**CS423 Lab 7: Multiple sequence alignment and trees**

**Authors: Shaun Stice, Caleb Piekstra**

**Write-up (62 points):**

Create a docx or pdf file to complete your writeup of this lab to show that you tested your code. Be sure to include your name(s) and lab number in your file.

1. (10 points) Complete the table of pairwise **global alignment** scores for the five sequences given in MSA.py.

**Table 1: Pairwise global alignment scores**

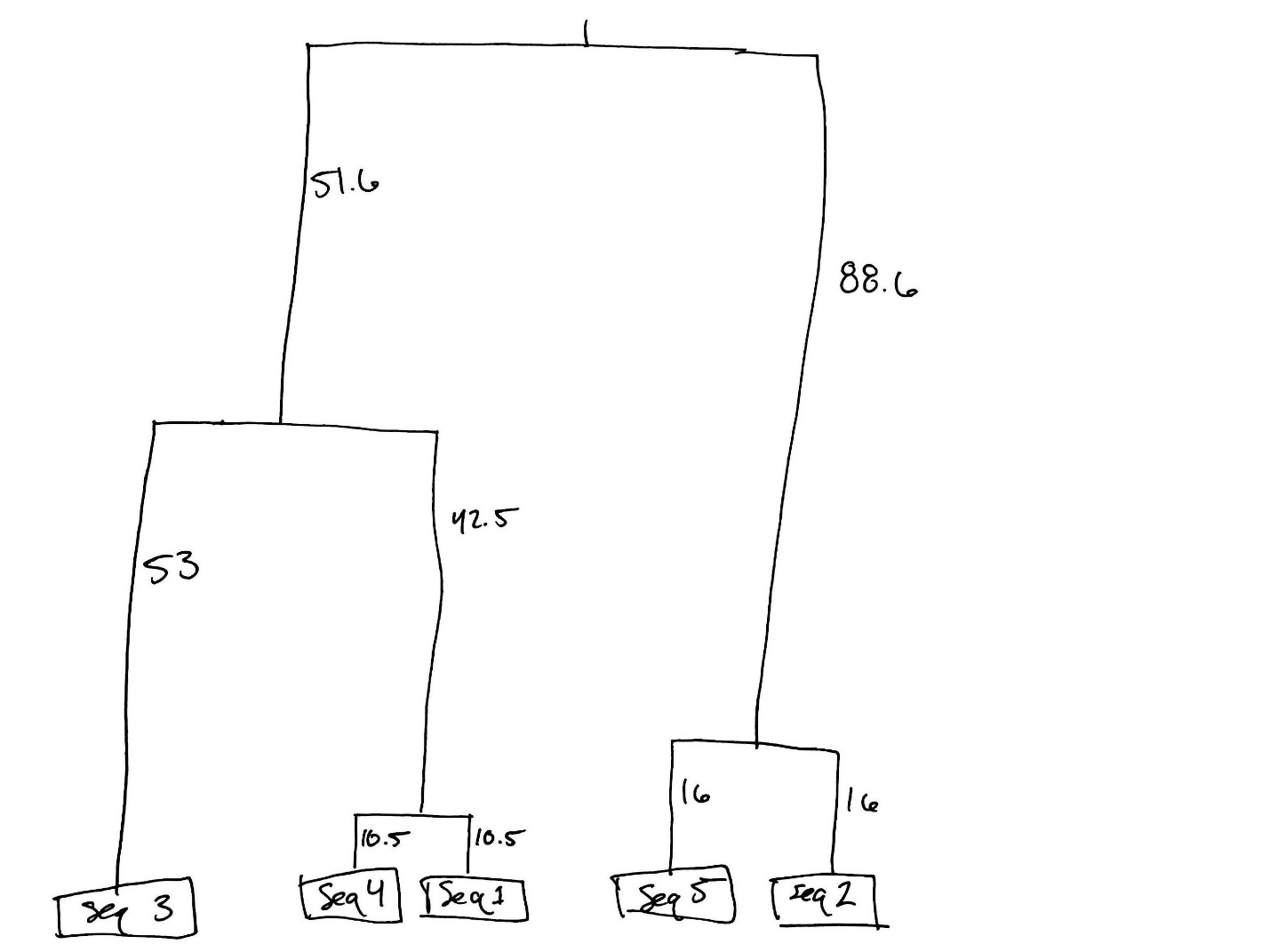
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **seq1** | **seq2** | **seq3** | **seq4** | **seq5** |  |
|  | **-4** | **75** | **177** | **12** | **seq1** |
|  |  | **50** | **2** | **133** | **seq2** |
|  |  |  | **66** | **60** | **seq3** |
|  |  |  |  | **6** | **seq4** |
|  |  |  |  |  | **seq5** |

1. (10 points) Complete the table of pairwise **distance** scores for the five sequences given in MSA.py.

**Table 2: Pairwise distance scores (we rounded these to integers)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **seq1** | **seq2** | **seq3** | **seq4** | **seq5** |  |
|  | **256** | **100** | **21** | **230** | **seq1** |
|  |  | **117** | **261** | **32** | **seq2** |
|  |  |  | **112** | **112** | **seq3** |
|  |  |  |  | **279** | **seq4** |
|  |  |  |  |  | **seq5** |

1. (10 points) Based on **distance** scores in Table 2, use the UPGMA algorithm to draw the guide tree for the five sequences. For this tree, calculate the actual tree branch distances and label the branches with these numbers. Also, try to draw the tree branches to scale, with higher distances having longer lines. (Use visio, MS drawing tools, or a scanned hand-drawn picture and paste the tree into your write-up here.)



