# COMS 3101 - Fall 2013
# Final Project

## Project Description:

1. Your final project consists of two parts: 1-2 page write-up (pdf format) describing your problem, and your code (implementation).
2. Please see submission instructions and due dates on the course website.
3. You may choose to work on any problem you wish. It can be an original problem, or a well known one. You can find some suggestions below.
4. The amount of work you should do on your project (the implementation part) is roughly equivalent to the work you did on the last two HWs (#3,4).
5. The write-up should include three sections:
    a. Problem summary: a detailed description of your problem.
    b. Solution approach: your approach for solving the problem.
    c. MATLAB functionality: a list of functions/structures/aspects of MATLAB you intend to use. It is sufficient to mention only the major/interesting components.
6. If you problem requires any input files/data or creates data/plots/figures, you should include these in your submission along with your code.
7. You may use code from outside sources (see link on the website) as long as you acknowledge this clearly in your write-up.
8. Make sure to document your code properly.

## Project Ideas:

This is a short list of ideas you can consider for your project:

1. *Analyzing a well-known problem*: consider the Collatz conjecture we discussed in class, generate some plots showing the behavior of the sequence. Suggest a proof based on these plots for some

particular cases. Obviously part of this was already done in the optional part of HWs 1-3. However you could choose any well-known problem and do something along the same lines.

2. *Formulating an original optimization problem:* consider the "BWT" problem from class. Find or devise a similar problem. Your write-up should consist of a clear formulation and perhaps some pseudo-code (see lecture slides). You code should implement and solve the optimization problem. You could illustrate the solution using plots, and discuss different cases depending on your input.

3. *Design a small set of routines to generate random data:* it is often useful to be able to generate different types of random data. For example, a chessboard pattern or a cosine curve. You should give the user different options (how many examples, classes, error, type of data), and then generate the appropriate (random) data. Perhaps plot it and save it to a file.

4. *Data analysis:* load some data from a file(s). Do some statistical analysis on the data. Output your conclusions. Perhaps do some curve fitting. For this option, you'd have to supply the data.