Notice

This homework requires you to create images, which can be difficult to check using `isequal`. To mitigate this, we have given you a function called `checkImage` that will compare two images. Use this to compare the image file that your code outputs with the image file that the solution code outputs. You can look at `help checkImage` to see how to use the function.

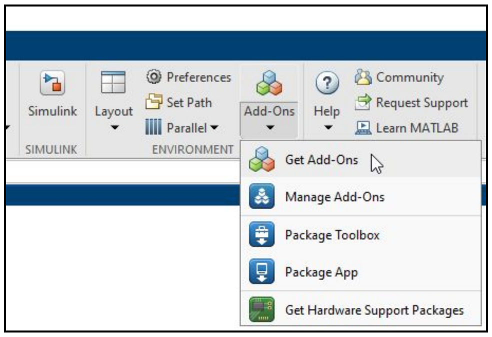
Remove all uses of `imshow()` in the code that you submit. You are encouraged to use it to debug, but you must remove it before submission. Any code that uses a function that displays the image in a window, such as `imshow()` or `image()`, will automatically receive zero points.

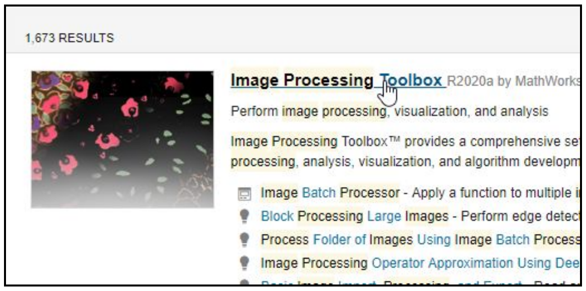
All output images should be saved as `.png` images.

In addition, you may not have the image processing toolbox, which is necessary for some functions in the homework. You can add this to your matlab by the following process.

1. Under Home>Environment>Add-ons, select Get Add-ons
2. Search for "Image processing" and select the "Image Processing Toolbox" result.
3. Select "Sign in to Install", sign in to your mathworks account you created at the beginning of the semester. Then install it.

Helpful Images:




1,673 RESULTS
Image Processing Toolbox R2020a by MathWorks
Perform image processing, visualization, and analysis

Happy coding!
~Homework Team

Function Name: `imageScale`

Inputs:

1. (*char*) A microscope image filename

Outputs:

1. (*double*) The height of the object
2. (*double*) The width of the object

Background:

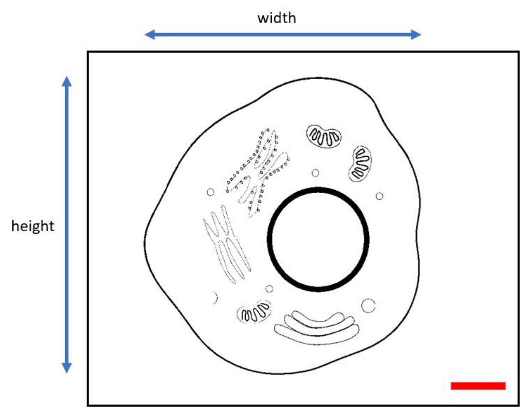
You are sad that you are unable to go see your friends, when you suddenly realize that there are billions of friends all around you! You bust out your microscope and start analyzing samples from around your house. Unfortunately, the scale on your microscope is broken, but MATLAB can help.

Function Description:

You are given an image with a black bordered object on a pure white background. Additionally, there is a pure red scale bar somewhere in the image. The width of this bar is 1 micrometer. Use this to determine how many pixels are in a micrometer. Find the limits of the object (the furthest black pixels to the left, right, top, and bottom of the image), and then calculate the height and width of the object in micrometers. Be sure to include *both* the first and last row or column when calculating dimensions. In other words, if the leftmost red pixel is in column 4 and the rightmost red pixel is in column 8, then the width of the bar is 5 pixels.

Example:

`'pic.png'` →



The outer black box represents the edges of the image. They are NOT part of the image.

```
[height, width] = imageScale('pic.png')
height → 5.42
width → 5.07
```

Notes:

- Round your answers to two decimal places.

Hints:

- The `find()`, `min()`, and `max()` functions may be useful.

