Part 2 (EBU5303)

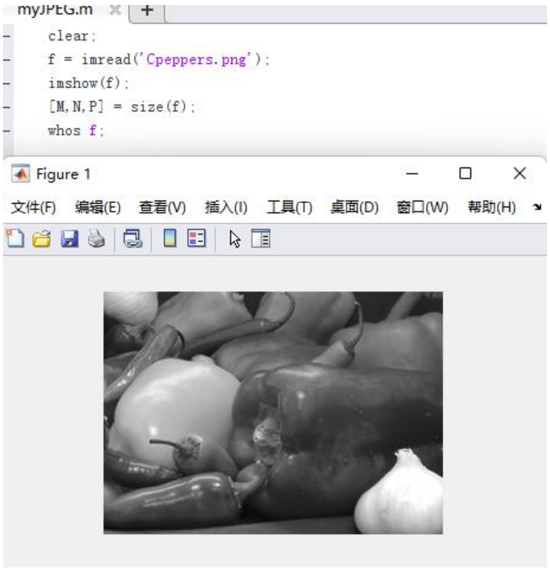
4. Image preparation

a. Create a Matlab function in the file "**myJPEG.m**". **Read** the grayscale image file Cpeppers.png using the `imread` matlab function and display it with `imshow`. ☑

b. What is the dimension of the image (width x hight)? What is its size in bits? Calculate the number of 8x8 blocks of pixels contained in the image.

Your answer.

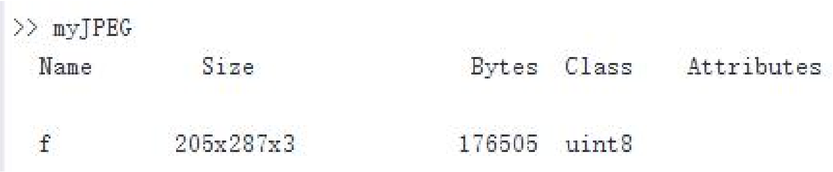
A. As shown in figure



```
clear;
f = imread('Cpeppers.png');
imshow(f);
[M,N,P] = size(f);
whos f;
```

myJPEG.m
Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)

B. Width of the image is 205 and its height is 287. Its size is 176505 bits.



```
>> myJPEG
  Name        Size                 Bytes  Class    Attributes

  f         205x287x3             176505  uint8
```

After add 3 all-zero rows and 1 all-zero columns at last, there are $26*36 = 936$ 8x8 blocks of pixels.

c. In myJPEG.m, for one of the channels only and working from left to right, top to bottom, **split** the image into 8x8 blocks of pixels. Comment your code.

d. **Print** the first two 8x8 matrices of pixel values as examples (use the `disp` matlab function).

☑

C. Please check in the source code

D. Copy and paste the matrices

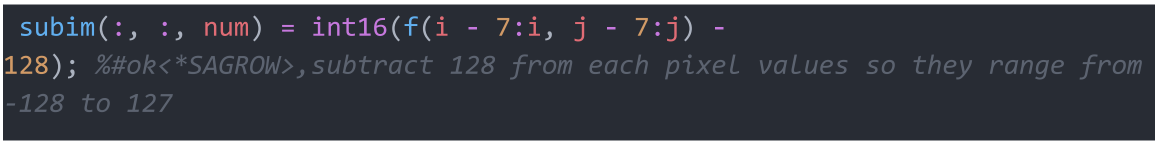
val(:,:,1) =	val(:,:,2) =
177 185 188 190 193 198 202 206	214 213 210 214 219 221 221 221
176 181 181 186 191 198 200 199	201 206 211 216 218 219 220 223
175 174 175 184 192 193 191 196	202 208 210 212 214 215 218 223
169 169 175 186 187 186 192 197	201 203 205 206 209 215 220 222
162 170 177 177 180 187 191 195	194 195 198 202 208 212 208 204
167 173 173 172 178 184 186 185	186 190 198 201 199 193 196 203
168 166 169 173 174 175 177 181	188 193 192 189 190 195 200 202
160 162 166 169 168 170 178 182	181 178 178 185 189 193 194 197

e. In myJPEG.m, **subtract** 128 from each pixel values so they range from -128 to 127. Comment your code.

f. **Print** (`disp`) again the first two 8x8 matrices of pixel values and check they have been "levelled off".