1. (12 points) Answer these questions after doing the multiple sequence alignment (MSA) of the ten orthologs of the human gene:
   1. (2 pts) Copy your MSA here.

**See Appendix 1**

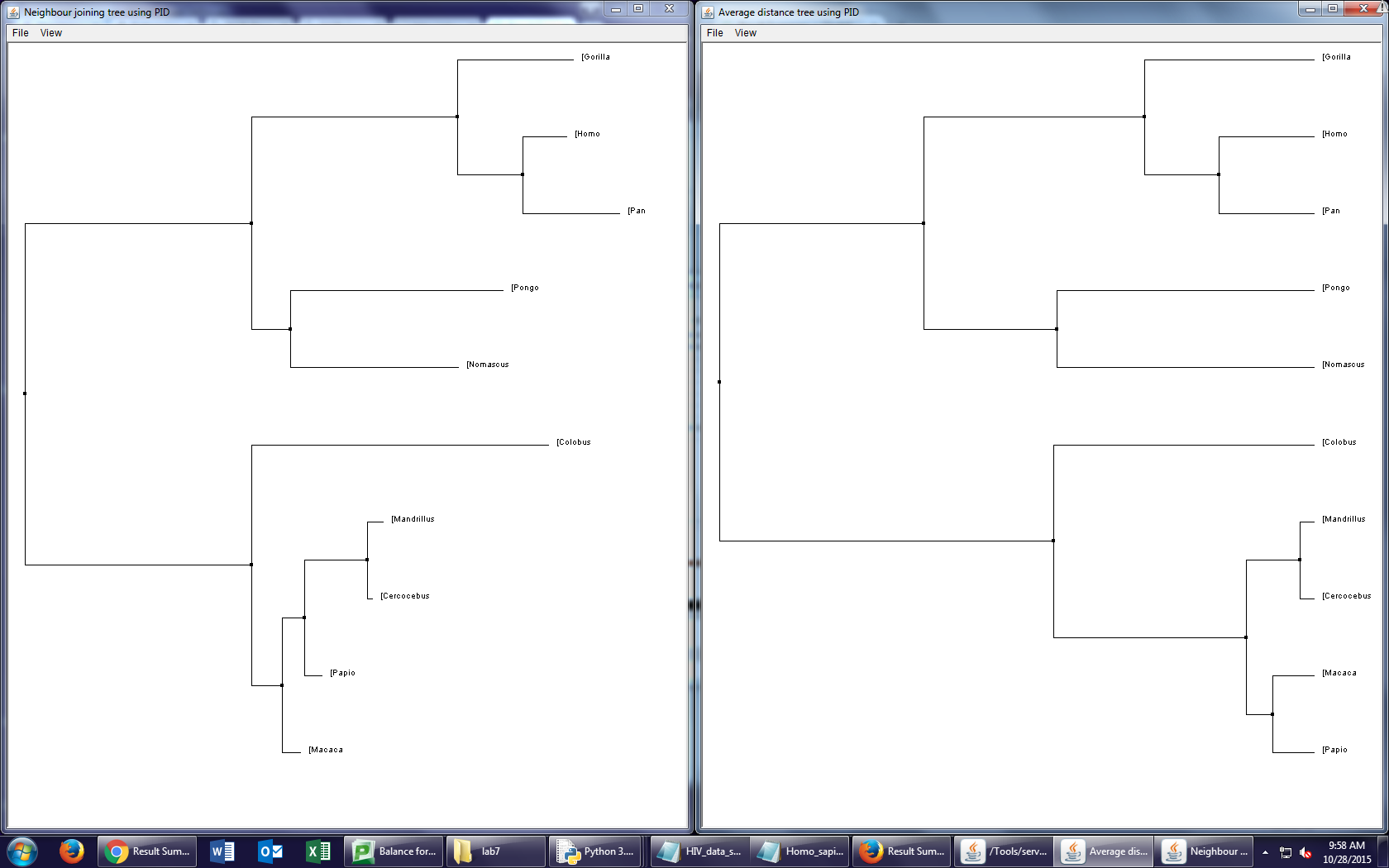
* 1. (2 pts) Are there regions within the protein family that are highly conserved? Where?

