Assignment 1

CSE 473 – Introduction to Artificial Intelligence, University of Washington, Spring 2010.

Due Friday, April 9, at 5:00 PM.

This assignment is intended to help you learn to program in Python with constructs that will be helpful in artificial intelligence. Part I consists of exercises of the form of writing function definitions. Part II is about making a question answering system called *Linneus* a little more intelligent.

Turn in three files: (i) source code (A1P1Functions.py) for your function definitions of Part I, (ii) source code for the new version of Linneus – call it Smarty.py, and (iii) a transcript that shows the example runs for both parts – transcript.pdf. Turn the files in via Catalyst CollectIt.

Part I. Exercises

a. functions
Write a function **isPalindrome(s)** that takes a string as its argument and returns True if the string is a palindrome and False otherwise. Demonstrate it on "MADAMIMADAM" and "Not a palindrome".

b. recursive functions
Write a recursive function **choose(n, k)** that computes binomial coefficients. For example choose(5, 3) = 10. Demonstrate it on (5, 3), (3, 3), (3, 1), and (10, 7).

c. lists
Write a function **combinations(set, k)** that returns a list of the combinations of the given set, taken k at a time. For example, combinations([0, 1, 2], 2) should return
[[0, 1], [0, 2], [1, 2]].
Test it on this example, and also combinations(['H', 'Li', 'Na', 'K', 'Rb'], 3).

d. hashes
Write a function **countWords(s)** that takes a string, breaks it into words using the space character as a separator, and returns the number of occurrences of each word as a list of pairs. For example
countWords("Let's compare red apples and green apples")
should return a list such as this (but the order of the pairs can be different):
[['and', 1],['apples', 2], ['compare', 1], ['green', 1], ["Let's", 1], ['red', 1]]
Your function should use a hash (dictionary) to keep count of the numbers of occurrences of each word.

e. regular expressions
Create a function **generalizeNumbers(s)** that takes a string, and returns a new version of it in which each integer has been replaced by "(some number)". For example
generalizeNumbers("I drank 3 lattes and 12 ounces of water.")
should return "I drank (some number) lattes and (some number) ounces of water."

V1.1 – 29 March 2010, 11:20 AM

You should use a regular expression for non-negative integers. (Don't worry about negative numbers, and you don't have to handle any special forms of numbers such as scientific notation.) Demonstrate your function on this example, and on some other example containing 3 numbers.

Part II. Working with ISA Hierarchies.

Using the Linneus.py program as starter code, add the following features:

A. Detection of reflexive cases of input.
If the user types "A rat is a rat." then the program should do the processing it already does, but it should also say, "That's rather obvious."

B. Detection of symmetric cases of input.
If the user types "A pig is a hog." and later types "A hog is a pig."
then the program should do the processing it already does, but it should also say,
'Then "pig" and "hog" are probably two names for the same thing.'

C. Detection of simple transitive cases of input.
If the user types "A lion is a cat." and then types "A cat is a mammal." and then types "A lion is a mammal." then the program should not add the links between lion and mammal but should say,
"That was already clear from your statements that a lion is a cat and a cat is a mammal."
This should also work if the chain of statements is of any length greater than zero.

D. Optional (instead of A and C)
Detect all forms of redundancy in the user input related to the partial order properties of the ISA relation. Maintain the relation in "reduced form". The relation is in reduced form if it has as few pairs as possible while still supporting the correct answers to questions of the form "Is an X a Y?". Whenever the user inputs redundant information, the program should make a comment about it. For example, if the user inputs "An X is a Y", "An X is a Z", and "A Y is a Z", then the program should report that "Your earlier statement that an X is a Z is now redundant." Then it should remove the redundant link.

E. Optional (instead of B)
Handle the inconsistency of symmetric input in (B) above, but not only detecting and reporting it. Your program should merge the two equivalent nodes of the ISA hierarchy (e.g., “pig” and “hog”) and allow either name to refer to the node in later inputs by keeping a list of aliases associated with the node. The comment would then take a form such as this:
'From now on, I will consider "hog" to be equivalent to "pig". '

F. Optional (instead of all other items in Part II)
Implement D and E in one new version of the program.