☑

E.



```
subim(:, :, num) = int16(f(i - 7:i, j - 7:j) -
128); %#ok<*SAGROW>,subtract 128 from each pixel values so they range from
-128 to 127
```

F.

Copy and paste the "levelled off" matrices

val(:,:,1) =	val(:,:,2) =
49 57 60 62 65 70 74 78	86 85 82 86 91 93 93 93
48 53 53 58 63 70 72 71	73 78 83 88 90 91 92 95
47 46 47 56 64 65 63 68	74 80 82 84 86 87 90 95
41 41 47 58 59 58 64 69	73 75 77 78 81 87 92 94
34 42 49 49 52 59 63 67	66 67 70 74 80 84 80 76
39 45 45 44 50 56 58 57	58 62 70 73 71 65 68 75
40 38 41 45 46 47 49 53	60 65 64 61 62 67 72 74
32 34 38 41 40 42 50 54	53 50 50 57 61 65 66 69

5. Applying 2D DCT to each block

a. In myJPEG.m, working from left to right, top to bottom, **apply** the 2D DCT Matlab function (`dct2`) to each block of pixel values. Save the results in 8x8 matrices of DCT coefficients values. Comment your code. ☑

b. **Print** (`disp`) the first two 8x8 matrices of DCT coefficients values.

A. See the source code

B. Copy and paste the DCT matrices

```
val(:,:,1) =

 424.3750  -62.0000  -0.6813  -5.1237   0.3750  -1.4346  -0.6649  -1.2739

  57.7108   -9.0649  -3.3671   2.9944  -0.5170  -2.2814  -0.6802  -1.3703

  -1.1829    6.1159   3.6062  -1.1671  -1.5656   0.3987  -0.7134  -0.2871

   9.6860    0.8428   1.7191  -1.9384  -8.4881  -2.0568   0.6694   0.3861

   0.1250   -6.7614   0.2986  -9.1746   0.1250   1.9577  -0.2590   0.4650

   1.0768    4.2101  -1.6908   0.0342   7.0466   1.0042   0.1358   0.1535

   0.0841    0.4127   0.5366   1.6367  -0.8398  -2.4026  -0.1062   0.4959

  -1.0250   -0.2793  -0.3653  -0.5293   0.5857  -1.6018   2.3792  -0.0010

dval(:,:,2) =

 608.6250  -42.2834  -1.6052  -3.0370   0.6250  -1.9839  -0.2822  -0.8022

  80.9429   -1.6047  -2.4039   1.2992  -0.3080  -1.0264  -0.7065  -0.1669

 -12.4240    1.8571   4.2097   3.1631   2.1724  -0.2912  -0.8776  -1.0840

  -0.1805    9.3519  -5.0183   0.1354   0.7503   0.0937   0.0392   1.0371

  -0.1250   -3.3787   2.2585  11.7610  -0.1250   1.3619   0.5528  -0.7723

   8.2761    3.4985  10.7441  -2.1260  -3.2571  -0.5015  -1.0710   0.3844

  -4.1895    3.1755  -2.1276  -2.7741   4.9180  -0.4926  -0.2097   0.3725

   1.5901    3.1134   0.1022   5.4939  -3.2475  -1.2363  -0.2626  -0.5291
```

6. Quantisation

a. In myJPEG.m, **create** a standard quantisation matrix named Q_{std} similar to the matrix shown below. ☑

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

b. In myJPEG.m, **create** two more quantisation matrices, one for lower quantisation (Q_{low}) and one for higher quantisation (Q_{high}). Comment your code. ☑

Hint: use quantisation factors to multiply Q_{std}. The scaled matrices must then be rounded and clipped to have positive integer values ranging from 1 to 255.

c. **Print** (`disp`) Q_{low} and Q_{high} and explain how you created them (i.e., what quantisation factor you chose and why).

Copy and paste the quantisation matrices and explain your choice of quantisation factors.

