248. 156.
    4. 224. 138. 152. 106. 148.]
 [322. 108. 272. 200. 386. 252.  26. 240. 198.  66.  26. 268. 272. 200.
  386. 252.  26. 240. 198.  66.]
 [358. 290. 280.  50. 164. 346. 156. 244. 302. 374. 366. 300. 280.  50.
  164. 346. 156. 244. 302. 374.]
 [236. 260. 350. 342. 104. 288.   6. 128. 344. 264. 306. 378. 350. 342.
  104. 288.   6. 128. 344. 264.]
 [182. 164. 116. 266.  32. 118. 106. 244. 120. 348. 386. 128. 116. 266.
   32. 118. 106. 244. 120. 348.]]
```

B/c1 with c1 = 2 :

```
 [[42.   31.   87.5 19.   79.5 61.5 45.   57.5 71.   81.5 40.   41.   31.   87.5
  19.   79.5 61.5 45.   57.5 71. ]
 [29.   35.   78.   22.   40.   90.   26.5 21.5 61.    2.   10.5 11.5 35.   78.
  22.   40.   90.   26.5 21.5 61. ]
 [86.5  0.   32.   23.5 77.5 97.    4.   16.   98.   98.   48.   35.5  0.   32.
  23.5 77.5 97.    4.   16.   98. ]
 [73.5 83.5 24.5 76.   64.5 83.5 12.5  3.5 55.   53.   61.5 39.5 83.5 24.5
  76.   64.5 83.5 12.5  3.5 55. ]
 [71.   62.5 40.   84.   76.    1.   97.   50.    3.   28.   25.   63.5 62.5 40.
  84.   76.    1.   97.   50.    3. ]
 [96.5 64.5 26.5 36.   51.5 46.   71.   11.5 51.   35.   52.   11.5 64.5 26.5
  36.   51.5 46.   71.   11.5 51. ]
 [40.   33.5 38.5 99.   70.   30.   80.5 16.5 75.    7.5 83.   96.   33.5 38.5
  99.   70.   30.   80.5 16.5 75. ]
 [12.5  3.   59.   66.   42.   12.5 93.5 12.   94.5 80.5 83.5 50.    3.   59.
  66.   42.   12.5 93.5 12.   94.5]
 [88.5  2.5 15.5 66.5  1.5 81.5 71.5 64.   92.   66.5 83.5 47.    2.5 15.5
  66.5  1.5 81.5 71.5 64.   92. ]
 [37.5 65.   33.5 76.   81.   37.   54.   21.   48.5 62.   82.5 51.   65.   33.5
  76.   81.   37.   54.   21.   48.5]
 [91.   72.   27.   72.5  1.5 31.   25.5 41.5  8.   39.5 41.   52.   72.   27.
  72.5  1.5 31.   25.5 41.5  8. ]
 [ 1.5 48.5  9.5  8.5 63.5 18.   64.   16.    0.5 48.   99.5 94.5 48.5  9.5
   8.5 63.5 18.   64.   16.    0.5]
 [20.   77.   93.   20.5 39.5 75.5  6.5 98.   37.5 83.   80.   52.   77.   93.
  20.5 39.5 75.5  6.5 98.   37.5]
 [45.5 65.  55.  82.  70.5 90.5 96.5 29.  28.  49.  72.5 98.  65.  55.
  82.  70.5 90.5 96.5 29.  28. ]
 [17.  27.5 90.  51.5 59.   8.  13.5 47.5 84.5 35.  99.   4.  27.5 90.
  51.5 59.   8.  13.5 47.5 84.5]
 [82.  35.  82.  73.5 47.5 97.5  4.5 53.5 77.5 27.5  7.  79.5 35.  82.
  73.5 47.5 97.5  4.5 53.5 77.5]
 [26.  13.5 49.  90.  46.5 16.  73.  16.  66.  61.5 30.  83.  13.5 49.
  90.  46.5 16.  73.  16.  66. ]
 [94.5 67.  34.5 93.5 21.  99.5 30.  13.  35.  47.5 93.  66.  67.  34.5
  93.5 21.  99.5 30.  13.  35. ]
 [67.5 46.5 47.  91.  69.5 23.  61.  26.  37.  56.   4.5 94.  46.5 47.
  91.  69.5 23.  61.  26.  37. ]
 [31.   4.   6.5 73.  57.5 67.5 14.5 69.  42.5 66.5 52.  59.   4.   6.5
  73.  57.5 67.5 14.5 69.  42.5]]
```

B/c2 with c2 = 0.5 :

```
 [[168. 124. 350.  76. 318. 246. 180. 230. 284. 326. 160. 164. 124. 350.
    76. 318. 246. 180. 230. 284.]
 [116. 140. 312.  88. 160. 360. 106.  86. 244.   8.  42.  46. 140. 312.
    88. 160. 360. 106.  86. 244.]
 [346.   0. 128.  94. 310. 388.  16.  64. 392. 392. 192. 142.   0. 128.
    94. 310. 388.  16.  64. 392.]
 [294. 334.  98. 304. 258. 334.  50.  14. 220. 212. 246. 158. 334.  98.
   304. 258. 334.  50.  14. 220.]
 [284. 250. 160. 336. 304.   4. 388. 200.  12. 112. 100. 254. 250. 160.
   336. 304.   4. 388. 200.  12.]
 [386. 258. 106. 144. 206. 184. 284.  46. 204. 140. 208.  46. 258. 106.
   144. 206. 184. 284.  46. 204.]
 [160. 134. 154. 396. 280. 120. 322.  66. 300.  30. 332. 384. 134. 154.
   396. 280. 120. 322.  66. 300.]
 [ 50.  12. 236. 264. 168.  50. 374.  48. 378. 322. 334. 200.  12. 236.
   264. 168.  50. 374.  48. 378.]
 [354.  10.  62. 266.   6. 326. 286. 256. 368. 266. 334. 188.  10.  62.
   266.   6. 326. 286. 256. 368.]
 [150. 260. 134. 304. 324. 148. 216.  84. 194. 248. 330. 204. 260. 134.
   304. 324. 148. 216.  84. 194.]
 [364. 288. 108. 290.   6. 124. 102. 166.  32. 158. 164. 208. 288. 108.
   290.   6. 124. 102. 166.  32.]
 [  6. 194.  38.  34. 254.  72. 256.  64.   2. 192. 398. 378. 194.  38.
    34. 254.  72. 256.  64.   2.]
 [ 80. 308. 372.  82. 158. 302.  26. 392. 150. 332. 320. 208. 308. 372.
    82. 158. 302.  26. 392. 150.]
 [182. 260. 220. 328. 282. 362. 386. 116. 112. 196. 290. 392. 260. 220.
   328. 282. 362. 386. 116. 112.]
 [ 68. 110. 360. 206. 236.  32.  54. 190. 338. 140. 396.  16. 110. 360.
```

```
  206. 236.  32.  54. 190. 338.]
 [328. 140. 328. 294. 190. 390.  18. 214. 310. 110.  28. 318. 140. 328.
  294. 190. 390.  18. 214. 310.]
 [104.  54. 196. 360. 186.  64. 292.  64. 264. 246. 120. 332.  54. 196.
  360. 186.  64. 292.  64. 264.]
 [378. 268. 138. 374.  84. 398. 120.  52. 140. 190. 372. 264. 268. 138.
  374.  84. 398. 120.  52. 140.]
 [270. 186. 188. 364. 278.  92. 244. 104. 148. 224.  18. 376. 186. 188.
  364. 278.  92. 244. 104. 148.]
 [124.  16.  26. 292. 230. 270.  58. 276. 170. 266. 208. 236.  16.  26.
  292. 230. 270.  58. 276. 170.]]
```

Comment: Division by C<1 increase the value while C>1 decrease the value of the matrices

h) Element by element multiplication (A.*B) and (B.*A). Comment on the result.

