CISC106 Spring 2013 Lab07

- This lab an all subsequent labs will be due Thursday at 11:55 PM EDT on Sakai.
- The preparation problems below are to develop your understanding without creating extra work for you or the TA; these problems will not be graded. Be sure to read and understand them - they will help with the problems you must submit for grading
- Review the code examples from your notes in class.
- You may work in pairs on your lab. If you do, **one** of you should be designated to submit the assignment on Sakai. **Both of your names** should appear on code that you develop together[1].
- Whom do you think deducts more points: a happy TA, or a frustrated TA? Make your work easy to read! It isn't just good software engineering, it is good for your grade!
- EVERY MATLAB program/function must include header, doc string that contains a human-readable desciption of what the function does, and must be followed by a good series of tests, as discussed in class. Always test boundaries. Do not test erroneous input (e.g. a factorial function does not need to correctly handle strings).
- EVERY .m file must have a comment line at the very top containing your name(s), lab section, and a brief description of what the file is.

Preparation (do not submit for grading)

1. Run MATLAB and at the prompt in the shell window enter the following:

```
>> row1 = [1 2 3 4]
>> row1 = row1 * 2
```

   Note how this changes $row1$

2. Now enter the following;

```
>> row2 = [1 2 3 4]
>> row1 + row2
```

   Note what that last line yields

3. Now make a $2D$ list, *i.e.* a *matrix*:

```
>> matrix1 = [1 2 3 4; 0 0 0 0; 20 30 40 50]
```

   Enter the following and see what the results are:

[1] If you would like to work with someone but don't know whom, your TA may be able to help connect you to other students looking for lab partners.

```
>> matrix1(1, :)
>> matrix1(:, 2)
```

Note how you can take slices of entire *rows* or *columns* of the matrix.

You can also replace an entire row or column:

```
>> matrix1(2, :) = row1
```

With that last line, $matrix1$ should be

```
[1 2 3 4; 2 4 6 8; 20 30 40 50]
```

.

Problems (to be graded)

1. Reimplement the `celsius_to_far` function from lab03 in MATLAB. You should put this function in a file called `celsius_to_far.m`.
2. Reimplement the `floats` function from lab04 in MATLAB. You should put this function in a file called `floats.m`.
3. Reimplement the `sum_square_difference` function from lab5$\frac{1}{2}$ in MATLAB. You should put this function in a file called `sum_square_difference.m`.
4. Reimplement the `insert_in_order` function from lab06 in MATLAB. You should put this function in a file called `insert_in_order.m`.
5. You probably remember from Algebra class how to solve systems of linear equations manually. As it turns out, you can program a computer to solve certain classes of them for you. Go to `http://en.wikipedia.org/wiki/Gaussian_elimination` and read about *Gaussian Elimination*. Specifically, take note of how you use a row in the matrix to make entries in the other rows 0. There's also a very nice step-by-step presentation available here: `http://www.math.iupui.edu/~momran/m118/integers.html`. Note that, for some pivot row i and another row j in a matrix M, an easier way of telling the computer to do the "criss-cross" step is to simply subtract from row j row i muliplied through by $M(j,i)/M(i,i)$. You should use this information to implement a `reduce` function which takes a matrix and solves it. For example, the following test should pass:[2]

```
assertEqual(
        reduce([2 1 -1 8; -3 -1 2 -11; -2 1 2 -3]),
        [1 0 0 2; 0 1 0 3; 0 0 1 -1])
```

[2]Of course, we're not using a unit testing library in MATLAB, but you can test your reduce informally by calling it at the shell and making sure it gives you back the right matrix.

6. Now that you have reduce working, you should be able to use it to solve the following problem:

   Alice buys three apples, a dozen bananas, and one cantaloupe for $\$2.36$. Bob buys a dozen apples and two cantaloupes for $\$5.26$. Carol buys two bananas and three cantaloupes for $\$2.77$. Figure out how much single pieces of each fruit cost. **Hint:** Set up a system of 3 linear equations. Each equation has the form: $a \cdot x + b \cdot y + c \cdot z = cost$, where a is the number of apples, b is the number of bananas, and c is the number of cantaloupes. That is, encode these 3 linear equations in one matrix for the quantities of fruit purchased, and one column vector for the cost of the purchase. Solve for the 3 variables (x, y, z) which represent the price of each fruit. Your solution should show how much 1 of each fruit costs.

   You should add a comment somewhere in your `reduce.m` which says how much 1 of each fruit costs.

You should submit all of your .m files and any other docs required by your TA on Sakai.