In this homework, you will review logical indexing in MATLAB, a very important technique used to efficiently index a matrix or vector. You will also have a chance to work with functions in a separate file.

1. **Roll the Dice** In this part you will work with ***imshow*** and logical indexing.
   - Create 100x100 matrices A, B and C of all ones.
   - In matrix A, set the values of the entries a_{ij} equal to zero if $\sqrt{(i-25)^2+(j-75)^2} < 10$ *or* $\sqrt{(i-75)^2+(j-25)^2} < 10$. (**Hint:** Use ***meshgrid*** to create the indices, then logical indexing on A using |, & or ~).
   - In matrix B, set the values of the entries b_{ij} equal to zero if $\sqrt{(i-25)^2+(j-25)^2} < 10$ *or* $\sqrt{(i-75)^2+(j-75)^2} < 10$.
   - In matrix C, set the values of the entries c_{ij} equal to zero if $\sqrt{(i-50)^2+(j-50)^2} > 10$.
   - Now use ***figure*** and ***imshow*** to plot:
     - The complement of C
     - A
     - The next 3 faces of a die (so 3-5) on three separate figures. Use whatever logical operations (&, |, or ~) are necessary to accomplish this.
2. **Fun with find** Write a function to return the value and index of a number in a vector / matrix that is closest to a desired value. The function should be called as $[val, ind] = findClosest(x, desiredValue)$. This function can be accomplished in less than five lines. You will find ***abs***, ***min*** and/or ***find*** useful, **Hint**: You may have some trouble using *min* when x is a matrix. To convert the matrix to a vector, you can use $y = x(:)$. Show that it works by finding the value closest to 3/2 (and index of said value) in *sin(linspace(0,5,100)) + 1.*
3. **Calculus Nostalgia** This problem will reacquaint you with the first derivative test and points of inflection.
   - Write a function, called *signSwitch*, in a separate file which inputs a vector v and outputs a vector with the indices i which represent a sign change in v; i.e. report 15 if the sign changed in v between index 14 and index 15. Do not consider going from positive or negative to zero. We could loop through and check this condition at every point - don't do that. Instead think of a way to use logical indexing: One suggestion is to write conditions on the vector and some kind of shifted version of itself. Beware however, when you do this you will have non-overlapping points. It is up to you to figure out what to with them. This will be a local function (see the documentation if you forgot what this means).
   - Now we will write the main function, call it whatever you'd like, which will perform all the analysis. The function will input two vectors representing the x and y coordinates of the graph of the function and will return a vector with the approximate indices of the local extrema and a vector with the approximate indices of the points of inflection.
   - First have the function take the first derivative (approximately, see homework 2). Then use your local function to apply the first derivative test to see where the approximate local minima and maxima are.

- Next, have the function take the second derivative. Then use your local function again to find approximately where the points of inflection are.
- Finally, have the function plot all this information using ***figure*** then *plot(x,y,xminmax,yminmax,'ko',xpoi,ypoi,'r*')*. This will plot the function then plot black circles on the local minima and maxima and red stars on the points of inflection.
- Apply your function to $x^5 - 8x^3 + 10x + 6$ sampled at 10000 points on [-3,3].