```
# Question 2h
ele_AB = np.multiply(A,B)
ele_BA = np.multiply(B,A)
print("A.*B : " ,ele_AB,'\n')
print("B.*A: " ,ele_BA,'\n')
if np.array_equal(ele_AB,ele_BA):
    print("Both multiplication are same")
else:
    print("Both multiplication are different")
```

Results:

A.*B :

```
[[ 5292  8308 13825  1330 24168  3813 13140  8280  1278 13203  8400  2132
   4898  6125  5776  4929 17958  6480   920 11502]
 [ 5742  6650  9516  4048 10480 29340  9381  3268  1220     4   189  3979
   4270 14352  5764 13040 31860  4028   430   122]
 [19203     0  5760  4512  4030 17072   616  4416   980 26656 17760  6532
      0  6144  1222 13640 14938  1104   160 26656]
 [ 5880  1336  5096 26600 17028  6012  4225   462 21010  6784   861  9559
  17368  8575 20064  4644 28223  1650  1337  7040]
 [ 8236  1500  3440 33264 28880   154 33174  6100   780  3752  2600 20955
   5375 15840 31920 11704   342 11834 13000   402]
 [17177 10320  4346 11448 12978  9292  8236  3128  9894   490 11024  1081
  10578  8427  9072 10403  5336 19312  2231   714]
 [12800  5427 11627 31284 14840  1680 31556  3300  5700   510 28884 30720
  10117 12166 20988  3920 11760 16100  1254  5100]
 [ 3625   510 17228 19272  8904  1850  5049  3888  7749  6762  8851  4500
```

```
    876 17228 13992  6216   675 30294   984  7938]
 [26373   835  1550 12369   486 27547 17303  9600 12144 18088 19539  2632
    250  2883 21546   507 19723 10725  8448 25024]
 [12300 25610 10318  2432 21708 12950 15984  5208 15229 23684   825 15402
  20020  1072 20368 28350 10952 13392  6594 18527]
 [34944  4464  3996 19140   423  4030  7038  1660  2672  2370  7380 16640
  10656  7128 20445   195  8556  1020 13861   480]
 [  522 13580  2983  2159 22352   324 11648   640    83  9888 35621 19656
  15229  2413  2992  1143  3276  2560  2656   103]
 [ 5880  8162 18414   820  6873  1208  1469 26656  4875  6640 31360  3016
  15246  3720  3567   632 17063  1768 12740  3000]
 [ 9737  5330   880 14104 27495  7783 29915  1160  8960 10976 20880 19012
   1040  9460 31980  6063 28055  3860  9280  6272]
 [  612  5885 33120 14317 12626  2768  2970  4275 10140  6230 36234    64
  10120 25020 11021 20414  1760  1215  5700 15041]
 [ 8528  8050 20336 11466   190 21840   621  8132  8215  4070  2660 22578
   8680 12792   294 10640 13455   684  5671 11470]
 [ 8372  1458 13328 18000 17949  4032  1898  3840 13068  4059   780 22244
   3672  9800 34740 11718   416 17520  3168  4356]
 [33831 19430  9660  4675  3444 34427  4680  3172 10570 17765 34038 19800
  18760  1725 15334  7266 15522  7320  3926 13090]
 [15930 12090 16450 31122  7228  6624   366  3328 12728 14784  1377 35532
  16275 16074  9464 20016   138  7808  8944  9768]
 [ 5642   656   754 19418  1840  7965  1537 16836  5100 23142 20072  7552
    464  1729  2336  6785  7155  3538  8280 14790]]
```

B.*A:

```
[[ 5292  8308 13825  1330 24168  3813 13140  8280  1278 13203  8400  2132
   4898  6125  5776  4929 17958  6480   920 11502]
 [ 5742  6650  9516  4048 10480 29340  9381  3268  1220     4   189  3979
   4270 14352  5764 13040 31860  4028   430   122]
 [19203     0  5760  4512  4030 17072   616  4416   980 26656 17760  6532
      0  6144  1222 13640 14938  1104   160 26656]
 [ 5880  1336  5096 26600 17028  6012  4225   462 21010  6784   861  9559
  17368  8575 20064  4644 28223  1650  1337  7040]
 [ 8236  1500  3440 33264 28880   154 33174  6100   780  3752  2600 20955
   5375 15840 31920 11704   342 11834 13000   402]
 [17177 10320  4346 11448 12978  9292  8236  3128  9894   490 11024  1081
  10578  8427  9072 10403  5336 19312  2231   714]
 [12800  5427 11627 31284 14840  1680 31556  3300  5700   510 28884 30720
  10117 12166 20988  3920 11760 16100  1254  5100]
 [ 3625   510 17228 19272  8904  1850  5049  3888  7749  6762  8851  4500
    876 17228 13992  6216   675 30294   984  7938]
 [26373   835  1550 12369   486 27547 17303  9600 12144 18088 19539  2632
    250  2883 21546   507 19723 10725  8448 25024]
 [12300 25610 10318  2432 21708 12950 15984  5208 15229 23684   825 15402
  20020  1072 20368 28350 10952 13392  6594 18527]
 [34944  4464  3996 19140   423  4030  7038  1660  2672  2370  7380 16640
  10656  7128 20445   195  8556  1020 13861   480]
 [  522 13580  2983  2159 22352   324 11648   640    83  9888 35621 19656
  15229  2413  2992  1143  3276  2560  2656   103]
 [ 5880  8162 18414   820  6873  1208  1469 26656  4875  6640 31360  3016
  15246  3720  3567   632 17063  1768 12740  3000]
 [ 9737  5330   880 14104 27495  7783 29915  1160  8960 10976 20880 19012
   1040  9460 31980  6063 28055  3860  9280  6272]
 [  612  5885 33120 14317 12626  2768  2970  4275 10140  6230 36234    64
  10120 25020 11021 20414  1760  1215  5700 15041]
 [ 8528  8050 20336 11466   190 21840   621  8132  8215  4070  2660 22578
   8680 12792   294 10640 13455   684  5671 11470]
 [ 8372  1458 13328 18000 17949  4032  1898  3840 13068  4059   780 22244
   3672  9800 34740 11718   416 17520  3168  4356]
 [33831 19430  9660  4675  3444 34427  4680  3172 10570 17765 34038 19800
  18760  1725 15334  7266 15522  7320  3926 13090]
 [15930 12090 16450 31122  7228  6624   366  3328 12728 14784  1377 35532
  16275 16074  9464 20016   138  7808  8944  9768]
 [ 5642   656   754 19418  1840  7965  1537 16836  5100 23142 20072  7552
    464  1729  2336  6785  7155  3538  8280 14790]]
```

Comment: Both multiplication are same

i) Find out the location(s) of a specific value....X (Value of X you can select) on both A and B.

```
# Question 2i
f_ele = 3
print("locations of 3 in A are: \n",np.argwhere(A==f_ele),'\n')
print("locations of 3 in B are: \n",np.argwhere(B==f_ele))
```

Results:

locations of 3 in A are:

```
 [[18  6]
 [18 16]]
```

locations of 3 in B are:

```
 [[ 8  4]
 [ 8 15]
 [10  4]
 [10 15]
 [11  0]]
```

j) Find the specific value of X (only first occurrence) using the scan and search mechanism and amplify the value by a factor of 2. (Value of X you can select)

```
# Question 2j
f_ele = 3

loc_A = np.argwhere(A==f_ele)
loc_B = np.argwhere(B==f_ele)

first_loc_A = loc_A[0]
first_loc_B = loc_B[0]

print("First location of 3 in A is : ",first_loc_A, '\n')

print("First location of 3 in B is : ",first_loc_B,'\n')

x_A = first_loc_A[0];
y_A = first_loc_A[1];

x_B = first_loc_B[0];
y_B = first_loc_B[1];

new_A = np.copy(A)
new_B = np.copy(B)

new_A[x_A][y_A]*=2
new_B[x_B][y_B]*=2

print("Modified Matrix A is: \n" ,new_A,'\n')
print("Modified Matrix B is: \n" ,new_B,'\n')
```

Results:

First location of 3 in A is : `[18  6]`

First location of 3 in B is : `[8 4]`

Modified Matrix A is:

```
 [[ 63 134  79  35 152  31 146  72   9  81 105  26  79  35 152  31 146  72
    8  81]
 [ 99  95  61  92 131 163 177  76  10   1   9 173  61  92 131 163 177  76
   10   1]
 [111 198  90  96  26  88  77 138   5 136 185  92  90  96  26  88  77 138
    5 136]
 [ 40   8 104 175 132  36 169  66 191  64   7 121 104 175 132  36 169  66
  191  64]
 [ 58  12  43 198 190  77 171  61 130  67  52 165  43 198 190  77 171  61
  130  67]
 [ 89  80  82 159 126 101  58 136  97   7 106  47  82 159 126 101  58 136
   97   7]
 [160  81 151 158 106  28 196 100  38  34 174 160 151 158 106  28 196 100
   38  34]
 [145  85 146 146 106  74  27 162  41  42  53  45 146 146 106  74  27 162
   41  42]
 [149 167  50  93 162 169 121  75  66 136 117  28  50  93 162 169 121  75
   66 136]
 [164 197 154  16 134 175 148 124 157 191   5 151 154  16 134 175 148 124
  157 191]
 [192  31  74 132 141  65 138  20 167  30  90 160  74 132 141  65 138  20
  167  30]
 [174 140 157 127 176   9  91  20  83 103 179 104 157 127 176   9  91  20
   83 103]
 [147  53  99  20  87   8 113 136  65  40 196  29  99  20  87   8 113 136
   65  40]
 [107  41   8  86 195  43 155  20 160 112 144  97   8  86 195  43 155  20
  160 112]
 [ 18 107 184 139 107 173 110  45  60  89 183   8 184 139 107 173 110  45
   60  89]
 [ 52 115 124  78   2 112  69  76  53  74 190 142 124  78   2 112  69  76
   53  74]
 [161  54 136 100 193 126  13 120  99  33  13 134 136 100 193 126  13 120
   99  33]
 [179 145 140  25  82 173  78 122 151 187 183 150 140  25  82 173  78 122
  151 187]
 [118 130 175 171  52 144   6  64 172 132 153 189 175 171  52 144   3  64
  172 132]
 [ 91  82  58 133  16  59  53 122  60 174 193  64  58 133  16  59  53 122
   60 174]]
```

Modified Matrix B is:

```
 [[ 84  62 175  38 159 123  90 115 142 163  80  82  62 175  38 159 123  90
  115 142]
 [ 58  70 156  44  80 180  53  43 122   4  21  23  70 156  44  80 180  53
   43 122]
 [173   0  64  47 155 194   8  32 196 196  96  71   0  64  47 155 194   8
   32 196]
 [147 167  49 152 129 167  25   7 110 106 123  79 167  49 152 129 167  25
    7 110]
 [142 125  80 168 152   2 194 100   6  56  50 127 125  80 168 152   2 194
```

```
  100   6]
 [193 129  53  72 103  92 142  23 102  70 104  23 129  53  72 103  92 142
   23 102]
 [ 80  67  77 198 140  60 161  33 150  15 166 192  67  77 198 140  60 161
   33 150]
 [ 25   6 118 132  84  25 187  24 189 161 167 100   6 118 132  84  25 187
   24 189]
 [177   5  31 133   6 163 143 128 184 133 167  94   5  31 133   3 163 143
  128 184]
 [ 75 130  67 152 162  74 108  42  97 124 165 102 130  67 152 162  74 108
   42  97]
 [182 144  54 145   3  62  51  83  16  79  82 104 144  54 145   3  62  51
   83  16]
 [  3  97  19  17 127  36 128  32   1  96 199 189  97  19  17 127  36 128
   32   1]
 [ 40 154 186  41  79 151  13 196  75 166 160 104 154 186  41  79 151  13
  196  75]
 [ 91 130 110 164 141 181 193  58  56  98 145 196 130 110 164 141 181 193
   58  56]
 [ 34  55 180 103 118  16  27  95 169  70 198   8  55 180 103 118  16  27
   95 169]
 [164  70 164 147  95 195   9 107 155  55  14 159  70 164 147  95 195   9
  107 155]
 [ 52  27  98 180  93  32 146  32 132 123  60 166  27  98 180  93  32 146
   32 132]
 [189 134  69 187  42 199  60  26  70  95 186 132 134  69 187  42 199  60
   26  70]
 [135  93  94 182 139  46 122  52  74 112   9 188  93  94 182 139  46 122
   52  74]
 [ 62   8  13 146 115 135  29 138  85 133 104 118   8  13 146 115 135  29
  138  85]]
```

k) Search by scan and search mechanism (may be available multiple times) in the given matrices and replace those values with your birth date. (Value of X you can select). Give the count value i.e. occurrence value. (Do it for both A and B)

```
# Question 2k

# MY BIRTHDAY
my_db = 20

f_ele = 13

loc_A = np.argwhere(A==f_ele)
```

```
loc_B = np.argwhere(B==f_ele)

print("Total locations of 13 in A are" , len(loc_A) , "\n which
are as follow: \n",np.argwhere(A==f_ele),'\n')
print("Total locations of 13 in B are" , len(loc_B) , "\n which
are as follow: \n",np.argwhere(B==f_ele))

new_A = np.copy(A)
new_B = np.copy(B)

new_A = np.where(new_A==f_ele, my_db, new_A)
new_B = np.where(new_B==f_ele, my_db, new_B)

print("Modified Matrix A is: \n" ,new_A,'\n')
print("Modified Matrix B is: \n" ,new_B,'\n')
```

Results:

```
Total locations of 13 in A are 3
 which are as follow:
 [[16  6]
 [16 10]
 [16 16]]

Total locations of 13 in B are 4
 which are as follow:
 [[12  6]
 [12 17]
 [19  2]
 [19 13]]

Modified Matrix A is:
 [[ 63 134  79  35 152  31 146  72   9  81 105  26  79  35 152  31 146  72
    8  81]
 [ 99  95  61  92 131 163 177  76  10   1   9 173  61  92 131 163 177  76
   10   1]
 [111 198  90  96  26  88  77 138   5 136 185  92  90  96  26  88  77 138
    5 136]
 [ 40   8 104 175 132  36 169  66 191  64   7 121 104 175 132  36 169  66
  191  64]
 [ 58  12  43 198 190  77 171  61 130  67  52 165  43 198 190  77 171  61
  130  67]
 [ 89  80  82 159 126 101  58 136  97   7 106  47  82 159 126 101  58 136
   97   7]
 [160  81 151 158 106  28 196 100  38  34 174 160 151 158 106  28 196 100
```

