Matlab Assignment

1. **Scalar variables.** Make the following variables
   a. $a = 10$
   b. $b = 2.5 \times 10^{23}$
   c. $c = 2 + 3i$, where i is the square root of -1
   d. $d = e^{j2\pi/3}$, where j is the square root of -1 and e is Euler's number[1] (use **exp**, **pi**)

2. **Vector variables.** Make the following variables
   a. $aVec = \begin{bmatrix} 3.14 & 15 & 9 & 26 \end{bmatrix}$
   b. $bVec = \begin{bmatrix} 2.71 \\ 8 \\ 28 \\ 182 \end{bmatrix}$
   c. $cVec = \begin{bmatrix} 5 & 4.8 & \cdots & -4.8 & -5 \end{bmatrix}$ (all the numbers from 5 to -5 in increments of -0.2)
   d. $dVec = \begin{bmatrix} 10^{0} & 10^{0.01} & \cdots & 10^{0.99} & 10^{1} \end{bmatrix}$ (Logarithmically spaced numbers between 1 and 10, use **logspace**, make sure you get the length right!)
   e. $eVec = Hello$ ($eVec$ is a string, which is a vector of characters)

3. **Matrix variables.** Make the following variables
   a. $aMat = \begin{bmatrix} 2 & \cdots & 2 \\ \vdots & \ddots & \vdots \\ 2 & \cdots & 2 \end{bmatrix}$ a 9x9 matrix full of 2's (use **ones** or **zeros**)
   b. $bMat = \begin{bmatrix} 1 & 0 & \cdots & & 0 \\ 0 & \ddots & 0 & \ddots & \\ \vdots & 0 & 5 & 0 & \vdots \\ & \ddots & 0 & \ddots & 0 \\ 0 & & \cdots & 0 & 1 \end{bmatrix}$ a 9x9 matrix of all zeros, but with the values $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$ on the main diagonal (use **zeros**, **diag**).
   c. $cMat = \begin{bmatrix} 1 & 11 & \cdots & 91 \\ 2 & 12 & \ddots & 92 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 20 & \cdots & 100 \end{bmatrix}$ a 10x10 matrix where the vector 1:100 runs down the columns (use **reshape**).

d. $dMat = \begin{bmatrix} NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \\ NaN & NaN & NaN & NaN \end{bmatrix}$ a 3x4 NaN matrix (use **nan**)

e. $eMat = \begin{bmatrix} 13 & -1 & 5 \\ -22 & 10 & -87 \end{bmatrix}$

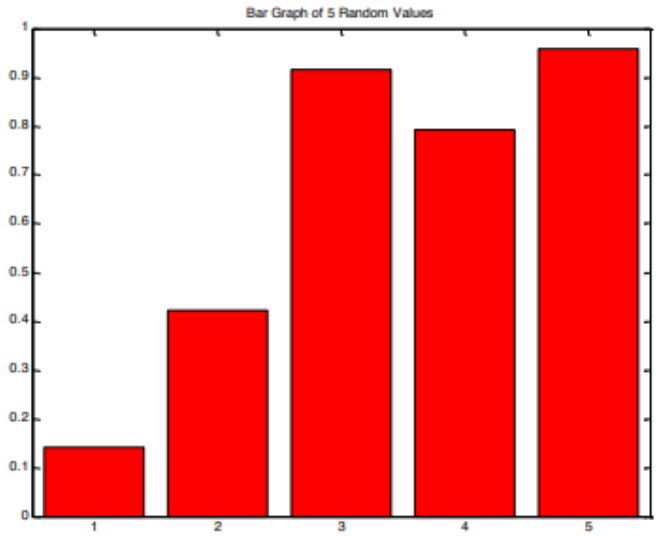
f. Make $fMat$ be a 5x3 matrix of random integers with values on the range -3 to 3 (First use **rand** and **floor** or **ceil.** Now only use **randi**)

4. **Matrix equations.** Using the variables created in 2 and 3, find the values of xMat, yMat and zMat below. Use matrix operators.
   a. $\text{xMat} = (\text{aVec} \cdot \text{bVec}) \cdot \text{aMat}^2$
   b. $\text{yMat} = (\text{bVec} \cdot \text{aVec})$, note that this is *not* the same as $(\text{aVec} \cdot \text{bVec})$
   c. $\text{zMat} = |\text{cMat}|(\text{aMat} \cdot \text{bMat})^{\text{T}}$, where $|cMat|$ is the determinant of $cMat$, and T again indicates the transpose (use **det**).