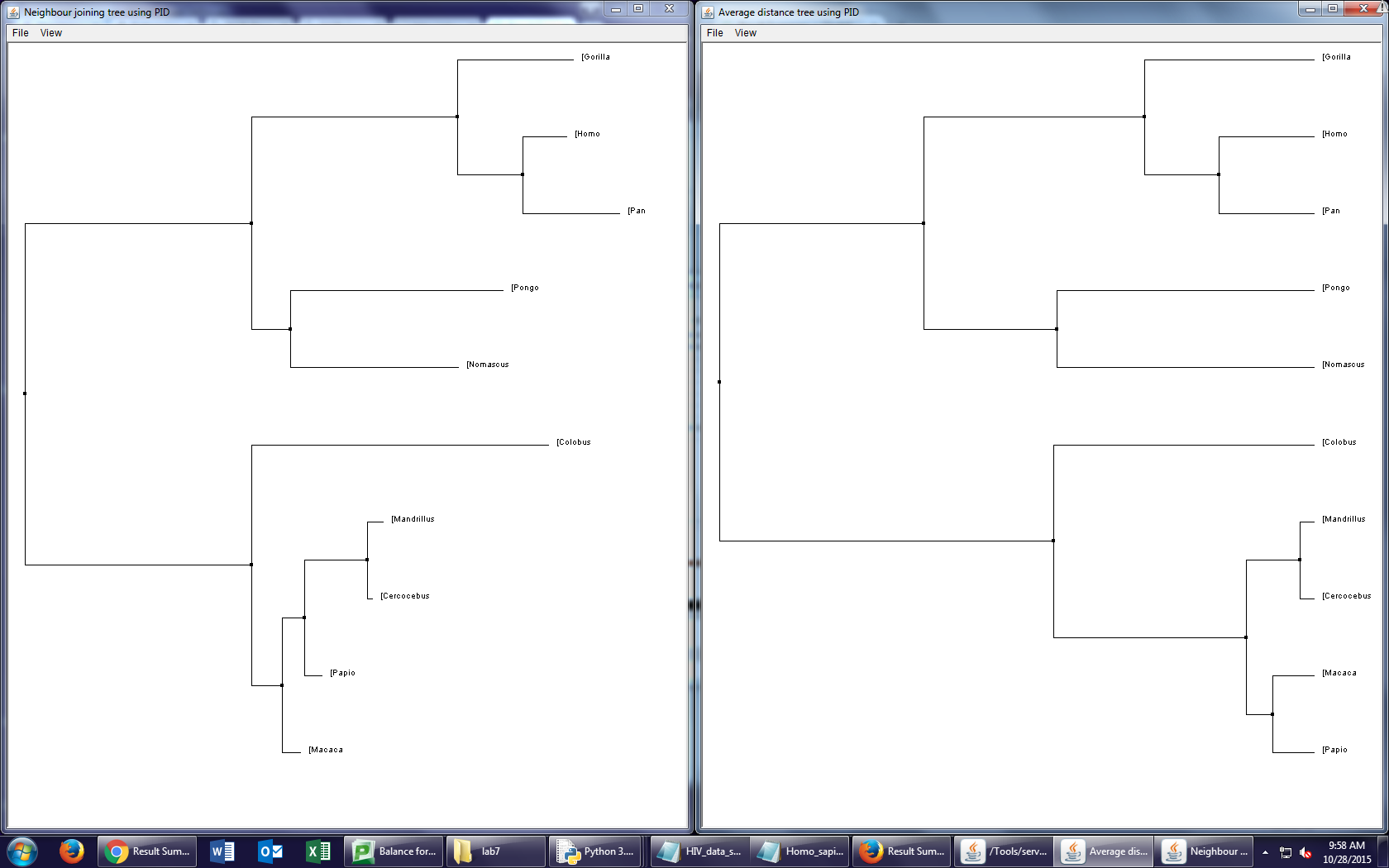
**Yes, the last 60% are highly conserved between the orthologs.**

* 1. (2 pts) Why might some regions of the protein family be more highly conserved?

**These regions are likely more important to the function of the protein and if they were mutated the protein would not function the same. The ortholog genes also have a similar function so we would expect them to be highly conserved.**

* 1. (2 pts) Copy your screen shot of the two trees here.





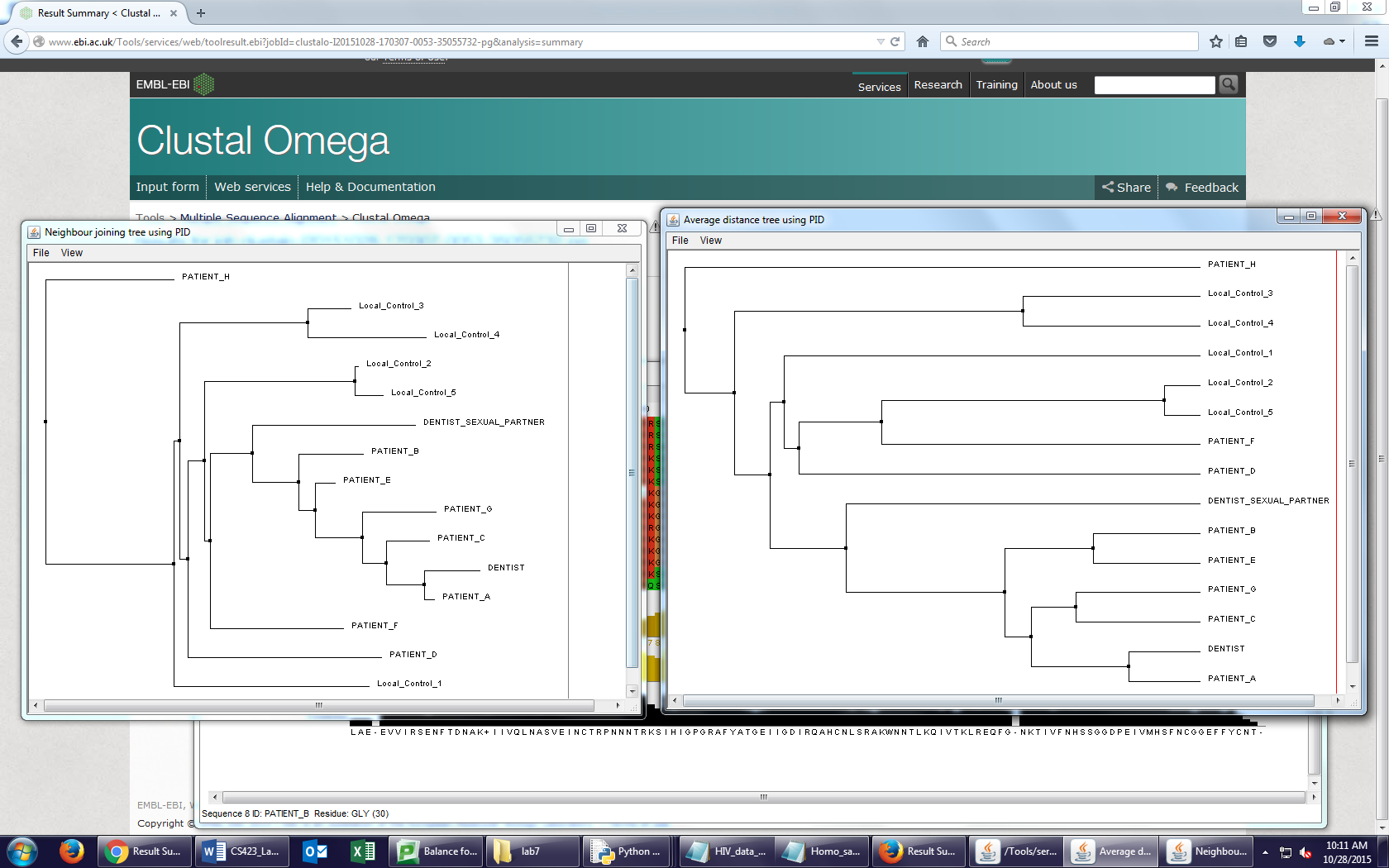
* 1. (2 pts) Which pair of sequences (not necessarily the human gene) are most similar according to the UPGMA algorithm and percent identity? (which two organisms?)

