Assignment 5

Due Wednesday, September 30 before midnight (California time)

We are going to write a UPGMA algorithm in Python and analyze some human data.

On Blackboard, you can find a file named upgmaData.py, which contains:

1. *humansList* (and *testList*) are species lists. They are in the form of a tuple.
2. *humansDistances* (and *testDistances*) are dictionaries specifying pairwise distances between species. They are in the form where the key is a tuple which is a pair of species name, e.g., ("species A","species B") and the value is the pairwise distance. The following are the details for the human data:

Modern human data: distances based on mitochondrial sequence

San	San individual from southern Africa
Yoruba	Yoruba individual from western Africa
Finnish	Finnish individual from northern Europe
Kikuyu	Kikuyu individual from eastern Africa
Papuan	Papuan individual from New Guinea
Han	Han individual from China

Note: use "*from upgmaData import **" on the first line of your code to be able to use the data.

Write the following Python *class* for a tree node (as taught in Jinsen's discussion):

1. *TreeNode* [3pt]
   Has four properties including *key*, *distance*, *left* and *right*, where *key* is for storing species name, *distance* is for branch length, *left* is for the left child and *right* is for the right child. The class has a function *print* to print the tree structure in the form of (*key*, *distance*, *left*, *right*). For example, ('W_X',0.5,('W',0,(),()),('X',0,(),())) is for the tree W_X with left child ('W',0,(),()) and right child ('X',0,(),()); the branch length to its child is 0.5.

Write the following functions:

2. *findClosestPair(Distances)* [3pt]
   Inputs *Distances* dictionary and returns the key for the pair of nodes that are closest. For example, if the input is *testDistances*, the function returns *('W', 'X')*.

3. *updateDist(Distances, newNode)* [3pt]
   Inputs *Distances* dictionary and a new tree node. Updates the *Distances* dictionary by adding the distances between *newNode* and all the other nodes. Does not calculate the distance between

newNode and its children in the *Distances*. This function returns the updated *Distances* dictionary.

4. *upgma(Distances)* [4pt]
   This function runs the UPGMA algorithm described in lecture. First, it initializes a list *TreeNodesList* with multiple new *TreeNode* referring to *humansList* (the list *TreeNodesList* should contain six *TreeNodes*). Then, it repeats the following steps until there is only one tree node left in the list *TreeNodesList*, at which point this tree is returned.
   a. Find the key for the closest pair of nodes in *Distances* (use *findClosestPair*)
   b. Create a new *TreeNode* and assign the pair of nodes stored in *TreeNodesList* as its children. Calculate the distance between the new *TreeNode* to its children and store the value in the new *TreeNode*.
   c. Update the *TreeNodesList* by removing the pair of nodes referring to a.
   d. Update the *Distances* dictionary (use *updateDist*)
   e. Add the new *TreeNode* into *TreeNodesList*.

Use aforementioned functions to analyze human data:

5. Use *upgma* to generate one tree using *humansList* and *humansDistances*. Turn in your code, the output the code for humans, and explain what you find from the results [3pt].