```
   38  34]
 [145  85 146 146 106  74  27 162  41  42  53  45 146 146 106  74  27 162
   41  42]
 [149 167  50  93 162 169 121  75  66 136 117  28  50  93 162 169 121  75
   66 136]
 [164 197 154  16 134 175 148 124 157 191   5 151 154  16 134 175 148 124
  157 191]
 [192  31  74 132 141  65 138  20 167  30  90 160  74 132 141  65 138  20
  167  30]
 [174 140 157 127 176   9  91  20  83 103 179 104 157 127 176   9  91  20
   83 103]
 [147  53  99  20  87   8 113 136  65  40 196  29  99  20  87   8 113 136
   65  40]
 [107  41   8  86 195  43 155  20 160 112 144  97   8  86 195  43 155  20
  160 112]
 [ 18 107 184 139 107 173 110  45  60  89 183   8 184 139 107 173 110  45
   60  89]
 [ 52 115 124  78   2 112  69  76  53  74 190 142 124  78   2 112  69  76
   53  74]
 [161  54 136 100 193 126  20 120  99  33  20 134 136 100 193 126  20 120
   99  33]
 [179 145 140  25  82 173  78 122 151 187 183 150 140  25  82 173  78 122
  151 187]
 [118 130 175 171  52 144   3  64 172 132 153 189 175 171  52 144   3  64
  172 132]
 [ 91  82  58 133  16  59  53 122  60 174 193  64  58 133  16  59  53 122
   60 174]]
```

Modified Matrix B is:

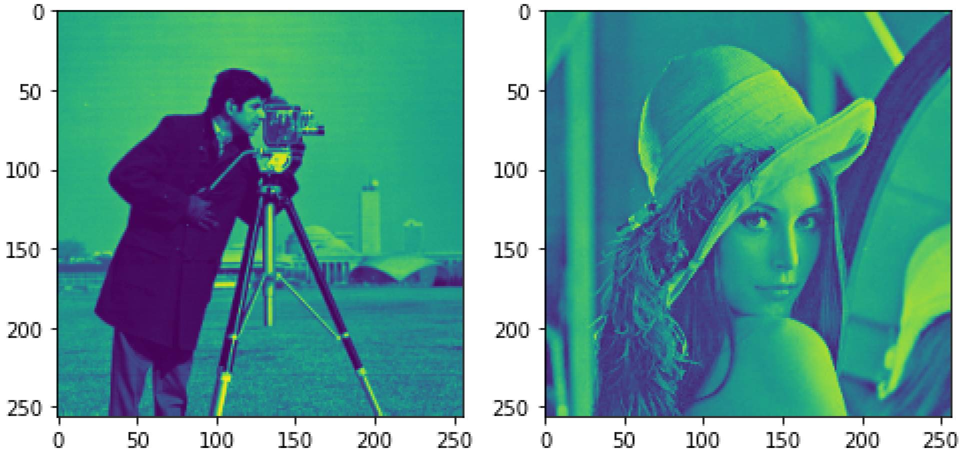
```
 [[ 84  62 175  38 159 123  90 115 142 163  80  82  62 175  38 159 123  90
  115 142]
 [ 58  70 156  44  80 180  53  43 122   4  21  23  70 156  44  80 180  53
   43 122]
 [173   0  64  47 155 194   8  32 196 196  96  71   0  64  47 155 194   8
   32 196]
 [147 167  49 152 129 167  25   7 110 106 123  79 167  49 152 129 167  25
    7 110]
 [142 125  80 168 152   2 194 100   6  56  50 127 125  80 168 152   2 194
  100   6]
 [193 129  53  72 103  92 142  23 102  70 104  23 129  53  72 103  92 142
   23 102]
 [ 80  67  77 198 140  60 161  33 150  15 166 192  67  77 198 140  60 161
   33 150]
 [ 25   6 118 132  84  25 187  24 189 161 167 100   6 118 132  84  25 187
   24 189]
 [177   5  31 133   3 163 143 128 184 133 167  94   5  31 133   3 163 143
  128 184]
 [ 75 130  67 152 162  74 108  42  97 124 165 102 130  67 152 162  74 108
   42  97]
 [182 144  54 145   3  62  51  83  16  79  82 104 144  54 145   3  62  51
   83  16]
 [  3  97  19  17 127  36 128  32   1  96 199 189  97  19  17 127  36 128
   32   1]
 [ 40 154 186  41  79 151  20 196  75 166 160 104 154 186  41  79 151  20
  196  75]
 [ 91 130 110 164 141 181 193  58  56  98 145 196 130 110 164 141 181 193
   58  56]
 [ 34  55 180 103 118  16  27  95 169  70 198   8  55 180 103 118  16  27
   95 169]
 [164  70 164 147  95 195   9 107 155  55  14 159  70 164 147  95 195   9
  107 155]
 [ 52  27  98 180  93  32 146  32 132 123  60 166  27  98 180  93  32 146
   32 132]
 [189 134  69 187  42 199  60  26  70  95 186 132 134  69 187  42 199  60
   26  70]
 [135  93  94 182 139  46 122  52  74 112   9 188  93  94 182 139  46 122
   52  74]
 [ 62   8  20 146 115 135  29 138  85 133 104 118   8  20 146 115 135  29
  138  85]]
```

Question 3

Q3. Perform the following operations on the specified images using any software

Software Used - Python

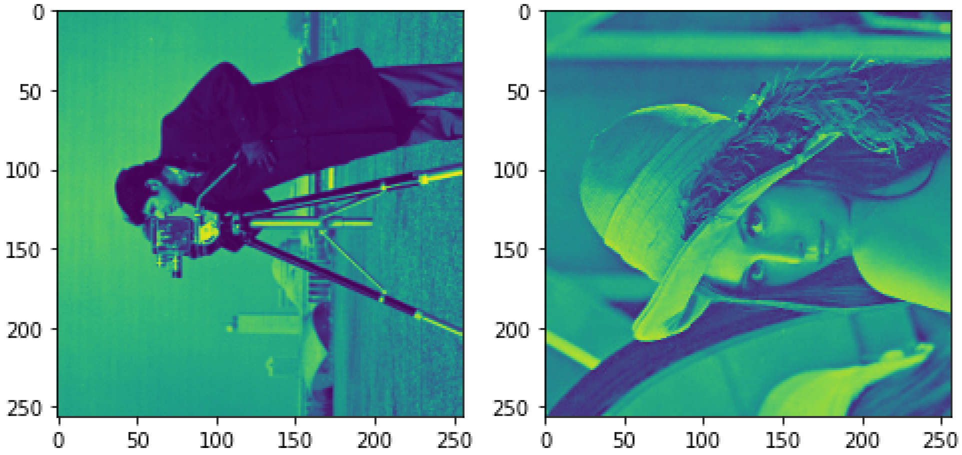
```
# libraries import
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt

# function to print images
def showImg(I,J):
    image = [I,J]
    w=256
    h=256
    fig=plt.figure(figsize=(8, 8))
    columns = 2
    rows = 1
    for i in range(1, columns*rows +1):
        img = image[i-1]
        fig.add_subplot(rows, columns, i)
        plt.imshow(img)
    plt.show()

I = mpimg.imread('cameraman.png')
J = mpimg.imread('lena.jpg')

showImg(I,J)
```



a) Transpose of I and J.

```
# Question 3a
def transpose(A,B):
    A_trans = np.transpose(A)
    B_trans = np.transpose(B)
    showImg(A_trans,B_trans)

showImg(I,J)
transpose(I,J)
```

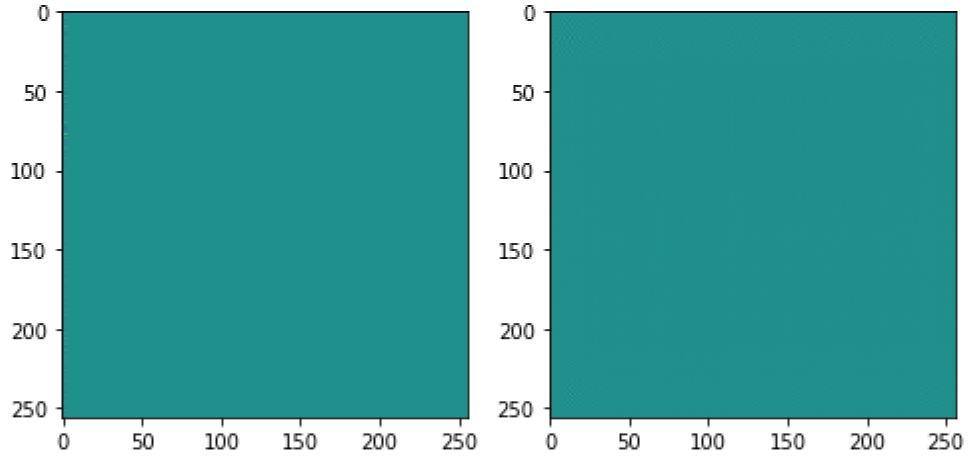
Results:



b) Inverse of images I and J.

```
# Question 3b
def inverse(A,B):
    A_inv = np.linalg.inv(A)
    B_inv = np.linalg.inv(B)
    showImg(A_inv,B_inv)

showImg(I,J)
inverse(I,J)
```

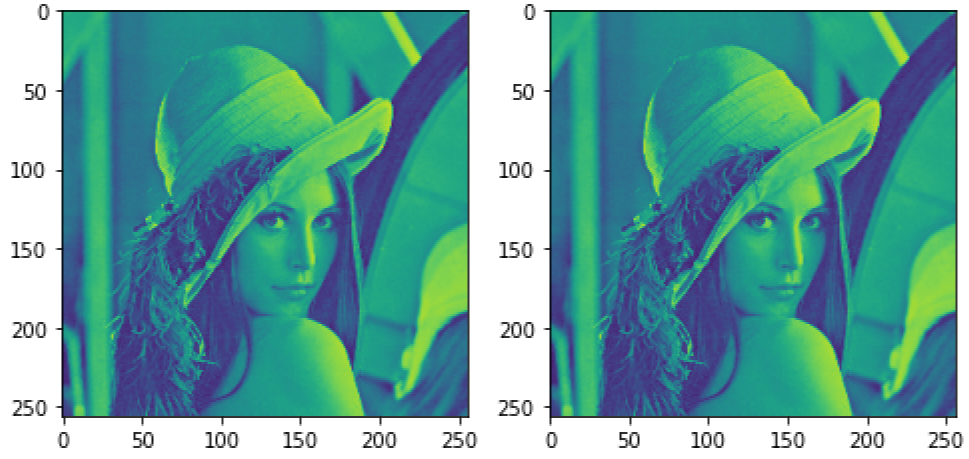
Results:



c) Addition (I+J) and (J+I). Comment on the result.