Qlow	Qhigh
5 3 3 5 7 12 15 18	32 22 20 32 48 80 102 122
4 4 4 6 8 17 18 17	24 24 28 38 52 116 120 110
4 4 5 7 12 17 21 17	28 26 32 48 80 114 138 112
4 5 7 9 15 26 24 19	28 34 44 58 102 174 160 124
5 7 11 17 20 33 31 23	36 44 74 112 136 218 206 154
7 11 17 19 24 31 34 28	48 70 110 128 162 208 226 184
15 19 23 26 31 36 36 30	98 128 156 174 206 242 240 202
22 28 29 29 34 30 31 30	144 184 190 196 224 200 206 198

The quantization coefficients are 0.4 and 2 respectively. They were chosen for convenience and not out of range (2 is the largest integer that keeps the original matrix in range)

其实不会答捏

d. In myJPEG.m, **apply** quantisation to all the 8x8 matrices of DCT coefficients, using Q_{std}, Q_{low} and Q_{high}. Comment your code. ☑

Hint: quantisation is achieved by dividing each DCT coefficient by the corresponding element in the quantisation matrix, and then rounding to the nearest integer value.

e. **Print** (`disp`) the first two 8x8 matrices of quantised DCT coefficients for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total.

Copy and paste the six quantised matrices here.

	The first matrix	The second

Apply Qstd	27 -6 0 0 0 0 0 0
	5 -1 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	1 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		38 -4 0 0 0 0 0 0
		7 0 0 0 0 0 0 0
		-1 0 0 0 0 0 0 0
		0 1 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

Apply Qlow	85 -21 0 -1 0 0 0 0
	14 -2 -1 0 0 0 0 0
	0 2 1 0 0 0 0 0
	2 0 0 0 -1 0 0 0
	0 -1 0 -1 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		122 -14 -1 -1 0 0 0 0
		20 0 -1 0 0 0 0 0
		-3 0 1 0 0 0 0 0
		0 2 -1 0 0 0 0 0
		0 0 0 1 0 0 0 0
		1 0 1 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

Apply Qlow	13 -3 0 0 0 0 0 0
	2 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		19 -2 0 0 0 0 0 0
		3 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

f. **Compare** the 6 matrices and **comment**.

Your answer.

g. Comment on how the choice of quantisation matrix affects the compression rate.

✓

Your answer.

7. Decompression

a. In myJPEG.m, multiply the quantised DCT values by the corresponding element of the quantisation matrix originally used (i.e., Q_{std}, Q_{low} or Q_{high}). Comment your code. ☑

b. **Print** (`disp`) the first two 8x8 matrices of DCT coefficients for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total. **Compare** with the two matrices of question 5b. and **comment**.

Copy and paste the matrices, and comment.

	1st	2st

The standard one	432 -66 0 0 0 0 0 0
	60 -12 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	14 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		608 -44 0 0 0 0 0 0
		84 0 0 0 0 0 0 0
		-14 0 0 0 0 0 0 0
		0 17 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

lthe low quantisd one	425 -63 0 -5 0 0 0 0
	56 -8 -4 0 0 0 0 0
	0 8 5 0 0 0 0 0
	8 0 0 0 -15 0 0 0
	0 -7 0 -17 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		610 -42 -3 -5 0 0 0 0
		80 0 -4 0 0 0 0 0
		-12 0 5 0 0 0 0 0
		0 10 -7 0 0 0 0 0
		0 0 0 17 0 0 0 0
		7 0 17 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

the high quantisd one	416 -66 0 0 0 0 0 0
	48 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
	0 0 0 0 0 0 0 0
		608 -44 0 0 0 0 0 0
		72 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

c. In myJPEG.m, **apply** the inverse DCT (`idct2`) to all 8x8 matrices of DCT coefficients (standard, low and high quantisation). **Add** 128 to each matrix element and **round** the values. Comment your code.

d. **Print** (`disp`) the first two 8x8 matrices of pixels for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total. **Compare** with the two matrices of question

Copy and paste the matrices, and comment.

