Homework 2
Due Tuesday, 02/12/08

1. Please do Problem 5 in section 3.2.4 on page 78 of Bird.

   From Alex

```
import nltk

for speech in nltk.corpus.state_union.files():
   text = nltk.corpus.state_union.words(speech)
   words = nltk.defaultdict(int)

for w in text:
           words[w] += 1

        print speech, ":", words['men'], words['women'], words['people']
```

   Not much of a trend is visible in the data in spite of the suggestions in the problem.

2. Please do Problem 11 in section 3.2.4 on page 78 of Bird.

   From Matt

```
from nltk.utilities import *
import urllib
import re

class CustomURLopener(urllib.FancyURLopener):
version = 'CS181Bot/0.1'

def ghits(word):
""" Inputs: A word to use as a google search query
    Outputs: The (approximate) number of hits for that query.
"""

# Yank the whole page from google
word = word.replace(" ", "+")
searchURL = "http://www.google.com/search?query=" + word
urllib._urlopener = CustomURLopener()
page = urllib.urlopen(searchURL).read()

# Clean up the HTML and normalize whitespace
page = clean_html(page)
page = page.split()
page = ' '.join(page)

# Now find "Results 1-10 of n" and return n.
n = re.findall(r"(?:Results )?1\s?\-\s?10 of (?:about )?(\d+(?:\,\d+)*)",
page, re.I)
if len(n) < 1:
   print page
   print "Could not find the number of hits for the query '%s'" % word
   return -1
else:
   hits = n[0].replace(",", "")
   return int(hits)


# Default test
print "%d" % ghits("test")
```

Output from default test is 1170000000.

3. Please do Problem 1b (money only) in section 3.4.4 on page 87 of Bird.

```
r'\$\d{1,3}(?:\,\d{3})*(?:\.\d\d)?'
```

4. Please do Problem 3 in section 3.4.4 on page 88 of Bird. Use the text in "austin-persuasion" in the Gutenberg corpus of nltk. Compare the entries for the second 200 words using each of the two stemmers and explain the differences.

   From Erik:

```
import nltk,re,sys

showAll = False  #True to show all stems, False to only show differences

#Create stemmers
porter = nltk.PorterStemmer()
lancaster = nltk.LancasterStemmer()

#Print header
if showAll:
   print "Showing all words and stems"
else:
   print "Showing only differences"
print "%12s %12s %12s" % ("Word", "Porter", "Lancaster")

#Stem each word
for word in nltk.corpus.gutenberg.words('austen-persuasion.txt')[200:400]:
   pword = porter.stem(word).lower()
   lword = lancaster.stem(word).lower()
   if showAll or pword != lword:
      print "%12s %12s %12s" % (word, pword,lword)
```

```
#Both stemmers have strengths and weaknesses, but the porter stemmer looks
#slightly more accurate in general. The Lancaster stemmer appears to have
#trouble with names. Output is as follows:

#Showing only differences
#        Word      Porter   Lancaster
#   Elizabeth   elizabeth     elizabe
#        June        june         jun
#        Anne         ann          an
#       still       still        stil
#        Mary        mari        mary
#   Precisely      precis        prec
#     printer     printer       print
#      Walter      walter        walt
#         his          hi         his
#      family      famili        famy
#       these       these        thes
#       after       after         aft
#        date        date         dat
#        Mary        mari        mary
#       birth       birth         bir
#     Married       marri       marry
#      county      counti      county
#  accurately       accur         acc
#       month       month         mon
#         his          hi         his
#        wife        wife         wif
#     history     histori        hist
#        rise        rise         ris
#     ancient     ancient         ant
#      family      famili        famy
#       usual       usual          us
#   mentioned     mention        ment
#     Dugdale      dugdal        dugd
#      office       offic         off
# parliaments  parliament      parlia
#     loyalty     loyalti       loyal
#     dignity     digniti        dign
#         all         all          al
#       Marys        mari        mary
#     married       marri       marry
```

5. Please do Problem 3.4 on page 39 of JM.

   Picture omitted. Do in two steps, first converting from lexical form to intermediate form. E.g., pin+PastPart ⇒ pinˆed. then go from intermediate to surface form: pinˆed ⇒ pinned.

   Most problems came from assuming there was a memory that could remember the letter to be doubled. Instead you must have a separate state for each last letter read. That way when you hit the separator, you can generate an extra copy of the letter.

6. Please do Problem 3.9 on page 39 of JM.

   The arc from q_5 to q_1 assures that if there is a "sz", "ss", or "sx" preceding the plural morpheme then it will get properly converted to plural. A good example to try is the word "assesses". *I must admit that while I know several words ending in "ss", I don't know any ending in "sz" or "sx".*

7. Please do Problem 3.10 on page 39 of JM.

   The answer to this depends on the cost functions for inserting, deleting, and substituting. If inserting and deleting costs 1 and substituting costs 2, then drive to brief costs 4, while drive to divers costs 3, thus the latter two are closer.