```
# Question 3c
def addImg(A,B):
    add_AB = np.array(A + B)
    add_BA = np.array(B + A)
    showImg(add_AB,add_BA)
    if np.array_equal(add_AB,add_BA):
        print("Both addition are same")
    else:
        print("Both addition are different")

showImg(I,J)
addImg(I,J)
```

Result:

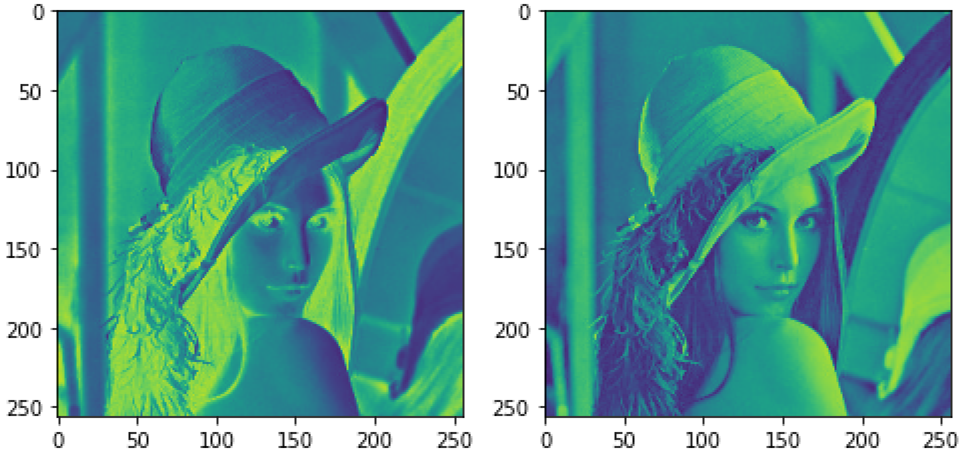


Comment: Both additions are the same i.e. order of addition doesn't matter.

d) Subtraction (I-J) and (J-I). Comment on the result.

```
    # Question 3d
    def subImg(A,B):
        sub_AB = np.array(A - B)
        sub_BA = np.array(B - A)
        showImg(sub_AB,sub_BA)
        if np.array_equal(sub_AB,sub_BA):
            print("Both Substraction are same")
        else:
            print("Both Substraction are different")

    showImg(I,J)
    subImg(I,J)
```

Results:

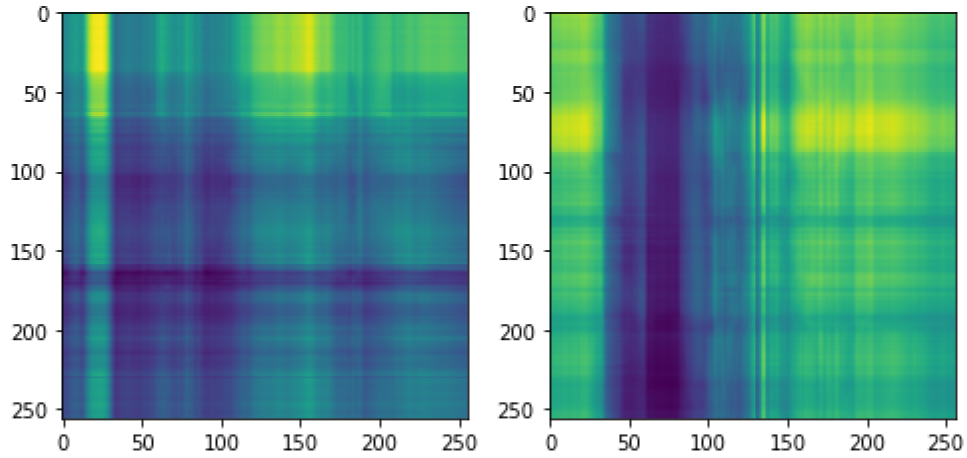


Comment: Both subtraction are different i.e. order of subtraction matters.

e) Multiplication (I*J) and (J*I). Comment on the result.

```
    # Question 3e
    def mulImg(A,B):
        mul_AB = A.dot(B)
        mul_BA = B.dot(A)
        showImg(mul_AB,mul_BA)
        if np.array_equal(mul_AB,mul_BA):
            print("Both multiplication are equal")
        else:
            print("Both multiplication are different")

    showImg(I,J)
    mulImg(I,J)
```

Result:



Comment: Both multiplications are different i.e. order of multiplication matters.

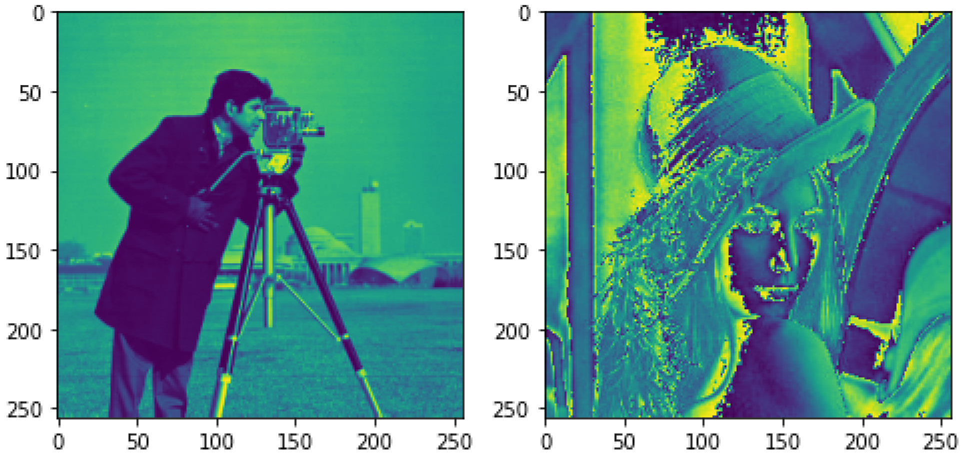
f) Multiply with a scalar to both images I and J. Comment on the result.

1. **C > 1**
2. **C < 1**

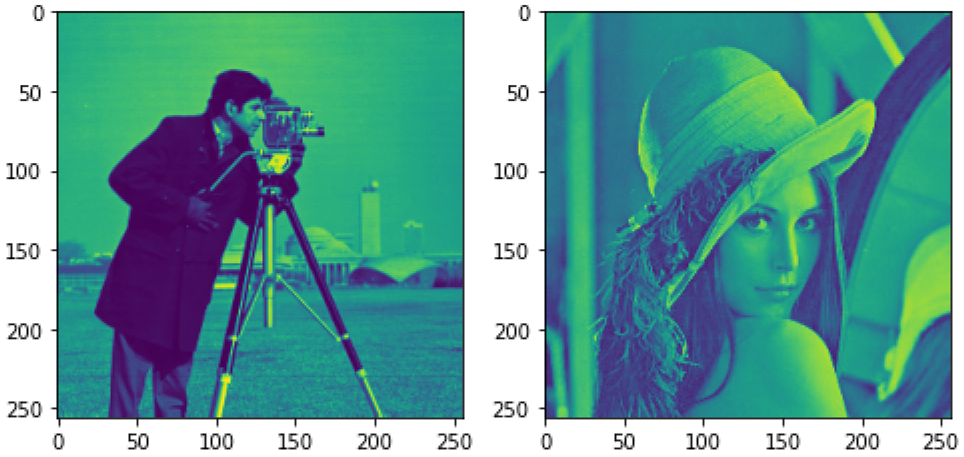
```
    # Question 3f
    c1 = 2
    c2 = 0.5
    def scalar_mul(A,B):
        c1_A = c1*A
        c2_A = c2*A
        c1_B = c1*B
        c2_B = c2*B
        showImg(c1_A,c1_B)
        showImg(c2_A,c2_B)