Function Name: `pretendVacation`

Inputs:

1. (*char*) filename of an image of you
2. (*char*) filename of a background image

File Outputs:

1. Image file with '_vacay' appended after the background image's filename.

Topics: (*green screen*), (*combining images*)

Background:

You are still bummed out that you had to cancel your Spring Break trip. Your Instagram account desperately needs new content, so you decide to create your own vacation pictures. You turn to MATLAB for help.

Function Description:

You are given a background image with one of the corners colored pure green. Create a mask that finds the pure green corner in the background image.Then, replace the pure green corner with the image of you; however, the image of you will have twice as many rows and columns as the green corner. You will need to resize the image of you to fit into the green corner. Finally, make the image grayscale and write it to the output file.

Notes:

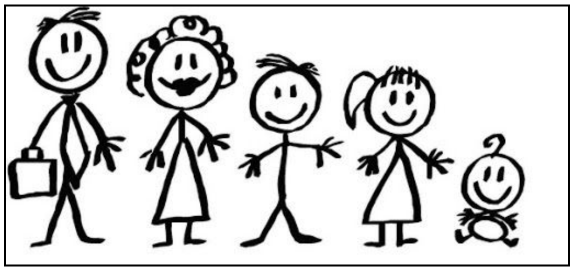
- The pure green color will only be found in the designated area**.**

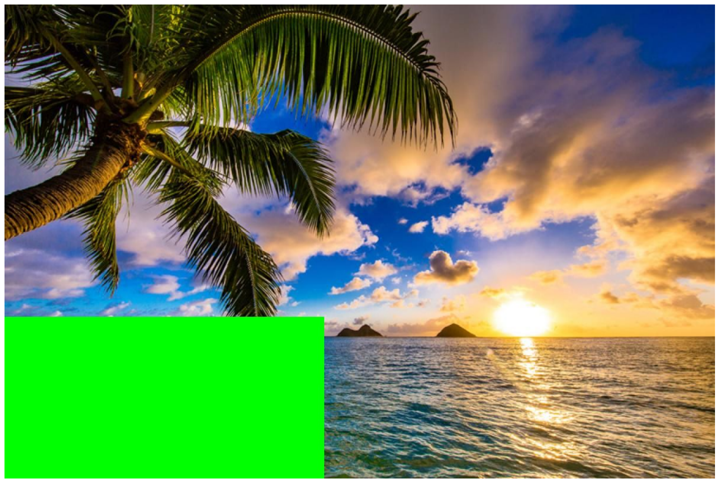
(*continued...*)

Example:

you = 'familyPic.png'

bkgd = 'hawaii.png'

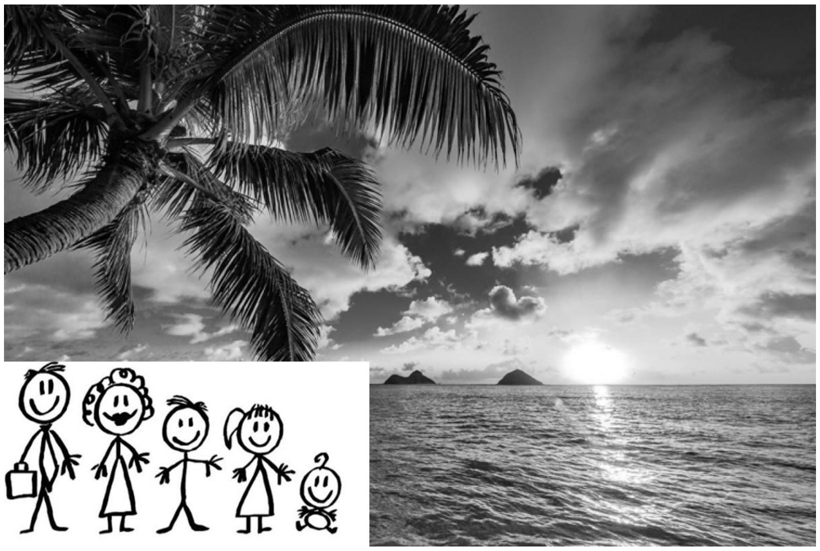




*Border shown for clarity purposes

pretendVacation(you, bkgd) →

'hawaii_vacay.png'



