1. Theory Exercises

(a) Segment the following image,

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>199</td>
<td>230</td>
<td>240</td>
<td>251</td>
<td>234</td>
</tr>
<tr>
<td>192</td>
<td>209</td>
<td>160</td>
<td>184</td>
<td>207</td>
</tr>
<tr>
<td>185</td>
<td>170</td>
<td>120</td>
<td>146</td>
<td>39</td>
</tr>
<tr>
<td>130</td>
<td>101</td>
<td>63</td>
<td>52</td>
<td>28</td>
</tr>
<tr>
<td>117</td>
<td>89</td>
<td>44</td>
<td>19</td>
<td>3</td>
</tr>
</tbody>
</table>

(i) into a binary image of 2 regions, using threshold value, \( T = 150 \).
(ii) into an image of 3 regions, using two threshold values, \( T_1=55 \) and \( T_2=190 \).

(b) What is the advantage of Otsu's thresholding method over the Basic Global Thresholding method?

(c) In what kind of situations/problems encountered during image thresholding that we have to make use of these following techniques?
   - Image smoothing
   - Using edges as mask image
   - Image partitioning to use multiple local threshold values

2. Matlab Exercises: Thresholding

(a) Finding Threshold Value by Observing Histograms

Thresholding in Matlab is very simple. Use a logical operator to split the pixel intensity values into 2 groups – one which is TRUE according to the specified logic, and one which is FALSE. TRUE values will be assigned '1', while FALSE values will be assigned '0'. Thus, resulting in a two-level image of 1s and 0s.

```matlab
>> b = imread('bacteria.tif'); imshow(b);
>> c = b > 50;
>> figure, imshow(c);
```

It looks like we may have remove too much details in the image! The threshold value must be either too low or too high. What is a suitable threshold value?

Finding the threshold value by trial-and-error is a tiring effort... Try using histograms to give
you a “rough” idea of where the threshold is. You will notice that in images like eight.tif, it is easy to see where we can split it into 2 groups, but in images like bacteria.tif, the choice is not as clear.

So, what is the best threshold value for bacteria.tif and eight.tif?

(b) **im2bw Function**

Matlab provides a `im2bw` function, which helps you threshold an image of any data type (whether values are between 0-255 or 0-1 of type `uint8`, `uint16`, or `double`) and any image type (whether the image is grayscale, colored or indexed). Use the Help to find out how to use this function. Our previous simple method in part (a) may have some limitations, and this function helps to address those limitations.

*Note:* Try on various kinds of images such as colored (RGB) images or indexed images.

(c) **Basic Global Thresholding**

Basic Global Thresholding is an iterative procedure which is able to automatically estimate the best possible threshold value of an image, by taking into consideration the mean intensity values of both regions to be segmented. It's a global method since it considers the entire composition of intensity values in the image.

Refer back to the steps in this algorithm, and write a Matlab function, called `globalthresh` to perform Basic Global Thresholding. Test it on the following images:

- eight.tif
- bacteria.tif
- rice.tif
- circles2.tif (this new image is provided to you)

*Hint:* Function `globalthresh` will take 1 input – the input image matrix, and return 1 output – the thresholded image matrix.

*Note:* For the ΔT value, you may have to do some experimentation (a few times) to fine-tune its value. There’s no rule on how you intend to fix ΔT. It merely controls the number of iterations the algorithm runs.

(d) **Variable Thresholding with Image Partitioning**

You may have tried very hard to threshold circles2.tif with Basic Global Thresholding, but it probably still fails! The reason is that circles2.tif is corrupted by an additive
intensity ramp in its background, and that makes it impossible to threshold with one single value.

Let us try and use **Image Partitioning** to split the image into a number of sub-images or image blocks, threshold with their respective threshold values from each block (which should be more clear-cut), then combined the blocks back to form the original image.

One suggestion is to partition this image into 4 blocks (red lines indicate partitions), each block being 64 pixels wide.

```matlab
>> c = imread('circles2.tif');
>> p1 = c(:,1:64);
>> p2 = c(:,65:128);
>> p3 = c(:,129:192);
>> p4 = c(:,193:256);
```

Next, we want to threshold each block using any method (for e.g. Basic Global Thresholding):

```matlab
>> g1 = globalthresh(p1);
>> g2 = globalthresh(p2);
>> g3 = globalthresh(p3);
>> g4 = globalthresh(p4);
```

Then, display them as a single image:

```matlab
>> imshow([g1 g2 g3 g4]);
```

This method works decently, but it is tedious. We may be faced with too much manual coding if we want to partition an image into 25 blocks! Matlab has an elegant function called **blkproc** which is able to implement a predetermined inline function to each block (as many as we wish) in an image.

Firstly, define the inline function,

```matlab
>> fun = inline('globalthresh(x)');
```

This is the function which we want to repeat for ALL blocks.

Next, apply block processing,

```matlab
>> t_img = blkproc(c,[256 64],fun);
```

The second input parameter specifies the size of each block, which is 256 pixels high and 64 pixels wide. **blkproc** will automatically process each block from top to bottom, left to right with the inline function **fun**.

Have fun using **blkproc** on other images!