    showImg(I,J)
    scalar_mul(I,J)
```

Results:

C>1 :



C<1 :



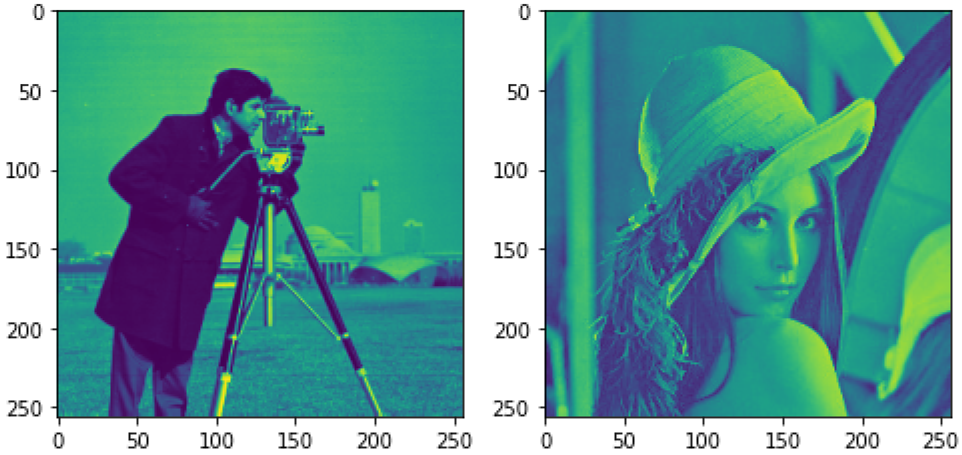
Comment: for C>1 value at every index increases and C<1 value at every index decreases.

g) Divide with a scalar to both images I and J. Comment on the result.

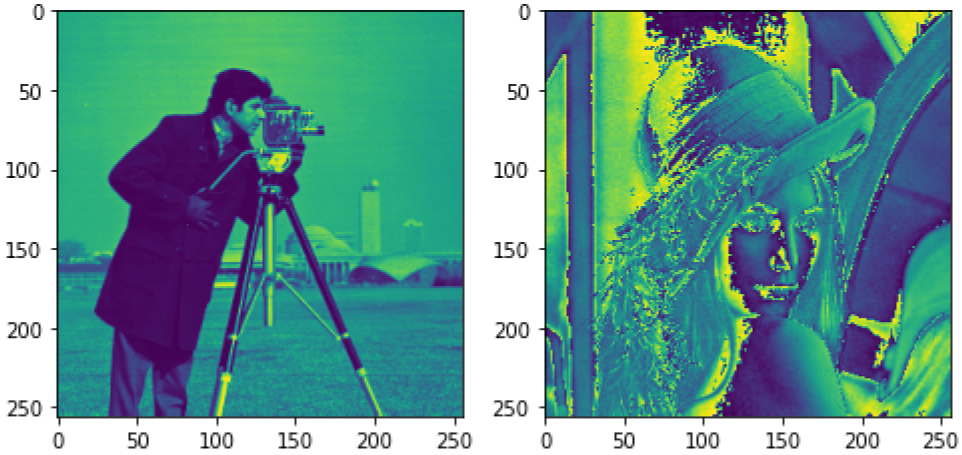
1. **C > 1**
2. **C < 1**

```
# Question 3g
c1 = 2
c2 = 0.5
def scalar_div(A,B):
    c1_A = A/c1
    c2_A = A/c2
    c1_B = B/c1
    c2_B = (B/c2)%256
    showImg(c1_A,c1_B)
    showImg(c2_A,c2_B)
showImg(I,J)
scalar_div(I,J)
```

Results:
For C>1 :



For C<1 :

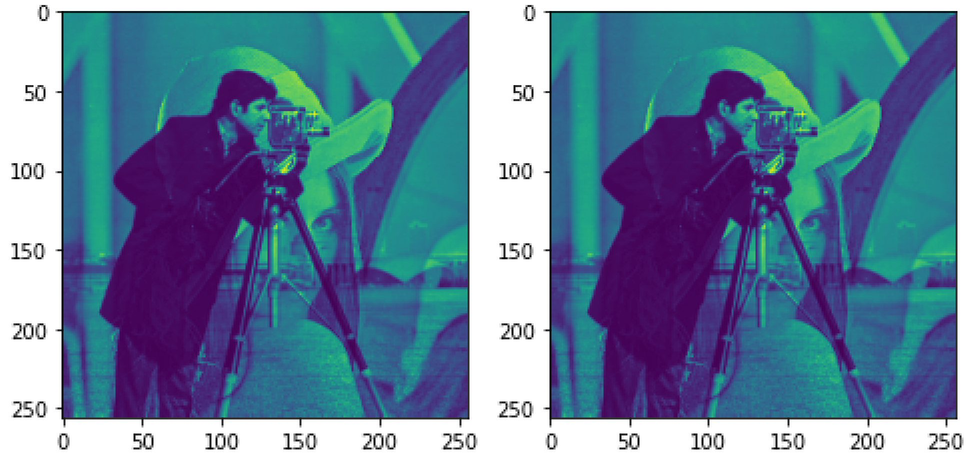


Comment: for C<1 value at every index increases and C>1 value at every index decreases

h) Element by element multiplication (I.*J) and (J.*I). Comment on the result.

```
    # Question 3h
    def ele_mul(A,B):
        ele_AB = np.multiply(A,B)
        ele_BA = np.multiply(B,A)
        showImg(ele_AB,ele_BA)
        if np.array_equal(ele_AB,ele_BA):
            print("Both multiplication are same")
        else:
            print("Both multiplication are different")

    showImg(I,J)
    ele_mul(I,J)
```

Results:



Comment: Both results are same i.e. independent of order

i) Find out the location(s) of a specific value....X (Value of X you can select) on both I and J.

```
    # Question 3i
    f_ele_A = 0.6117647
    f_ele_B = 160

    def find_loc(A,B):

        loc_A = np.argwhere(A==f_ele_A)
        loc_B = np.argwhere(B==f_ele_B)

        print("locations of 0.6117647 in A are: \n")
        for i in range(len(loc_A)):
            print(loc_A[i])

        print("locations of 160 in B are: \n")
        for i in range(len(loc_B)):
            print(loc_B[i])

    find_loc(I,J)
```

Results:

locations of 0.6117647 in A are:

```
[[  0   0]
```

```
 [  0   5]
 [  0 234]
 ...
 [251  93]
 [253   5]
 [253  87]]
```

locations of 160 in B are:

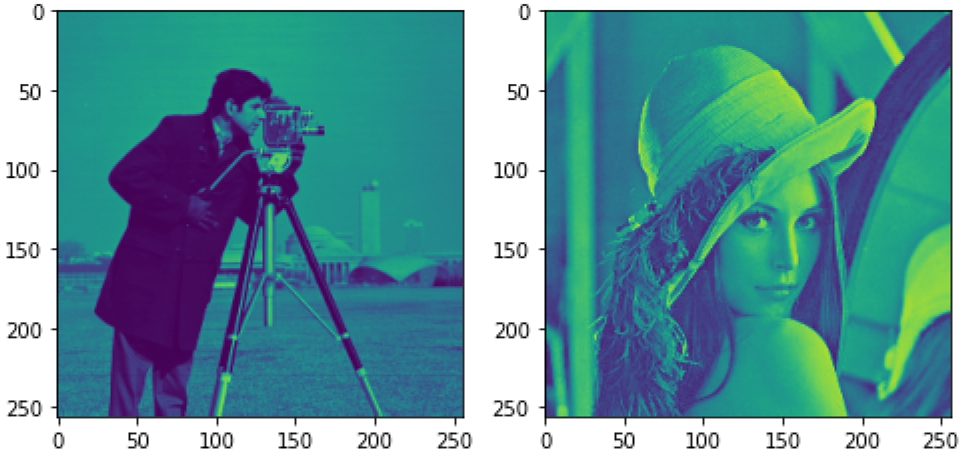
```
[[  0 169]
 [  1   1]
 [  1 170]
 ...
 [254  29]
 [254 164]
 [255 164]]
```

j) Find the specific value of X (only first occurrence) using the scan and search mechanism and amplify the value by a factor of 2. (Value of X you can select)

```
# Question 3j
f_ele_A = 0.6117647
f_ele_B = 160

def first_loc(A,B):

    loc_A = np.argwhere(A==f_ele_A)
    loc_B = np.argwhere(B==f_ele_B)

    first_loc_A = loc_A[0]
    first_loc_B = loc_B[0]

     print("First location of 0.6117647 in A is : ",first_loc_A,
'\n')

    print("First location of 160 in B is : ",first_loc_B,'\n')

    x_A = first_loc_A[0];
    y_A = first_loc_A[1];

    x_B = first_loc_B[0];
    y_B = first_loc_B[1];

    new_A = np.copy(A)
    new_B = np.copy(B)

    new_A[x_A][y_A]*=2
    new_B[x_B][y_B]*=2

    showImg(new_A,new_B)

showImg(I,J)
first_loc(I,J)
```

Result:

```
First location of 0.6117647 in A is :  [0 0]

First location of 160 in B is :  [0 169]
```

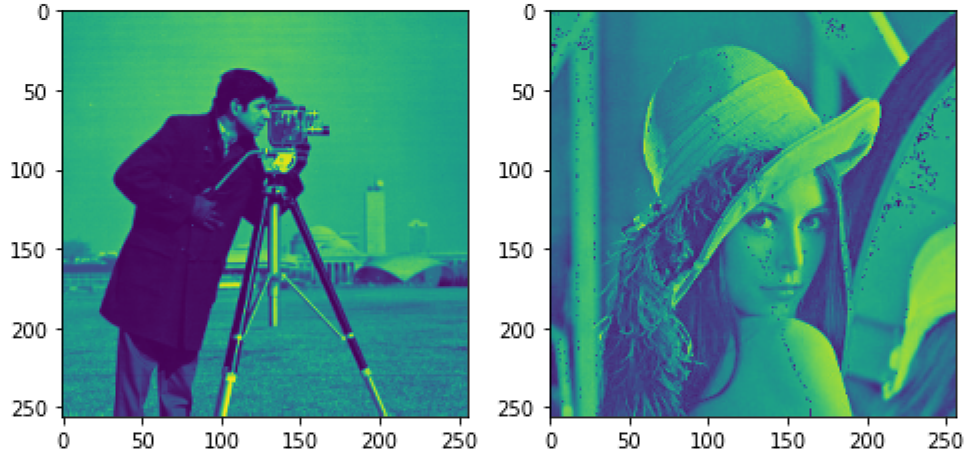


k) Search by scan and search mechanism (may be available multiple times) in the given images and replace those values with your birth date. (Value of X you can select). Give the count value i.e. occurrence value. (Do it for both I and J)