Function Name: `askew`

Inputs:

1. (*char*) filename of a tilted image with text

Outputs:

1. (*double*) the amount of degrees of rotation needed to straighten text
2. (*double*) the height of the text after it has been straightened

Topics: (*image rotation*), (*color masking*)

Background:

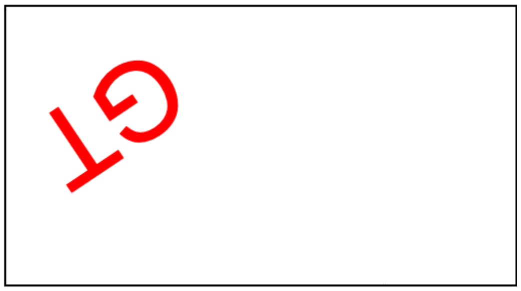
You and a fellow TA were scanning all the tests students just took so that they can be graded on Gradescope. You ended up mistakenly using a faulty scanner, and all the tests you scanned are all tilted a different amount. To fix this mistake you decide to call on your friend MATLAB!

Function Description:

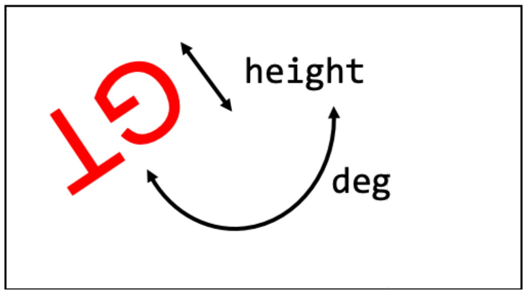
You are given the filename of an image with a white background and red text that is rotated clockwise an unknown number of degrees. Determine the angle of rotation (in degrees) that straightens the text in the image by rotating **counterclockwise** back to an upright position. Also output the height of the image when it is at its straightest. The text in the image is guaranteed tilted clockwise between 1 and 179 degrees.

Example:

```
'askew.png' ->
```



```
[deg,height] = askew('askew.png')
deg → 146
height → 28
```



(*continued...*)

Notes:

- The number of degrees of rotation will always be a whole number.
- Assume the text in the image is never taller than it is wide.

Hints:

- You may need to consider all possible rotations of the input image from 1 degree to 179 degrees.
- How can you find the number of rows containing at least one red pixel for a specific orientation of the image, and what does that tell you about the orientation of the text?

Function Name: `areYouBlind`

Inputs:

1. (*char*) The filename of an image
2. (*double*) A 3x3 convolution matrix

File Outputs:

1. The new image after convolution

Banned Functions:

`conv2`

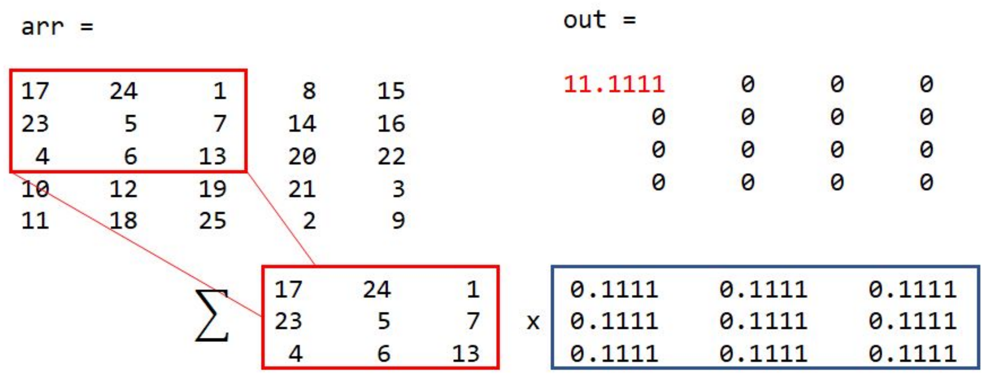
Topics: (*for loops*), (*image processing*)

Background:

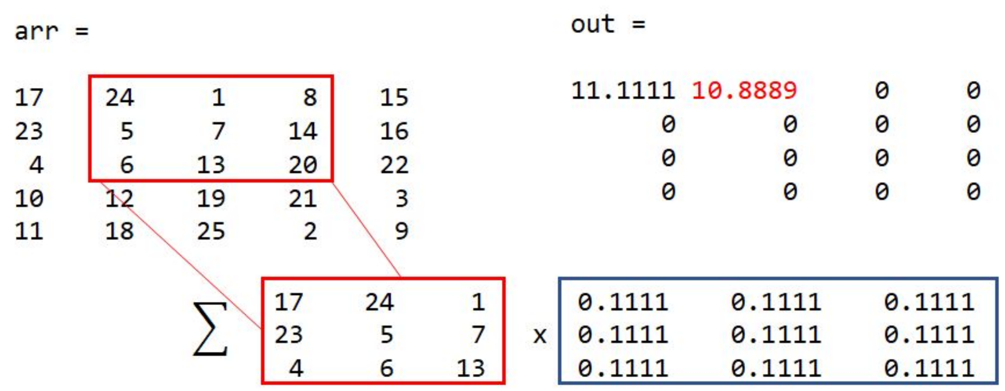
In image processing, a convolution matrix is a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by dividing an image into small segments the size of the convolution matrix and performing an element-wise multiplication of the segment and the matrix. For example, consider the following 5x5 matrix and a 3x3 convolution matrix.

```
arr =  [17    24     1     8    15          c = [0.1111    0.1111    0.1111
        23     5     7    14    16               0.1111    0.1111    0.1111
         4     6    13    20    22               0.1111    0.1111    0.1111]
        10    12    19    21     3
        11    18    25     2     9]
```

The convolution matrix above will replace each value in the array with the average of itself and its neighbors.



(*continued...*)

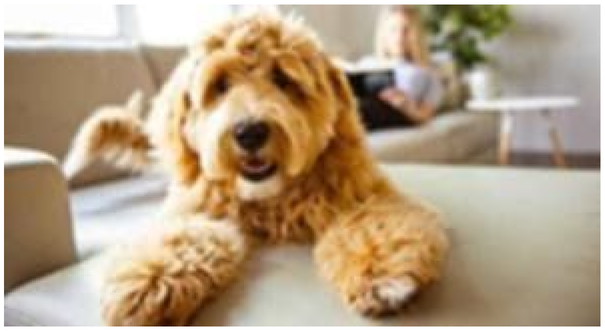


```
arr =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
out =

   11.1111   10.8889         0         0
         0         0         0         0
         0         0         0         0
         0         0         0         0
```

$$\sum \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \end{bmatrix} \times \begin{bmatrix} 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \end{bmatrix}$$

The result of this process on an image is blurring! For example, if we use the same convolution matrix on the image to the left, we get the image to the right.



Function Description:

Write a function that takes in an image filename and a convolution matrix. Perform the convolution, and write the modified image to a new file.

- First, read in the image and **cast it to class double**. You will not be able to perform the convolution correctly if you do not cast your image.
- Next, find the dimensions of the image and create an array to hold your output. You will need to make a second array to hold your output so that you don't corrupt the original image values while performing your convolution.
- Loop over every row, column, and layer of the image. You should omit the first and last row and column, since the edges of the image do not have enough neighboring pixels to perform the convolution.
- Index out the 3x3 segment of your image corresponding to your iterating variables. Multiply the segment by your convolution matrix (element-wise) and sum the result. Place the resulting value into your output array.
- When your loop finishes, cast the output array back to uint8 and write it to a file called `'new_<original_filename>.png'`.

Notes:

- Your output array will need to have two fewer rows and columns than the input image (see bullet point 3).

Function Name: `crimeScene`

Inputs:

1. (*char*) The filename of the image file.

