CS 111: Homework 2: Due by 5:00pm Friday, January 18

1. Consider the code for the temperature problem in `Temperature.ipynb`, in the Jan 7 lecture files, especially the routines `make_A()` and `make_b()` that create the matrix A and right-hand side b. Let $k = 100$.

1a. How many elements are there in b?

1b. Considering all possible choices for the boundary temperatures, what is the largest number of elements of b that could possibly be nonzero?

1c. Explain why the rest of the elements of b are zero, no matter what the boundary temperatures are.

2. Experiment with the temperature problem, using different ways of setting the boundary conditions. Make a plot of the most interesting result that you get (in your opinion), and explain how you got it. If you want (not required) you can experiment with `matplotlib` to make a more interesting plot of your result, too. (The CS 111 logo on the course web page was obtained this way in 2010; maybe we can get a new logo this year!)

3. Write the following matrix in the form $A = LU$, where L is a unit lower triangular matrix (that is, a lower triangular matrix with ones on the diagonal) and U is an upper triangular matrix.

$$A = \begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix}$$

4. The following three statements are all **false**. For each one, give a counterexample consisting of a 3-by-3 matrix or matrices, and show the computation that proves that the statement fails.

4a. If P is a permutation matrix and A is any matrix, then $PA = AP$.

4b. If matrix A is nonsingular, then it has a factorization $A = LU$ where L is lower triangular and U is upper triangular.

4c. The product of two symmetric matrices is a symmetric matrix.

5a. Consider the permutation matrix

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Find a 4-element permutation vector `v = np.array(something)` such that, for *every* 4-by-4 matrix A, we have `A[v,:]  == P @ A`. Test your answer by running a few lines of Python, and turn in the result.

5b. For the same P, find a 4-element permutation vector `w = np.array(something)` such that, for *every* 4-by-4 matrix A, we have `A[:,w] == A @ P`. Test your answer and turn in the result.

6. Write `Usolve()`, analogous to `Lsolve()` in the Jan 9 lecture file `LUsolve.ipynb`, to solve an upper triangular system $Ux = y$. Warning: Notice that, unlike in `Lsolve()`, the diagonal elements of U don't have to be equal to one. Test your answer, both by itself and with `LUsolve()`, and turn in the result. Hint: Loops can be run backward in Python, say from $n - 1$ down to 0, by writing

```
for i in reversed(range(n)):
```