# Homework 6

**Due Date**: May 26, 2009 at 5:00PM **Points**: 100

## Written Exercises

1. (*4 points each*) Give the truth table for each of the following Boolean expressions:
   a. not (A and B)
   b. not A and not B
   c. not A and B
2. (*3 points each*) Given the initial statements:
   ```
   import string
   s1 = [ 2, 1, 4, 3 ]
   s2 = [ 'c', 'a', 'b' ]
   ```
   show the values of s1 and s2 after executing each of the following statements. Treat each part independently (that is, assume that s1 and s2 start with their original values each time).
   a. `s1.remove(2)`
   b. `s1.sort()`
   c. `s2.reverse()`
   d. `s1.append([s2.index('b')])`
   e. `s2.pop(s1.pop(2))`
   f. `s2.insert(s1[0], 'd')`

   [*text*, §11.8, Discussion Exercises problem 2, modified]

## Programming Exercises

Remember to turn in your error logs and refinement files.

3. (*30 points*) The Syracuse sequence is generated by starting with a natural number and repeatedly applying the following function until reaching 1:

$$syr(x) = \begin{cases} x/2 & if\ x\ is\ even \\ 3x+1 & if\ x\ is\ odd \end{cases}$$

   For example, the Syracuse sequence starting with 5 is: 5, 16, 8, 4, 2, 1. It is an open question whether the sequence will *always* go to 1 for all possible non-negative starting values. Write a program that gets a starting value from the user and then prints the Syracuse sequence for that starting value. Please turn in the program in the file `syr.py`.
   [*text*, §8.7, Programming Exercises problem 4, modified]

4. (*40 points*) This problem has you write an automatic word transformation program in several steps. We are only going to consider a few words to be transformed, but by extending the dictionary you could transform many more words. For the purposes of this problem, a "word" is a maximal sequence of letters, digits, numbers, and the character "-" (a hyphen).

   a. First, write a function that reads in a dictionary file (which has lines of pairs of words, one pair per line, the words separated by white space) and builds a dictionary. For example, if the file contains the lines

```
hello   hi
goodbye  bye!
```

then the dictionary would be

```
{ "hello" : "hi", "goodbye" : "bye!" }
```

b. Then, write a function to take a line of text (with punctuation) and the dictionary you built in part a. The function returns the line with the words in the dictionary as keys replaced by their values. For example, if the dictionary were the one in the example for part a, and the line were

```
You say goodbye, and I say hello.
```

then the function would return

```
You say bye!, and I say hi.
```

*Warning*: you must preserve spacing and punctuation in the line.

c. Using these functions, write a program that asks the user for the name of a file and the name of a dictionary file (in that order). Each is entered on a separate line. The program outputs the contents of the file with all words in the first column of the dictionary file replaced with the corresponding word in the second column.

Please turn in your program (which will contain the functions in a and b, of course) in the file `trans.py`.

## Extra Credit (Programming)

Remember to turn in your error logs and the refinement file.

5. (*10 points*) Extend the program in problem 4 to handle dictionaries containing word pairs that consist of any strings that do not contain blanks. Please turn in your program in the file `trans2.py`.