Spring 2018

Lab 6 - if/elseif/else, switch, conditionals

Objectives: To learn how to use MATLAB logic, in particular:

- How an if and switch statement are constructed
- How to add logical flow to your script

In most of your programming endeavors you will be required to process certain commands if a condition is met. The test you'll likey use most often is an if statement. The basic structure of an if/elseif/else statement is as follows:

```
if (condition 1)
    statement 1 ...
    statement 2 ...
    ...
elseif (condition 2)
    statement 3 ...
    statement 4 ...
    ...
else
    statement 5 ...
    statement 6 ...
    ...
end
```

A switch statement is similar to an if statement with one major difference: an if statement carries through all checks before terminating whereas a switch terminates once a condition is met. The structure of a switch statement is as follows:

```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
    :
   otherwise
      statements
end
```

Here are other functions you may find useful in this lab:

```
imread()
[i j] = size()
length()
```

1. **Boot It Up**
   (a) Start up MATLAB
2. **Warm Up**
   (a) If you want some experience with the coding, try these to get the hang of what you'll need to use in the problems below (not required, but some people like this...).
   - Enter this for loop in the command window (to enter a new line in the command window, press 'shift' + 'enter'):

```
>> r = rand;
if r < .1
  fprintf('%.4f < .1\n',r)
elseif r > .4 & r < .5
  fprintf('0.4 < %.4f < 0.5\n',r)
elseif r >.9
  fprintf('%.4f > .9\n',r)
else
  fprintf('%.4f does not fall within our restricted values.\n')
end
```

and

```
>> x = magic(6);
[a,b] = size(x);
for i=1:a
  for j=1:b
      if(i == j)
          x(i,j) = 100000;
      end
  end
end
```

and

```
>> color = input('Name a color: ');
switch lower(color)
  case {'red', 'light red', 'rose'}
      disp('color is red')
  case 'blue'
      disp('color is blue')
  case 'white'
      disp('color is white')
  otherwise
      disp('Unknown color.')
end
```

3. **Logic**

(a) Create a plotting tool that accepts a vector or matrix from the user and asks which type of plot to generate (offer: plot, bar, pie, polar, mesh, meshc, surface). You will need to determine whether the type of plot requested works for the data provided (a vector should be plotted in plot, bar, pie or polar; a matrix should be plotted with mesh, meshc, surface). Use a switch statement to cycle through the plot options. Name your script *plotTool.m*

(b) Write a script entitled *hiddenMessage.m* that loads an image (provided on Canvas) and accepts a string from the user to embed in the image. Take the string, convert it to binary and place the information in the image. Display the image to the user with the hidden message. You will need to plan your script and likely use the following structure:

- Ask the user for the image filename.
- Read the image into MATLAB using *imread*.
- Search the image for existing values equal to zero and replace them with ones (this will prevent having binary conversion problems later on).
- Ask the user for the message to embed.
- Recall that a string is a vector. Walk through each string value and convert the character to its binary equivalent (use a switch statement here). You will want to post these new values into another zeros matrix that is the same size as the image. Put each letter in the string on a new row. This is most likely the most difficult part...try and figure this out on your own before asking the TA for assistance.

Hint: The indexing may look like c(s,1:8) when building your blank matrix. You may also need this:

a	01100001	n	01101110
b	01100010	o	01101111
c	01100011	p	01110000
d	01100100	q	01110001
e	01100101	r	01110010
f	01100110	s	01110011
g	01100111	t	01110100
h	01101000	u	01110101
i	01101001	v	01110110
j	01101010	w	01110111
k	01101011	x	01111000
l	01101100	y	01111001
m	01101101	z	01111010

- Next, scan your new matrix for values of 1 and set the corresponding points in the image to zero (black). Remember your image file is 3 layers deep: RGB (1st = red; 2nd = green; 3rd = blue).
- Finally, display your updated image using *image*.

Once you've gotten you final image, zoom into the upper left corner and locate your message. Using pencil and paper, decipher your message then show your zoomed image and decoded message to the TA.

(c) Once the TA has checked your files (*plotTool.m* and *hiddenMessage.m*) you are free to go. Do not forget to upload your files to Canvas! Log off of your machine before you leave the lab.