Key to HW3

Graphics by Jon Yamato, explanations by Mary Kuhner. (Jon Yamato provided the pencil comments on the homework he typed in and verified each program.)

Mean: 89.9. Median: 91.0. Mean points over HW1-3: 274.9.
#1

RP = Red Panda
GP = Giant Panda
Ra = Racoon
Do = Dog

```
  Rp   GP
  /\   /\  \\
 Ra  Do Ra  Do
  |   |   |   |
  Rp   GP
```

```
  Rp   Ra   Rp   GP
  /\   /\   /\   /\  \\
 Rp  Do Do  Do  Ra  Do
  |   |   |   |   |
  Rp   Ra   Rp   GP
```
#2

RP = Red Panda   Ra = Racoon
GP = Giant Panda  Do = Dog

Score: 7

Score: 6

Score: 7
#3

RedPanda   GiantPanda   Racoon  Dog
RedPanda       0                   2                4          4
GiantPanda     2                   0                2         3
Racoon            4                   2                0          3
Dog                 4                   3                3          0

#4

<table>
<thead>
<tr>
<th></th>
<th>RedPanda</th>
<th>GiantPanda</th>
<th>Racoon</th>
<th>Dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>RedPanda</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>GiantPanda</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Racoon</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Dog</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

#5

-0.75 \times \ln(1-4/3(7/100)) = 0.073
Problems 3-6 explanations:

Problem 3: I expected you to choose the most parsimonious tree found in question 2, root it with dog as the outgroup, and note that this produces a tree in which giant panda and red panda are sibling species and thus most closely related. Quite a few people used a distance approach instead; I'm sorry that the question was ambiguous.

Problem 4: I accepted either raw counts of differences, or percentages. (For practical use percentages would generally be better.)

Problem 6: I took off 1 point if the program would give a Python error on rawdist == 0.75 (because this leads to log(0) which is an error) and 3 points if the program would give a Python error on any distance greater than 0.75. When testing a program, make sure that it handles wrong input correctly as well as right input. Numbers at the boundary between good and bad input are particularly important to check.
import sys
import math

rawdist = float(sys.argv[1])
if rawdist < 0.75 and rawdist > 0.0 :
    newdist = (-3.0/4.0) * math.log(1.0 - (4.0/3.0)* rawdist)
    print newdist
elif rawdist == 0.0 :
    print 0.0
elif rawdist >= 0.75 :
    print "Distance is infinite"
else :
    print "Negative distance not allowed"

#open file for reading
import sys
filename = sys.argv[1]
filehandle = open(filename,"r")

#read all the lines and close the file
lines = filehandle.readlines()
filehandle.close()

#setup a success flag and initialize output message
failed = False
msg = "Ok"

#remove the first line from the list and parse it
line = lines[0]
lines = lines[1:]
firstline = line.split()
#7 cont'd

if len(firstline) != 2:
    failed = True
    msg = "The first line of this type of file contains only 2"
    msg += " numbers, how many sequences and how many"
    msg += " markers."

    #more tests are possible...are the values numbers?
    #                                        are they integers?  etc.

if not failed:
    #is the first value the number of sequences in the file?
    if long(firstline[0]) != len(lines):
        msg = "The first number on the first line must be the"
        msg += " number of sequences. It claims there are "
        msg += firstline[0] + " sequences, but we detected "
        msg += str(len(lines)) + " sequences."
        failed = True

if not failed:
    for line in lines:
        #seperate the name, 10 characters, from the markers
        name = line[0:10]
        sequence = line[10:]

        #now remove whitespace
        sequence = sequence.strip()
        sequence = sequence.expandtabs(1)
        sequence = sequence.replace(" ","

        #is the second value from the first line the number of
        #markers?
        if long(firstline[1]) != len(sequence):
            msg = "The sequence named " + name + " has "
            msg += str(len(sequence)) + " bases but the first line"
            msg += " says it has " + firstline[1] + "."
            break
#7 cont'd
# is the sequence composed of legal characters for DNA?
# first put the entire sequence into lowercase
sequence = sequence.lower()
# now make a temp copy and remove all legal characters
copyseq = sequence
copyseq = copyseq.replace("a"," ")
copyseq = copyseq.replace("c"," ")
copyseq = copyseq.replace("g"," ")
copyseq = copyseq.replace("t"," ")
if not copyseq.isspace() :
    copyseq.replace(" ","")
    msg = "The sequence " + name + " has the following"
    msg += " disallowed characters in it: " + copyseq
    break
print msg

#8
import sys
filehandle = open(sys.argv[1],"r")
nbases = filehandle.readline()
garbage,nbases = nbases.split()
nbases = long(nbases)
seq1 = filehandle.readline()
seq2 = filehandle.readline()
filehandle.close()
seq1 = seq1[10:]
seq2 = seq2[10:]
seq1 = seq1.strip()
seq2 = seq2.strip()
count = 0
for (mrk1,mrk2) in zip(seq1,seq2) :
    if mrk1 != mrk2 :
        count += 1
print "The raw distance is",str(float(count)/float(nbases))
Problems 7-8 explanations:

There were many good way to do these problems; your solution might look totally different and still be excellent.

Some people wrote programs which got the correct answer on the sample data set, but wouldn’t work in general: hard-coding the number of lines, for example. Try to make your program generally useful.

The strings returned from readlines() and readline() have a newline character at the end. This messed up many students’ counting. It is tricky to debug what is going wrong in such a program, as the newline is invisible. You can see it by printing that position in your string: print mystring[40]. Be sure not to “fix” a program whose answer is wrong by 1 by adding or subtracting 1; it is important to find the problem and really fix it. The string function rstrip() can be helpful in getting rid of unwanted newlines; it strips whitespace off the right-hand end of a string.

Problem 7: I took off a point if the program printed OK for files that were not OK. I took off a bit more if the program failed to check every line in the file, or failed to check both number of sequences and sequence length. I didn’t take off points for programs that printed a message for every line, but they would become unpleasant with a big file.

A program of this kind is much more useful if it says ”wrong count of bases on line 8” or ”wrong count of bases for species ’red panda’” rather than ”wrong count of bases.”

Problem 8: The most severe problem encountered here was programs which compared the species names as well as the DNA. Small counting errors in the length of the sequences or their names were also common. Again, be sure you understand a problem before fixing it; don’t apply fixes at random.

I accepted either percent differences or raw differences. I didn’t give full points for counting similarities, though, as a count of similarities really is not a distance.