8. Please do Problem 3.12 on page 39 of JM. You may start with my program in file minEditDistance.py in /common/cs/cs181/programs.

   Your output should list the table in a style similar to that shown in Lecture 4 (though you'll have to figure out another way to produce something to take the place of arrows). Also present a list of exactly the changes to be made to get from the start to the finish and its cost. Do your best to make the output as readable as possible.

   Matt's solution, below, provides a very clear table of values and lines up words illustrating the changes in transforming from one word to the next.

```
# characters to indicate directions for trace in table
UP = u"\u2191"
LEFT = u"\u2190"
DIAG = u"\u21B0" #u"\u21B8"

# return cost of replacing fst by snd
def substCost(fst,snd):
    if fst == snd:
        return 0
    else:
        return 2

# Given cost of 1 for insertion or deletion and 2 for substitution
# return dynamic programming table for distance[i,j] as defined in class
# for transforming source to target.
# Minimum cost is distance[n][m] for n = len(target), m = len(source)
def minEditDist(target, source):
    n = len(target)
    m = len(source)
    # m+1 rows, n+1 cols
    distance = [[(0, "   ") for i in range(n+1)] for j in range(m+1)]
    for col in range(1,n+1):
        (tmpdist, trace) = distance[0][col-1]
        distance[0][col] = (tmpdist + 1, LEFT + u"  ")
    for row in range(1,m+1):
        (tmpdist, trace) = distance[row-1][0]
        distance[row][0]= (tmpdist + 1, u"  " + UP)
    for col in range(1,n+1):
        for row in range(1,m+1):
            # Compute distances
            (above, abvTrc) = distance[row-1][col]
            (left, lftTrc) = distance[row][col-1]
            (diag, dgTrc) = distance[row-1][col-1]
            above += 1
            left += 1
            diag += substCost(source[row-1],target[col-1])

            # Find the minimum and mark the trace appropriately
            newDist = min(above, left, diag)
            newTrc = ""
            if left == newDist: newTrc += LEFT
            else: newTrc += " "
            if diag == newDist: newTrc += DIAG
            else: newTrc += " "
            if above == newDist: newTrc += UP
            else: newTrc += " "

            # Update the new distance
            distance[row][col] = (newDist, newTrc)

    return distance

def printDPTable(distances):
    """ Makes the distance DP table legible
    """
    finalString = u""
    for row in distances:
        finalString += u"["
        finalString += u' '.join([u"%s%-2d" % (trc, dist) \
                                  for (dist, trc) in row])
        finalString += u"]\n"
    print finalString

def getAlignment(target, source):
    """ Gets the min edit distance DP table and traces back the arrows that
        it records.  It disambiguates equal paths by the following priority:
        replacement first, deletion second, and addition last.

        Produces a list of aligning commands composed of 's' for substitute,
        'd' for delete, 'i' for insert, and ' ' for do nothing.
    """
    distance = minEditDist(target, source)
    #printDPTable(distance)   # Show the resulting DP table.
    #print

    # When we find an UP arrow, that means delete; LEFT arrow is add;
    # DIAG arrow is substitute or do nothing depending on the difference
    # between the two squares.
    n = len(target)
    m = len(source)
    edits = []
    while n != 0 or m != 0:
        (dist, trc) = distance[m][n]

        # First check for substitutions/do-nothings, since those are
        # highest priority
        if DIAG in trc:
            dgDist = distance[m-1][n-1][0]
            if dist == dgDist:
                edits.append(' ')
            else:
                edits.append('s')
            n -= 1
            m -= 1

        # Next we check for deletions
        elif UP in trc:
            edits.append('d')
            m -= 1

        # And lastly, additions
        elif LEFT in trc:
            edits.append('i')
            n -= 1

        # If there's nothing in the trace, we stop.
        else:
            break

    # Since we worked backwards, the operations list is
    edits.reverse()
    return edits


def showAlignment(target, source):
    """ Uses edits as given by getAlignment to display the operations
        necessary in converting the source string to the target string.
    """
    edits = getAlignment(target, source)
    srclist = list(source)
    tgtlist = list(target)
    for (idx, cmd) in enumerate(edits):
        if cmd == ' ' or cmd == 's': continue
        elif cmd == 'i': srclist.insert(idx, '*')
        elif cmd == 'd': tgtlist.insert(idx, '*')

    output = "".join(srclist) + "\n" + \
             "|" * len(edits) + "\n" + \
             "".join(tgtlist) + "\n" + \
             "".join(edits)
    return output


def testAlign(target, source):
    """ A function for printing a simple test of the alignment functions.
    """
    print "Source: %s\nTarget: %s\nDP Table:\n" % (source, target)
    try: printDPTable(minEditDist(target, source))
    except UnicodeEncodeError:
        print "Unicode unavailable; can't print DP table"
    print
    print showAlignment(target, source)

# Sample output
print "-"*80
testAlign("brief", "drive")
print "-"*80
testAlign("divers", "drive")
print "-"*80
```