```
# Question 3k
def replace_with_bd(A,B):
    # MY BIRTHDAY
    my_db = 20

    f_ele_A = 0.6117647
    f_ele_B = 160

    loc_A = np.argwhere(A==f_ele_A)
    loc_B = np.argwhere(B==f_ele_B)

    print("Total locations of 0.6117647 in A are" , len(loc_A))
    print("Total locations of 160 in B are" , len(loc_B))

    new_A = np.copy(A)
    new_B = np.copy(B)

    new_A = np.where(new_A==f_ele, my_db, new_A)
    new_B = np.where(new_B==f_ele, my_db, new_B)

    showImg(new_A,new_B)

showImg(I,J)
replace_with_bd(I,J)
```

Results:

```
Total locations of 0.6117647 in A are 597
Total locations of 160 in B are 530
```

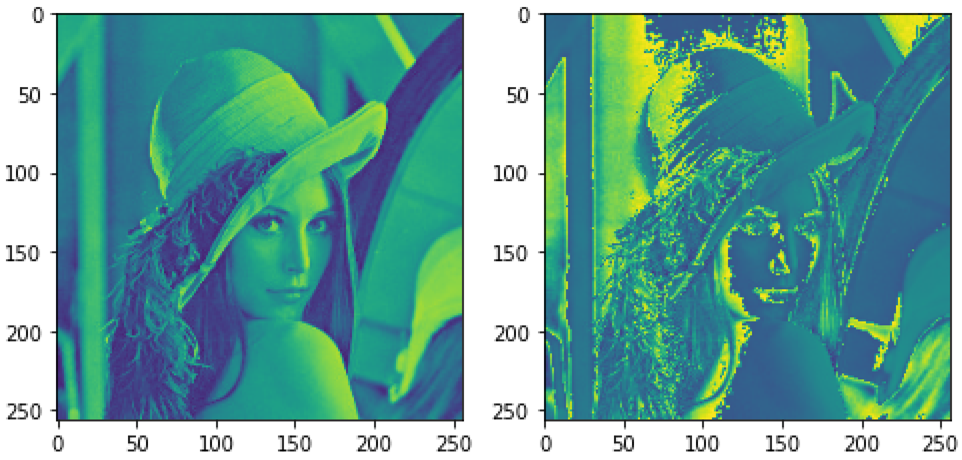


1) Perform following operations on I and J:

a. Multiply the intensity values with a constant 0.3 if the intensity value is greater than 127.

```
# Question 31-a
def mul_cond1(A, ele, ch):
    new_A = np.copy(A)
    new_A = np.where(new_A > ele, ch*new_A, new_A)
    showImg(A,new_A)

mul_cond1(I,127,0.3)
mul_cond1(J,127,0.3)
```

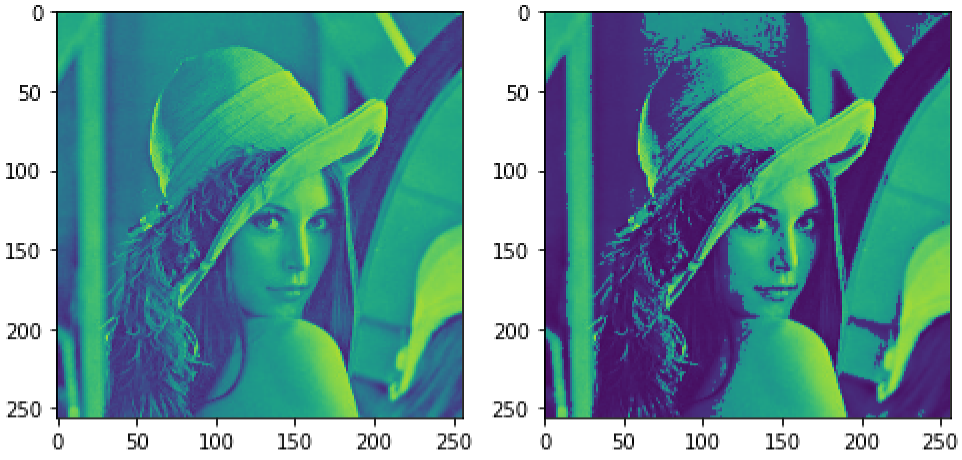
Result:



b. Multiply the intensity values with a constant 0.3 if the intensity value is less than 127.

```
# Question 31-b
def mul_cond2(A, ele, ch):
    new_A = np.copy(A)
    new_A = np.where(new_A < ele, ch*new_A, new_A)
    showImg(A,new_A)

mul_cond2(I,127,0.3)
mul_cond2(J,127,0.3)
```

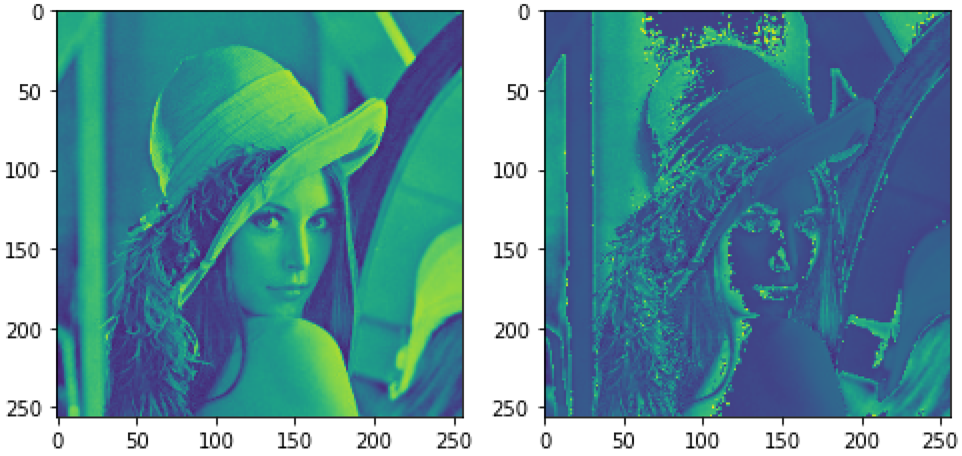
Result:



c. Multiply the intensity values with a constant 0.3 if the intensity value is greater than 127 and with a constant 0.7 if it is less than 128.

```
# Question 31-c
def mul_cond3(A, ele1, ele2, ch1, ch2):
    new_A = np.copy(A)
    new_A = np.where(new_A > ele1, ch1*new_A, new_A)
    new_A = np.where(new_A < ele1, ch2*new_A, new_A)
    showImg(A,new_A)

