Assignment 2
Due: 2:00pm, 4th Sept.
Note: There are different levels of programming in the class. You can choose to work on just two of the four programming problems. Everyone has to answer #1.

1. Describe an application of programming to Cognitive Science (in its broadest sense) that you think the class could spend one week or two working on. Explain the simplest example of what we could do, and what we would learn from it. Provide enough details (including a couple of useful references) to actually do this during the semester.

2. Write an program that solves the Farmyard problem (from the 5th grade). You have a farm with some number pigs and chickens. You ask the user to input the number of heads. Then you ask the user to input the number of legs. Work under the assumptions there are no amputees. You have to figure out how many chickens and how many pigs there are in the farm and printed back to the user. Use a brute force algorithms (exhaustive enumeration).

3. Write a recursive function that can tell whether a word is a palindrome. A palindrome is any sequence of characters which reads the same backwards as it does forwards. The function receives a string and returns True if the string is an anagram and False otherwise. Make sure your program ignores the letter case.

4. Advanced: Write a recursive program that solves the Towers of Hanoi. There are three rods (A, B, and C) and a number of disks (N, 1 is the smallest, topmost and N is the largest, bottommost) of different sizes which can slide onto any rod. To begin, the disks are neatly stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack from A to C, obeying the following rules: (1) Only one disk may be moved at a time; (2) Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod. (3) No disk may be placed on top of a smaller disk. The output of your algorithm should be the series of steps required to move N discs from peg A to peg C, for any number of disks, N.

5. Advanced: Write a recursive function that gives all the permutations of the elements in a list. For example, if the function receive as input [1, 2, 3], then it should return the following output: [[1, 2, 3], [2, 1, 3], [2, 3, 1], [1, 3, 2], [3, 1, 2], [3, 2, 1]]. (Do not use the existing permutations function from the itertools library).
Hand-in procedure

a. Save your code for solutions as a2s2.py, a2s3.py, etc.
b. Time and collaboration info. At the start of each file, in a comment, write down:
   i. The identification of the problem (# Assignment 2, Solution to problem 2).
   ii. Your name (#Name: Steve Jobs).
   iii. The number of hours (roughly) you spent on the problems in that part (#Time: 1h).
   iv. The names of the people you collaborated with (#Collaborators: Steve Wozniak).
c. Send solutions to programming problems as attachments in a single email to
   edizquie@indiana.edu with subject line: Q530 A2.
d. Write answer to #1 as part of body of the email.