**The Cercocebus and Mandrillus are most similar according to the UPGMA algorithm and percent identity.**

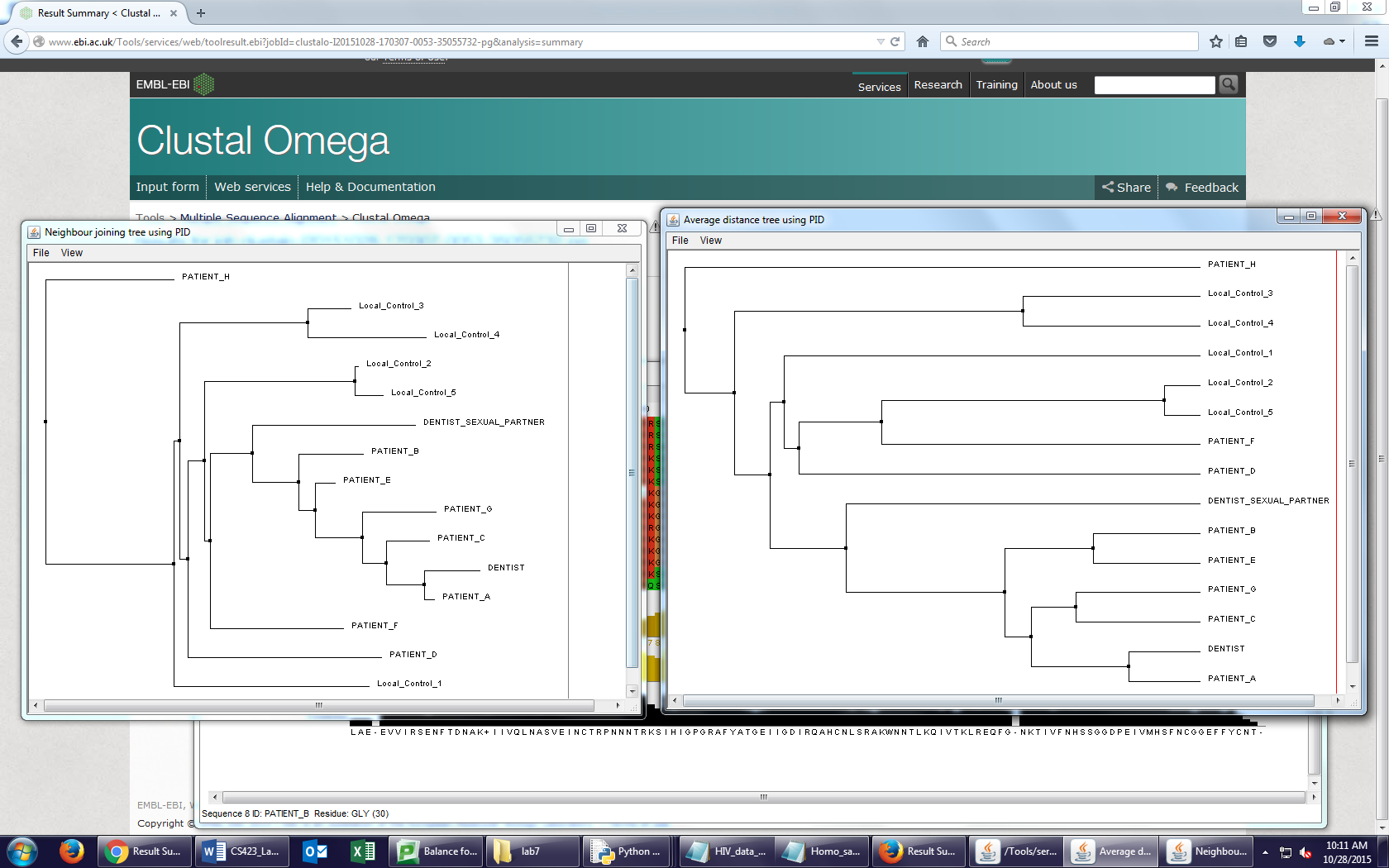
* 1. (2 pts) Which pair of sequences (not necessarily the human gene) is most similar according to the neighbor joining algorithm and percent identity? (which two organisms?)

