Your task will be to code a simulation of image compression based on the approximate low rank structure of the set of image patches. You will write functions

- code = my_block_compress(I, B, nz);
- I = my_block_reconstruct(code, B, m, n);

The function `my_block_compress` takes as input an \(m \times n\) image grayscale \(I\), extracts all the distinct \(8 \times 8\) blocks from the image using the matlab command `im2col`, and finds the representation of each block in the \((64 \times 64)\) orthonormal basis \(B\), and then discards all but the \(nz\) largest magnitude coefficients over all the blocks in the image. The output \(code\) is a \(nz \times 3\) array which gives the \(nz\) most valuable coefficients for reconstructing all the patches, and the locations. It should be structured as follows: each entry of the first column of the variable \(code\) contains a coefficient. Each entry in the second column specifies which block that coefficient belongs to. Each entry in the third column specifies which basis element the coefficient belongs to.\(^1\)

The function `my_block_reconstruct` takes as input a coded array \(code\), a \(64 \times 64\) basis \(B\), and output size parameters \(m\) and \(n\). It outputs an \(m \times n\) image \(I\) reconstructed from the coefficients in \(code\); you may use the function `col2im` if you find it convenient.

For example, suppose \(I\) is a \(24 \times 24\) image. We can write \(I\) in block form:

\[
I = \begin{pmatrix}
I_1 & I_4 & I_7 \\
I_2 & I_5 & I_8 \\
I_3 & I_6 & I_9
\end{pmatrix},
\]

where \(I_k\) is a \(8 \times 8\) matrix for \(k \in \{1, 2, ..., 9\}\). Using `im2col` on \(I\), we obtain

\[
J = (J_1 \ J_2 \ J_3 \ J_4 \ J_5 \ J_6 \ J_7 \ J_8 \ J_9) = \text{im2col}(I, [8 \ 8], \text{'distinct'}),
\]

where \(J_k\) is a \(64 \times 1\) column vector. If we write each column \(J_k\) of \(J\) in the \(64 \times 64\) basis \(B\), we get the matrix

\[
C = \begin{pmatrix}
C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 & C_9
\end{pmatrix}.
\]

\(^1\)Note this is not nearly the most efficient coding scheme; it has been chosen for simplicity of implementation
Your function `my_block_compress` should sort the absolute values of all the entries in $C$. The output of the function, `code`, should return the $nz$ largest absolute value coefficients, and their locations in $C$. The format of `code` is as follows:

- The first column of `code` gives the value of the coefficient.
- The second column gives the identity of the block the coefficient belongs to, that is, the column of $C$ the coefficient belongs to.
- The third column gives the identity of the basis element the coefficient belongs to, that is, the row of $C$ the coefficient belongs to.

The number $nz$, the basis $B$ and the image $I$ are inputs to the function. Finally, `my_block_reconstruct` should invert this process, reconstructing an image from a `code`.

You will then be provided with a training data set consisting of 200000 8 $\times$ 8 grayscale patches of images. You will need to reshape these into a $64 \times 200000$ matrix $X$, and construct a basis $B$ using the left singular vectors $X$. Compress the cow image from the website and then reconstruct using your basis. Compare the results with a random orthonormal basis of the form `orth(randn(64))`. 