CS 150: Intro to OOP, Spring 2012
Assignment 9

Due *via email* by 11:59pm on Thursday, April 19, 2012

For this assignment, you must work individually on these problems. Please note the distinction made in our academic integrity policy between general course material and work which is submitted for this course. We consider the use of the Python language syntax and the cs1graphics package (not needed for this assignment) in the category of general course material, which you may discuss freely. However, you must avoid any discussion of code which is specific to this assignment. You should not receive direct help from others, nor should you share your own source code with others.

1. The __contains__ method of the list class is used to determine whether a given element is contained within a list. Given our knowledge of a list as a sequence of references, we might re-examine the precise behavior of this command. There are 2 possibilities: (A) that it checks whether a given object is itself an element of the list, or (B) that it checks whether the list contains an element that is equivalent to the given parameter.

   Your job is to perform an experiment to conclusively determine which one of the interpretations is the behavior in Python's built-in lists. Please explain the behavior, and give the text of an interpreter session (including Python's responses) that constitutes your "experiment". Then clearly state your conclusion and why your experiment supports it.

2. Write a recursive function `binary(n)` that takes a nonnegative integer n and returns a string of '0' and '1' characters that is the binary representation of the integer.

   Hint: Notice that the *rightmost* bit of the result is equal to str(n % 2), since even numbers end with a 0 and odd numbers end with a 1. The remaining prefix is exactly the binary representation of n//2.

3. The standard notion for comparing two Python lists uses *lexicographical order*. Here, the list with the smaller first element is considered "smaller". However, in the case of the first elements being equal, the second elements are used as a tie-breaker, and so on. If all elements are pairwise equivalent, yet one has additional elements at the end, that list is considered "larger".

   Give an implementation of the method __le__(self, other) for the OurList class which returns True precisely when the original list is less than or equal to the other list, by the convention described above. Please use the setup from the book's OurList class for simplicity.

4. Provide an implementation of the `reverse` function in the OurList class, but think carefully about your (recursive) design. Hint: It's ok to use the other functions already written for the class in the textbook!

5. In a previous homework exercise, you wrote a function `pairSum(data, value)` that returned True if the list `data` contained two *distinct* values whose sum equaled the value, and False otherwise.

   Reimplement this function, but make use of dictionaries for efficiency. Hint: If you start with one number, what would the other have to be? if you find a pair with the correct sum, how can you ensure that the two elements are distinct?