**The Mandrillus and the Cercocebus are most similar according to the neighbor joining algorithm and percent identity.**

1. (22 points) Answer these questions about the dentist-HIV case:
   1. (1 pt) Paste the phylogenetic tree using average distance here.



* 1. (1 pt) Paste the phylogenetic tree using neighbor joining here.



* 1. (1 pt) Is the tree created using average distance rooted or unrooted?

**The average distance is rooted.**

* 1. (1 pt) Is the tree created using neighbor joining rooted or unrooted?

**The neighbor joining is unrooted.**

* 1. (3 pts) Based on the phylogenetic trees of the HIV sequences, do you think it is likely that the dentist infected any of his eight patients? If so, which patients?

**Yes, PATIENT\_A, PATIENT\_B, PATIENT\_C, PATIENT\_E¸ and PATIENT\_G.**

* 1. (3 pts) Are there any patients that are unlikely to have been infected by the dentist? If so, which patients?

**Yes, PATIENT\_D, PATIENT\_F, and PATIENT\_H.**

* 1. (4 pts) Why was it important for researchers to collect HIV samples from people who had no contact with the dentist and from the dentist’s sexual partner(s)?

**It was important because HIV mutates very quickly, so the gp120 sequences could become quite different in a relatively short amount of time. However, by having the control groups, it is possible to group those infected by the dentist (despite mutations) in one group, leaving the others outside of this group. The control group are known to not be infected by the dentist, and their gp120 sequences of HIV would evolve in a way to differentiate themselves from the dentist gp120 sequences compared to that of the infected patients.**

* 1. (4 pts) Why did the researchers choose the HIV gp120 protein for phylogenetic analysis and not a different protein?

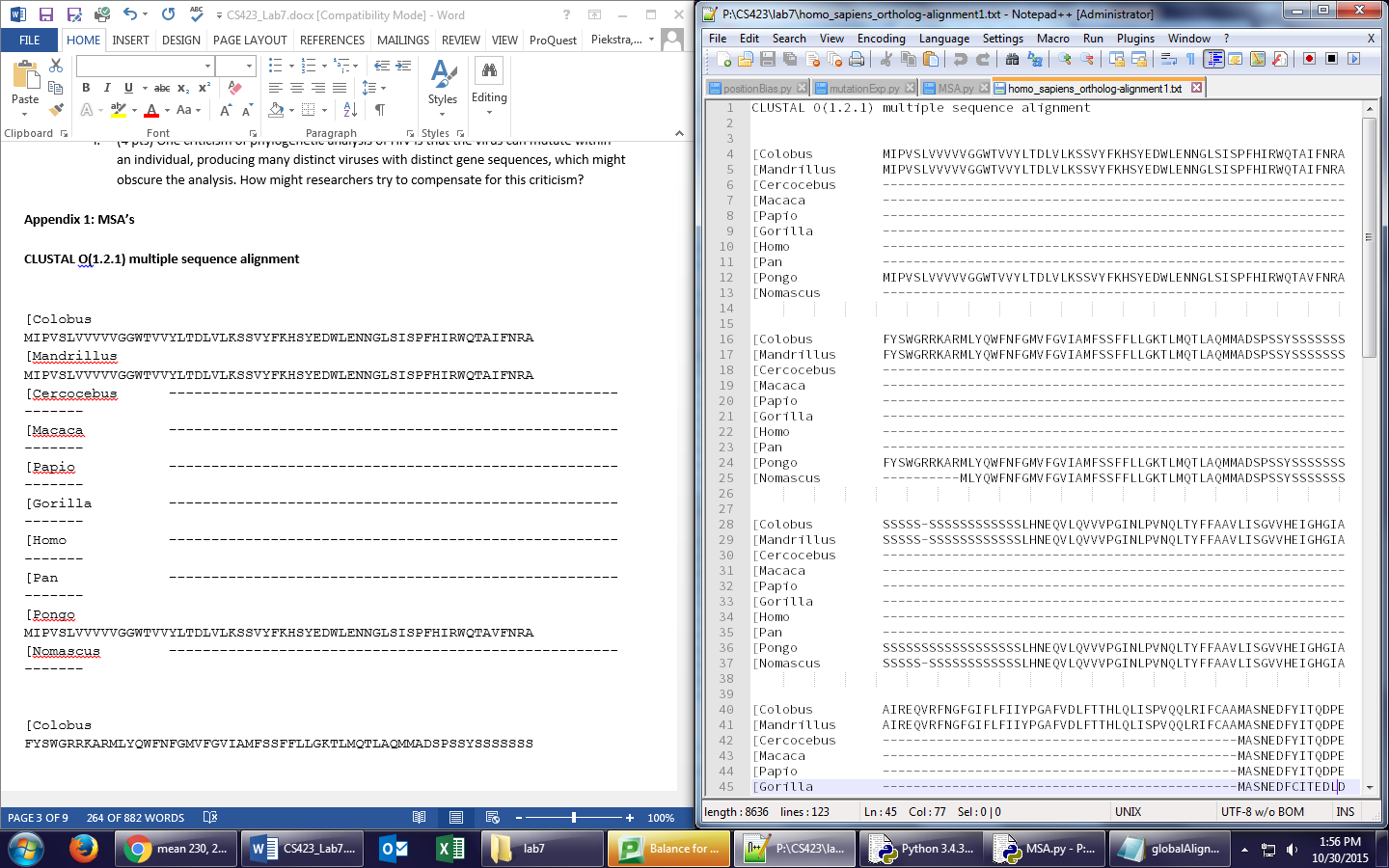
**This protein must have a higher mutation rate than other HIV proteins. The higher mutation rate allows for the differentiation between the groups. If the protein mutated slowly, it could be difficult to link the infection to the dentist. The protein is also conserved enough to allow for a proper comparison. Essentially, the phylogenetic tree becomes more robust with a highly conserved protein that expresses rapid sequence variation over short periods of time.**

