1.) Code Setup
Download the maxflow code* from our course website and unzip the folder.

Then in Matlab, navigate to the maxflow folder. To install the software, type the word: make

- You should see a wall of text. You can test if it installed properly by typing: "test1".
  If you see "Flow=3 Labels=1 1", then it worked properly and you can skip to #2.
- If you see a red error message about the compiler, you can try to correct it by typing:
  "mex -setup". Then press "n" and select the SDK 7.1 compiler.
- If you see more error messages, then you are doomed. For practice, see if you can code up the matrices described in #2 and #3.

2.) Cutting a Graph
The maxflow code computes the minimum cut of a graph, but you have to build the graph. The code assumes all edge weights are positive. A zero entry in the adjacency matrix indicates no edge, so we can create a sparse matrix. The basic function call is:

```
[flow,labels] = maxflow(A,T);
```

where the entries of A and T satisfy:

- \( A(i,j) = \text{weight between nodes } i \text{ and } j \) (regularization term)
- \( T(i,j) = \text{weight for assigning label } j \text{ to node } i \) (fidelity term)

Note the maxflow code only supports L=2 labels (binary cuts), so T must have 2 columns.

An example is shown below. Try to compute the value of the min cut by hand. Then confirm it in Matlab. Note the output labels is a 0-1 vector indicating the different sets separated by the cut.

```
A=sparse(3,3);
A(2,1)=3;   A(1,2)=3;
A(3,2)=5;   A(2,3)=5;
T=sparse(3,2);
T(1,1)=18;  T(2,1)=6;  T(3,1)=10;
T(1,2)=41;  T(2,2)=4;  T(3,2)=2;
[flow,labels] = maxflow(A,T)
```

3.) Chan-Vese Segmentation (*1st Attempt*)

Recall the 2-phase Chan-Vese segmentation model that we coded up in the previous lab:

\[
\min_{\Gamma} E_{CV}[\Gamma|f] = L(\Gamma) + \lambda_{\text{in}} \int_{\text{inside } \Gamma} (f - c_{\text{in}})^2 \, d\tilde{x} + \lambda_{\text{out}} \int_{\text{outside } \Gamma} (f - c_{\text{out}})^2 \, d\tilde{x}
\]

\[
c_{\text{in}} = \frac{\int_{\text{inside } \Gamma} f \, d\tilde{x}}{\int_{\text{inside } \Gamma} d\tilde{x}} \quad \quad \quad \quad \quad c_{\text{out}} = \frac{\int_{\text{outside } \Gamma} f \, d\tilde{x}}{\int_{\text{outside } \Gamma} d\tilde{x}}
\]

The flow network corresponding to CV segmentation on an image with N pixels is:

![Flow Network Diagram]

To create the matrices A and T, you should first convert the input image f to a vector. The `reshape` command reads the matrix column by column.

\[
\text{[m,n]} = \text{size}(f); \quad \text{f}_x = \text{reshape}(m*n, 1);
\]

You will need to use the \( \text{f}_x \) vector and the values of \( c \) to compute the matrix \( T \).

Note that to compute the \( mn \times mn \) matrix A, you just need to fill in a 1 on an edge connecting each of the 4 neighbors. For example, the code below connects each pixel to the one to the right.

\[
A = \text{sparse}(m*n, m*n); \\
A(1:m*(n-1), m+1:m*n) = 1;
\]

Try your code on a small grayscale image, like 50x50. Initialize the values \( c_{\text{in}}=100, c_{\text{out}}=200 \), \( \lambda_{\text{in}} = \lambda_{\text{in}} = 0.1 \). You may wish to experiment other values of \( c \), as this may affect the result. If you `reshape` the output labels matrix to the size of the original image, you should be able to view the segmentation image. If you wish to view the contour as we did in the previous lab, you could try:

\[
\text{imagesc}(f); \quad \text{hold on;} \\
\text{contour(reshape(labels,[m,n]),[0.5,0.5],'r'); hold off;} \\
\]

The min cut gives just one iteration of CV segmentation. You should recompute the \( c \) values based on the \( \text{labels} \) vector, adjust the T matrix, and then recompute the min cut. Note you should not have to alter the A matrix, so the creation of A should be outside the loop.

4.) Chan-Vese Segmentation (*2nd Attempt*)

You are probably noticing that the code is very slow and that memory runs out quickly. This is because we are creating a large sparse matrix A in Matlab and passing it to the C++ compiler. A more efficient version would create matrix A in C++ using the maxflow library commands. Try using the code below to create matrix A. Note you only need to create the matrix A one time.

\[
E = \text{edges4connected}(m,n); \\
V = \text{ones(size(E(:,1)))}; \\
A = \text{sparse}(E(:,1),E(:,2),V,m*n,m*n,4*m*n);
\]