File Outputs:

1. Image file with '_edged' appended after the filename.

Banned Functions:

`edge`

Background:

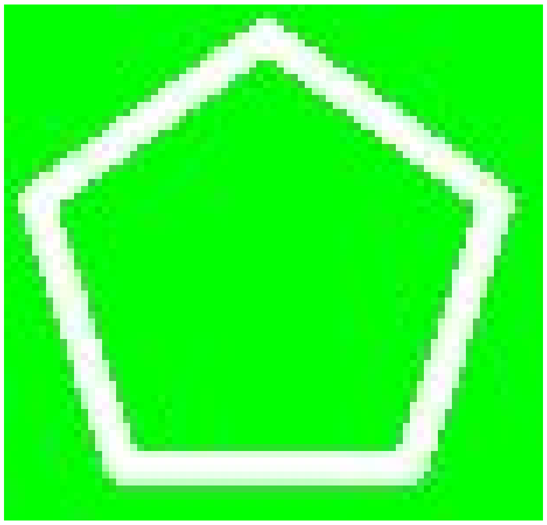
Oh no! There has been a giant robbery at the museum!. People have been found unconscious and a lot of stuff is missing. Your job is to tape the boundaries of the unconscious bodies as well as the missing items. You see that the museum surfaces are dusty, but there is no dust where the items initially were. You have taken some pictures and have decided to use matlab to find the boundaries that you need to tape.

Function Description:

Given a pure white image upon a pure green background, make a function that marks the boundaries of the white image with red color. The boundary of the white image consists of all the green-colored pixels that are adjacent to the white ones, that is, the green pixels that share an edge with the white pixels.

Example:

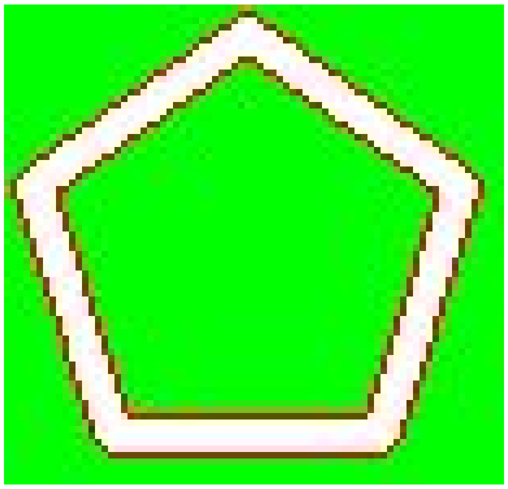
`photo = 'aritfact.png'`



`artifact.png`

(*continued...*)

`crimeScene(photo)`



`artifact_edged.png`

(Image clarity is reduced due to high magnification)

Notes:

- Only the green pixels that share an edge with the white pixels form the boundary of the image

Hints:

- Think about shifting the image slightly in multiple directions and how it can be used to obtain the edges.

Extra Credit

Function Name: `bacteriaCulture`

Inputs:

1. (*char*) File name of the image with circles
2. (double) Red value of circles
3. (double) Green value of circles
4. (double) Blue value of circles

Outputs:

1. (*double*) Number of circles on the image

Topics: (*reading images*)

Background:

As a biologist, you join a research lab, but your faculty mentor puts you in charge of counting the number of bacteria cells in images. Because there are thousands of bacteria cells in an image, you quickly realize that it's not possible to count them all so you quickly turn to your friend MATLAB.

Function Description:

Given an image with multiple circles, output the number of circles in the image. Every circle is guaranteed to be the same area (pi*r^2). The circles will be colored based on red value, green value, and blue value inputs.

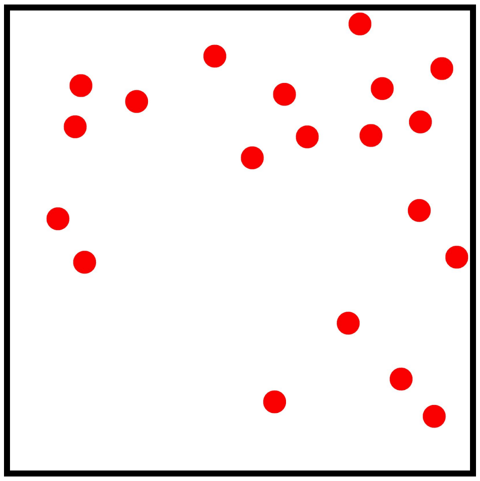
Notes:

- All the circles will be the same area (pi*r^2)

(*continued...*)

Example:

```
    imName = 'bacteria.png'
```



bacteria.png

```
rVal=250
gVal=0
bVal=0

numCircles=bacteriaCulture(imName,rVal,gVal,bVal)
numCircles → 20
```