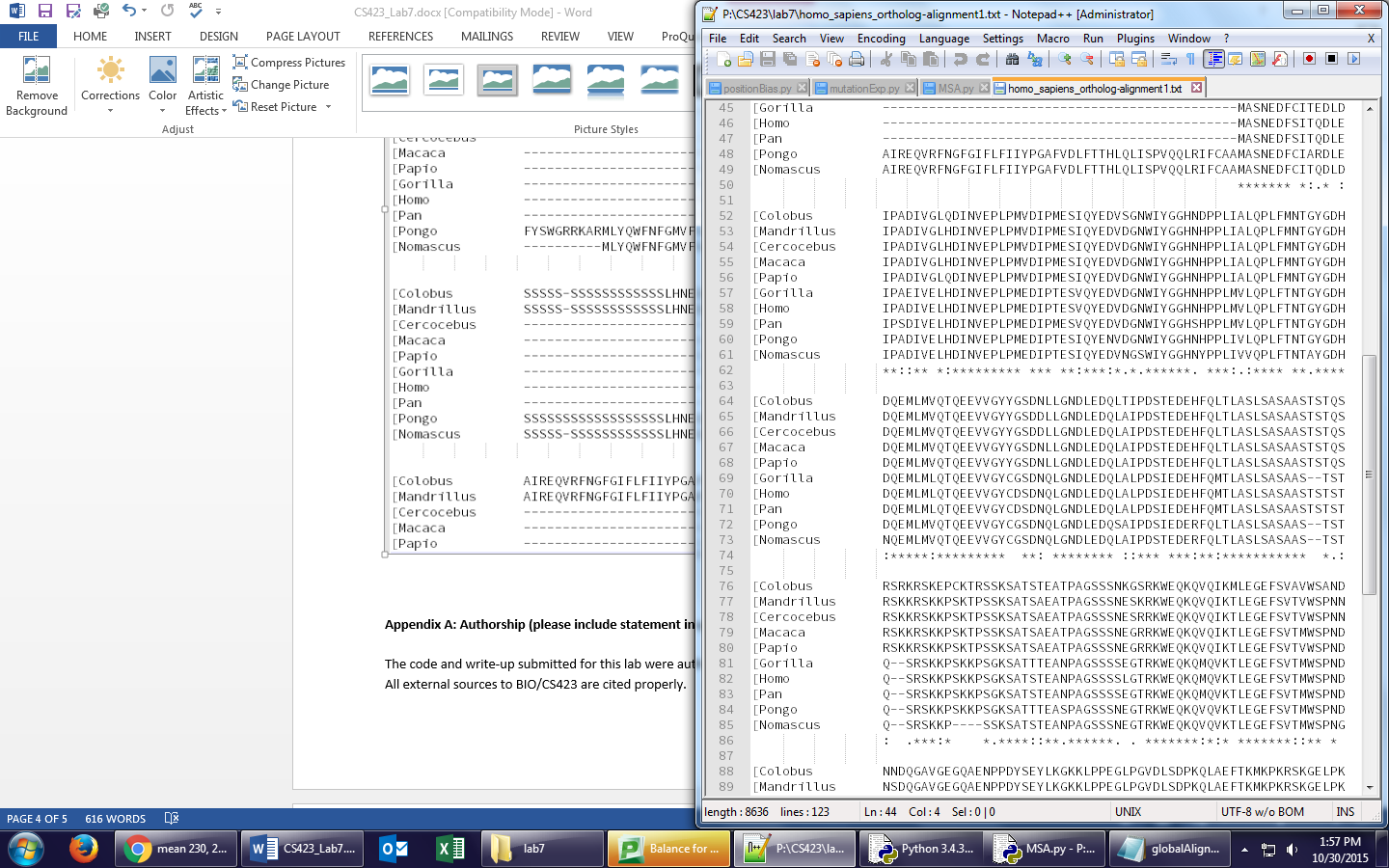
* 1. (4 pts) One criticism of phylogenetic analysis of HIV is that the virus can mutate within an individual, producing many distinct viruses with distinct gene sequences, which might obscure the analysis. How might researchers try to compensate for this criticism?