	1st	2st

The standard one	180 182 186 192 197 203 207 209
	176 179 182 188 193 198 202 204
	172 174 178 183 188 193 197 199
	171 173 176 180 185 190 193 195
	170 172 175 179 183 187 191 192
	169 170 173 177 180 184 187 188
	165 166 169 172 175 179 181 183
	161 162 165 168 171 174 177 178
		608 -44 0 0 0 0 0 0
		84 0 0 0 0 0 0 0
		-14 0 0 0 0 0 0 0
		0 17 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0
		0 0 0 0 0 0 0 0

lthe low quantisd one	174 185 191 189 190 197 204 205
	180 179 181 187 193 197 198 198
	177 172 174 185 192 192 192 197
	165 169 177 184 186 186 192 200
	161 170 178 179 180 186 192 194
	167 171 172 171 177 187 188 182
	168 168 167 169 174 179 181 179
	163 162 166 171 170 168 175 186
		214 212 212 214 219 222 222 221
		199 206 215 219 218 216 218 221
		201 206 211 212 212 214 221 227
		206 203 202 205 210 216 219 219
		193 195 199 206 210 210 206 201
		186 192 199 201 199 197 199 202
		187 190 191 189 188 191 200 207
		181 180 180 184 190 195 196 196

the high quantisd one	177 179 182 186 191 195 198 200
	176 177 181 185 189 194 197 198
	173 175 178 182 187 191 194 196
	170 172 175 179 184 188 191 193
	167 169 172 176 181 185 188 190
	164 166 169 173 178 182 185 187
	162 163 166 171 175 179 183 184
	160 162 165 169 174 178 181 183
		209 210 212 215 218 221 223 224
		207 208 210 213 216 219 221 222
		203 205 207 210 213 215 218 219
		199 200 202 205 208 211 213 214
		194 195 197 200 203 206 208 209
		189 190 193 195 198 201 203 205
		186 187 189 192 195 198 200 201
		184 185 187 190 193 196 198 199

4.d and **comment**.

8. Image reconstruction

a. In myJPEG.m, **reconstruct** the images from the 8x8 blocks of pixel values you obtained in question 8 for the three different levels of quantisation (standard, low and high). Comment your code. ☑

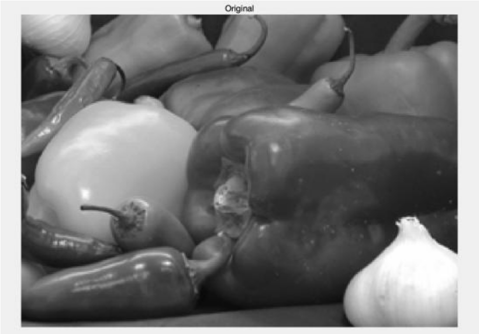
b. **Display** and **save** the four images: original, standard, low and high compression. ☑

c. **Compare the images and comment**. *You should clearly see blocky artefacts in the high compression image. If not, go back to Question 6 and modify the Q_{high} matrix.*

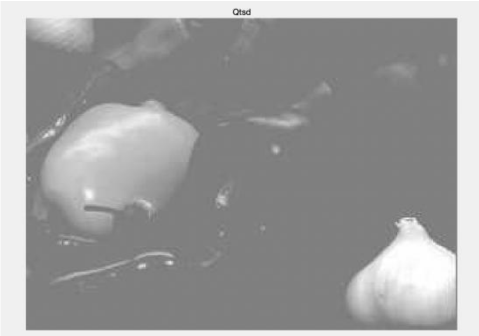
Hint: You may have to cast your pixel values to uint8 to display the images.

Your comments.

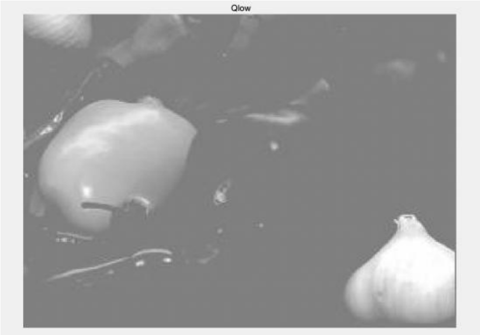
Original



Qstd



Qlow



Qhigh