5. **Common functions and indexing.**
   a. Make $cSum$ the column-wise sum of $cMat$. The answer should be a row vector (use **sum**).
   b. Make $eMean$ the mean across the rows of $eMat$. The answer should be a column (use **mean**).
   c. Replace the top row of $eMat$ with $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$.
   d. Make $cSub$ the submatrix of $cMat$ that only contains rows 2 through 9 and columns 2 through 9.
   e. Make the vector $lin = \begin{bmatrix} 1 & 2 & \cdots & 20 \end{bmatrix}$ (the integers from 1 to 20), and then make every even value in it negative to get $lin = \begin{bmatrix} 1 & -2 & 3 & -4 & \cdots & -20 \end{bmatrix}$.
   f. Make r a 1x5 vector using **rand**. Find the elements that have values <0.5 and set those values to 0 (use **find**).

6. **Semilog plot**. Over the past 6 years, the number of students in 6.057 has been 15, 25, 55, 115, 144, 242. Class size seems like it's growing exponentially. To verify this, plot these values on a plot with a log y scale and label it (**semilogy**, **xlabel**, **ylabel**, **title**). Use magenta square symbols of marker size 10 and line width 4, and no line connecting them. You may have to change the x limits to see all 6 symbols (**xlim**). If the relationship really is exponential, it will look linear on a log plot.

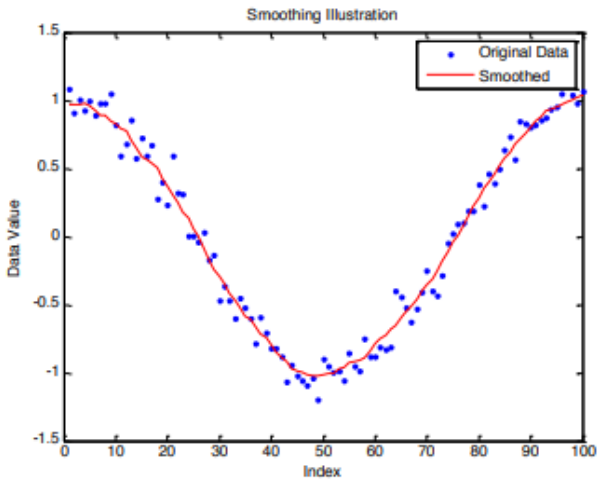
7. **Bar graph**. Make a vector of 5 random values and plot them on a bar graph using red bars, something like the figure below.



8. **Fun with find**. Write a function to return the index of the value that is nearest to a desired value. The function declaration should be: `ind=findNearest(x, desiredVal)`. `x` is a vector or matrix of values, and `desiredVal` is the scalar value you want to find. Do not assume that `desiredVal` exists in `x`, rather find the value that is closest to `desiredVal`. If multiple values (entries) have the same distance from `desiredVal`, return all of their indices. Test your function to make sure it works on a few vectors and matrices. Useful functions are **abs**, **min**, and **find**. **Hint:** You may have some trouble using `min` when `x` is a matrix. To convert a matrix Q into a vector you can do something like `y=Q(:)`. Then, doing `m=min(y)` will give you the minimum value in `Q`. To find where this minimum occurs in `Q`, do `ind=find(Q==m);`

9. **Loops and flow control**. Make function called `loopTest(N)` that loops through the values 1 through N and for each number n it should display 'n is divisible by 2', 'n is divisible by 3', 'n is divisible by 2 AND 3' or 'n is NOT divisible by 2 or 3'. Use a **for** loop, the function **mod** or **rem** to figure out if a number is divisible by 2 or 3, and **num2str** to convert each number to a string for displaying. You can use any combination of **if**, **else**, and **elseif**.

10. **Smoothing filter.** Although it is a really useful software, MATLAB doesn't contain an easy to use smoothing filter[1]! Write a function with the declaration: `smoothed=rectFilt(x,width)` The filter should take a vector of noisy data (`x`) and smooth it by doing a symmetric moving average with a window of the specified `width`. For example if `width=5`, then `smoothed(n)` should equal `mean(x(n-2:n+2))`. Note that you may have problems around the edges: when `n<3` and `n>length(x)-2`.
    a. The lengths of `x` and `smoothed` should be equal.
    b. For symmetry to work, make sure that `width` is odd. If it isn't, increase it by 1 to make it odd and display a warning, but still do the smoothing.
    c. Make sure you correct edge effects so that the smoothed function doesn't deviate from the original at the start or the end[2]. Also make sure you don't have any horizontal offset between the smoothed function and the original (since we're using a symmetric moving average, the smoothed values should lie on top of the original data).
    d. You can do this using a loop and **mean** (which should be easy but may be slow), or more efficiently by using **conv** (if you are familiar with convolution).
    e. Load the mat file called noisyData.mat. It contains a variable `x` which is a noisy line. Plot the noisy data as well as your smoothed version, like below (a width of 11 was used):



[1] It might be better to say "at least we are not aware of such a function!"

[2] Hint: max(1,someNumber) is guaranteed to be at least 1; min(length(x),someNumber) is guaranteed to be at most length(x).