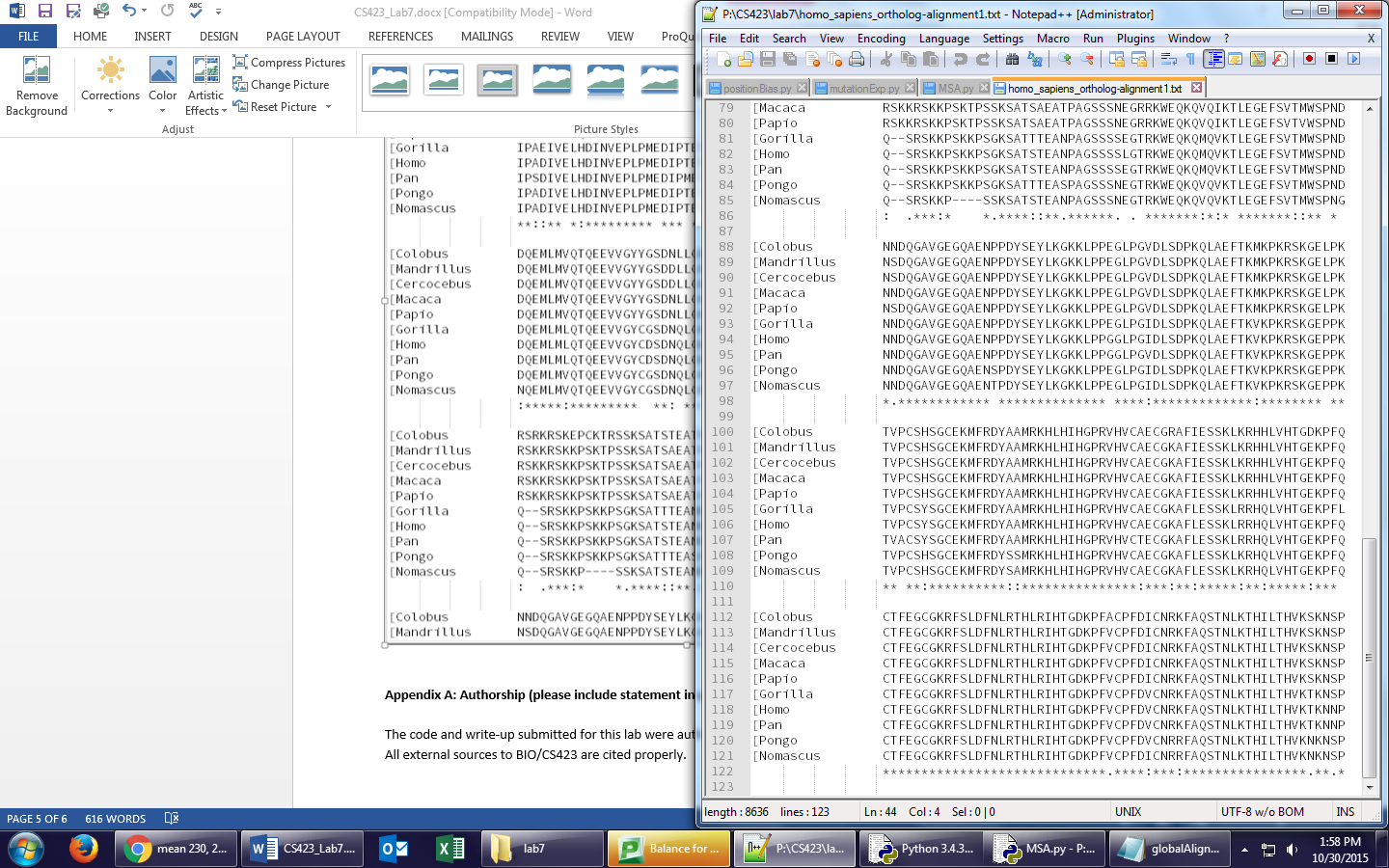
**Depending on how distinct the viruses are, researches could try to compensate for this criticism by taking a global alignment of the gene sequences to make an “average” to use for phylogenetic analysis. This “average sequence” of the virus could be used for the doctor. They could also repeat phylogenetic analysis over a period of months to see if the results still hold true. In addition it might be possible to obtain the concentration of the distinct gene sequences using quantitative PCR and base phylogenetic analysis on the gene sequence that has the highest concentration in the individual.**

**Appendix 1: MSA**

**Human Ortholog MSA**







**Appendix A: Authorship (please include statement in your write-up)**

The code and write-up submitted for this lab were authored by the named person(s) on this lab report. All external sources to BIO/CS423 are cited properly.

**Appendix B: Code (23 points, based on correctness and style):**

Copy and paste the code (MSA.py) you wrote for lab 7 here (use Courier 8pt- font). Also, upload the code files to Moodle as part of you zip folder.

#!/usr/bin/python

################################################

# Lab7 CS423

# Shaun Stice, Caleb Piekstra

# Fall 2015

################################################

import random, math

####################################################################

# Create a 2D table with the given number of rows and columns

# and fills all entries with value given as a parameter

# (function completed for you)

####################################################################

def createTable(numRows, numCols, value):

table = []

row = 0

# create 2D table initialized with value

while (row < numRows):

table.append([])

col = 0

while (col < numCols):

table[row].append(value)

col = col + 1

row = row + 1

return table

######################################################################

# Determines the score of the optimal global alignment of two sequences

# Taken from lab 5

# Just calculates the score (not the alignment; do not need to

# print out tables; just returns the optimal score)

######################################################################

def globalAlignmentScore(s1, s2):

# add a blank string as padding

s1 = ' ' + s1

s2 = ' ' + s2

# Scoring system

MATCH = 5

MISMATCH = -4

GAP = -6

# set table size

NUM\_ROWS = len(s2)

NUM\_COLS = len(s1)

# Create table and fill it with zeros

c = createTable(NUM\_ROWS, NUM\_COLS, 0)

# Create table for getting back the optimal alignment, fill table with "F"

# Suggest you use "D", "L", and "T" for diagonal, left, and top

d = createTable(NUM\_ROWS, NUM\_COLS, "F")