mul_cond3(I,127,128,0.3,0.7)
mul_cond3(J,127,128,0.3,0.7)
```

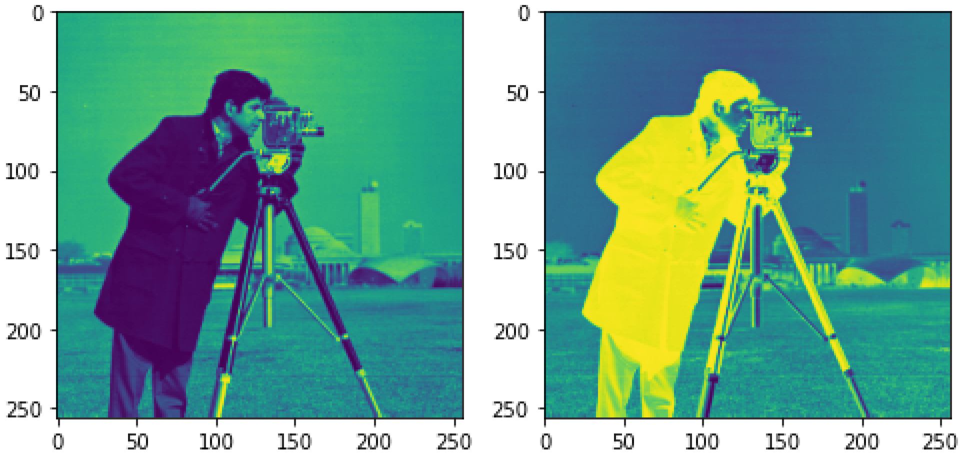
Result:



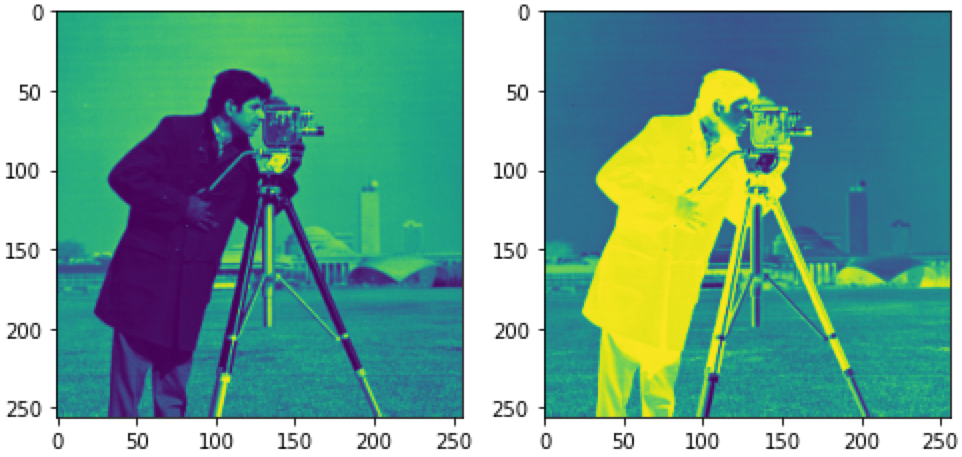
d. **Multiply the intensity values with an equation E1 if the intensity value is greater than 127 and with an equation E2 if it is less than 128. E1= 0.3x+2; x can take value as x=1, 2 and 3. Show and compare the results. E1= 0.3x-2; x can take value as x= 1, 2 and 3. Show and compare the results.**

```
# Question 31-d
def mul_cond3(A, ele1, ele2):
    for x in range(1,4):
        new_A = np.copy(A)
            new_A = np.where(new_A > ele1, (0.3*x + 2)*new_A,
new_A)
            new_A = np.where(new_A < ele1, (0.3*x - 2)*new_A,
new_A)
        print("for x =",x)
        showImg(A,new_A)

mul_cond3(I,127,128)
mul_cond3(J,127,128)
```

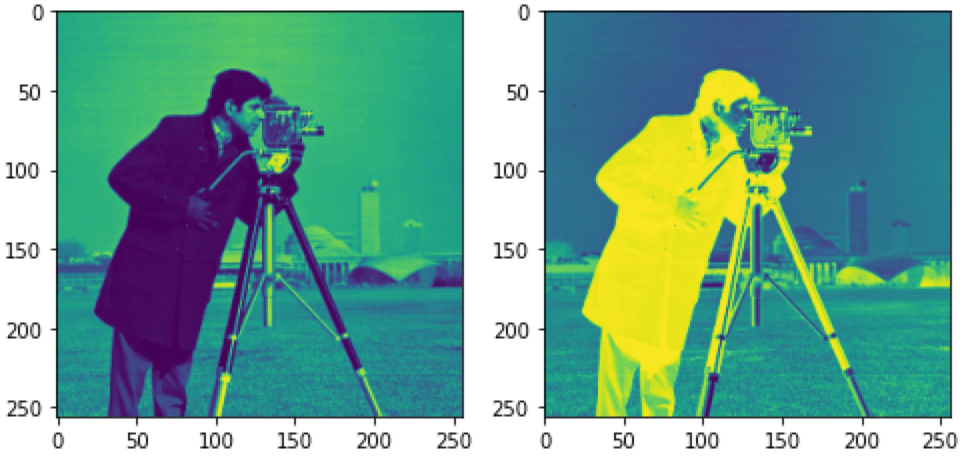
Results:
for x = 1



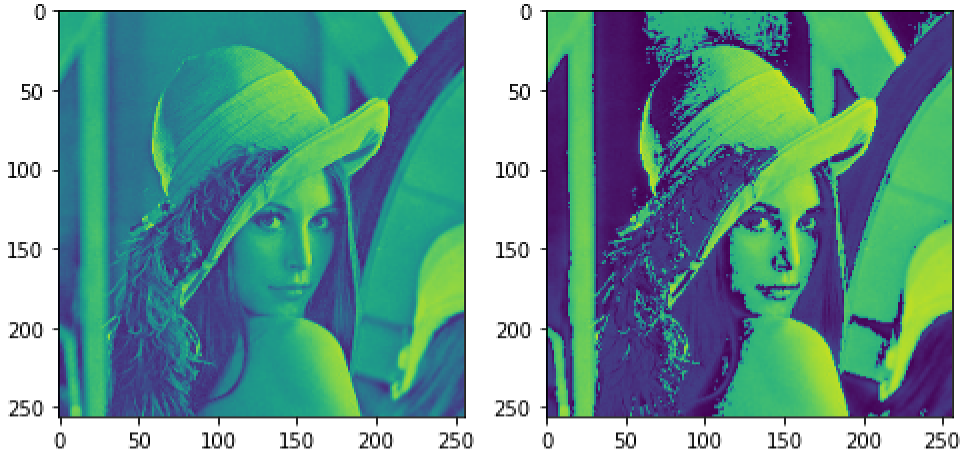
for x = 2



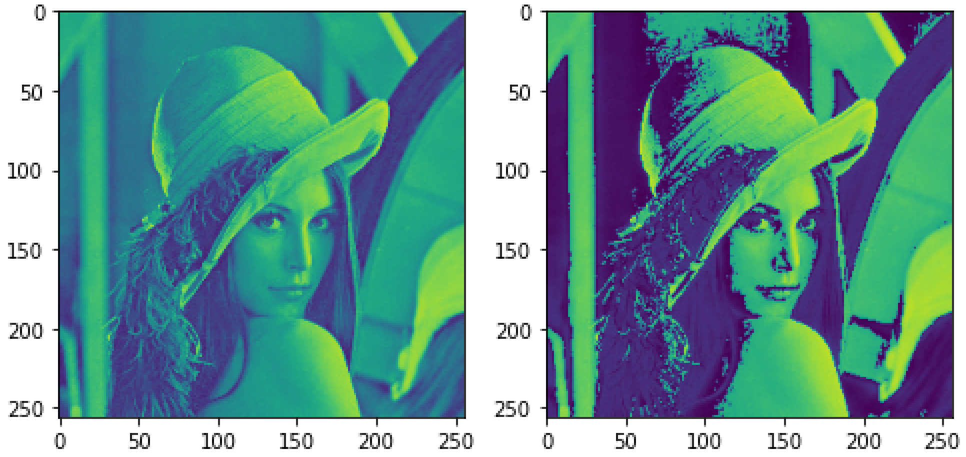
for x = 3



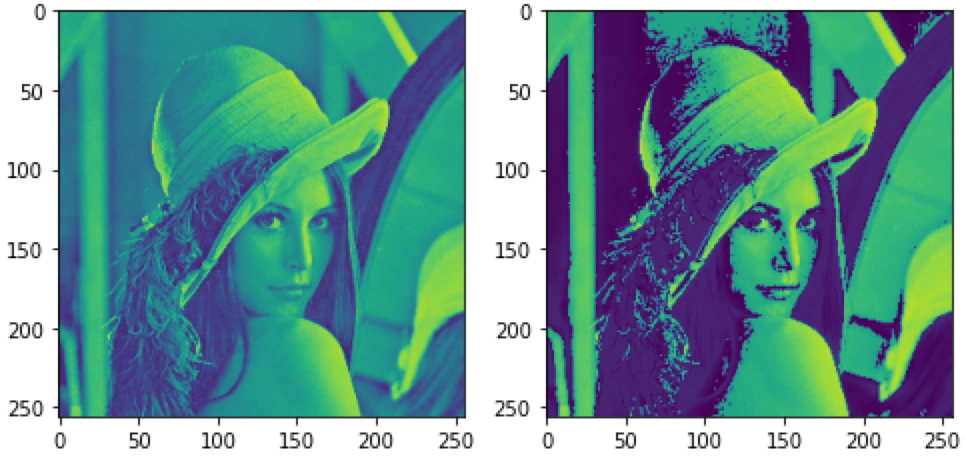
for x = 1


0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250

for x = 2


0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250

for x = 3


0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250