# complete this part of the function

# implement dynamic programming algorithm for global alignment here

# fill in entries in cost table and direction table

for i in range(1, NUM\_ROWS):

c[i][0] = GAP\*i

for j in range(1, NUM\_COLS):

c[0][j] = GAP\*j

for i in range(1, NUM\_ROWS):

for j in range(1, NUM\_COLS):

if s1[j] == s2[i]:

m = MATCH

else:

m = MISMATCH

left = c[i][j-1] + GAP

top = c[i-1][j] + GAP

diag = c[i-1][j-1] + m

c[i][j] = max(left,top,diag)

if c[i][j] == diag:

d[i][j] = "D"

elif c[i][j] == top:

d[i][j] = "T"

else:

d[i][j] = "L"

# return optimal score (lower right-hand cell in table]

return c[NUM\_ROWS-1][NUM\_COLS-1]

#################################################################

# Generate a random DNA sequence with given length and nucleotide

# probabilities

# complete this function

# A previous lab exercise may be helpful

#################################################################

def generateRandomSequence(length, A\_prob, C\_prob, G\_prob, T\_prob):

seq = ""

for i in range(0, length):

ran = random.random()

if ran < A\_prob:

seq += "A"

elif ran < A\_prob + T\_prob:

seq += "T"

elif ran < A\_prob + T\_prob + C\_prob:

seq += "C"

else:

seq += "G"

return seq

##################################################################

# Genenerate a random DNA sequence comparable in nucleotide

# distribution as the given input sequence

# complete this function

##################################################################

def generateComparableRandomSequence(s):

# calculate nucleotide frequencies and call generateRandomSequence function

totalLen = len(s)

sList = list(s)

percentA = sList.count('A')/totalLen

percentC = sList.count('C')/totalLen

percentT = sList.count('T')/totalLen

percentG = sList.count('G')/totalLen

# to create the random DNA sequence similar in nucleotide composition and same length

return generateRandomSequence(totalLen, percentA, percentC, percentT, percentG)

####################################################################

# For the two input sequences s1 and s2, calculate the distance score D where

# D is definted as 100.0\*(-ln(S\_norm)).

#

# S\_norm = (S\_global - S\_rand) / (S\_iden - S\_rand)

#

# S\_global is the optimal global alignment score between s1 and s2.

#

# S\_iden is the average of the global alignment scores of s1 aligned

# with s1 and s2 aligned with s2.

#

# S\_rand is the average of 1000 global alignment scores between

# sequences similar in composition as s1 and s2.

# complete this function

###################################################################

def calculateDistanceScore(s1, s2):

sGlobal = globalAlignmentScore(s1, s2)

sIden = (globalAlignmentScore(s1, s1) + globalAlignmentScore(s2, s2))/2.0

sRandSum = 0

for \_ in range(1000):

s1Rand = generateComparableRandomSequence(s1)

s2Rand = generateComparableRandomSequence(s2)

sRandSum += globalAlignmentScore(s1Rand, s2Rand)

sRand = sRandSum/1000

sNorm = (sGlobal - sRand) / (sIden - sRand)

D = 100.0 \* (-(math.log(sNorm)))

return D

########################################################

### End of functions ###################################

########################################################

# DNA sequences

seq1 = "CGATAGTGCTATATCTAGCGCCGTCTAGATGCATTATACGATATCG"

seq2 = "AACGACATGGCTCGTGCTATTACGCGCGAATATCC"

seq3 = "ATAGTGCTATACTCGTGCTATTCTAGATGCCGCGATATAT"

seq4 = "GGATAGGCTATATCTAGCGCGTCTAGATGCATTTACGATATC"

seq5 = "TACGACATGCGCTCGTGCATATTAGCGCGCGATATATCG"

# calculate global alignment scores for all ten pairs

print("Global alignment scores")

print("seq1 and seq2: " + str(globalAlignmentScore(seq1, seq2)))

print("seq1 and seq3: " + str(globalAlignmentScore(seq1, seq3)))

print("seq1 and seq4: " + str(globalAlignmentScore(seq1, seq4)))

print("seq1 and seq5: " + str(globalAlignmentScore(seq1, seq5)))

print("seq2 and seq3: " + str(globalAlignmentScore(seq2, seq3)))

print("seq2 and seq4: " + str(globalAlignmentScore(seq2, seq4)))

print("seq2 and seq5: " + str(globalAlignmentScore(seq2, seq5)))

print("seq3 and seq4: " + str(globalAlignmentScore(seq3, seq4)))

print("seq3 and seq5: " + str(globalAlignmentScore(seq3, seq5)))

print("seq4 and seq5: " + str(globalAlignmentScore(seq4, seq5)))

# calculate distance score for all ten pairs

print("")

print("Distance scores")

print("seq1 and seq2: " + str(calculateDistanceScore(seq1, seq2)))

print("seq1 and seq3: " + str(calculateDistanceScore(seq1, seq3)))

print("seq1 and seq4: " + str(calculateDistanceScore(seq1, seq4)))

print("seq1 and seq5: " + str(calculateDistanceScore(seq1, seq5)))

print("seq2 and seq3: " + str(calculateDistanceScore(seq2, seq3)))

print("seq2 and seq4: " + str(calculateDistanceScore(seq2, seq4)))

print("seq2 and seq5: " + str(calculateDistanceScore(seq2, seq5)))

print("seq3 and seq4: " + str(calculateDistanceScore(seq3, seq4)))

print("seq3 and seq5: " + str(calculateDistanceScore(seq3, seq5)))

print("seq4 and seq5: " + str(calculateDistanceScore(seq4, seq5)))