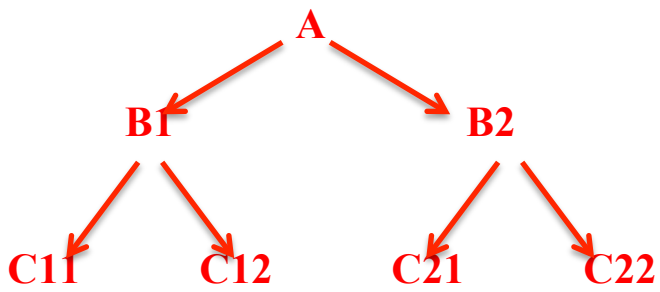
Assignment 6 (lectures 11, 12,13), Due date: Monday, October 23rd at 11:59pm

1. MRCA means "Most Recent Common Ancestor" and provides important information about the time to species splitting. In the example we worked through in class, the MRCA of *Pan troglodytes* and *Tarsius tarsier* is Haplorrhini.
We will use an abstracted and simplified tree (shown below) and the corresponding dictionary describing its relationships. The tree given below can be summarized in Newick format, in case you are more familiar with that, with the following:
((((C11,C12),B1),((C21,C22),B2)),A)



tax_dict={**"C11"**:**"B1"**,**"C12"**:**"B1"**,**"C21"**:**"B2"**,**"C22"**:**"B2"**,**"B1"**:**"A"**,**"B2"**:**"A"**}

Write a program that includes a **recursion** function that takes two arguments, the dictionary describing child -> parent relationships given above and a list of taxa, and returns the MRCA of the inputted taxa.

Of course, there is more than one way of solving this problem (so feel free to disregard the following suggestions if you have thought up a better way to solve it) but here are a few hints for the way that I solved it. For my solution, I adopted the following strategy of breaking down the problem into multiple functions:

- One function that processes the list of taxon that you want to compare so that you are comparing two elements of the list at a time (I found the .pop method useful)
- A second function that finds the common ancestor of these two elements produced by the first function
- A third function (in my solution this was the one that used recursion) to find all the ancestors of each of the two taxons. In my solution, this was the same as the code that we went through during lecture.
- The above processes need to continue for as long as you have elements in your input list.

You will need to write one program that **should solve for the most general case (in case your taxon had more members or more internal nodes than currently listed). For instance, if I swapped your library that detailed the relationship pictured above for another library that added a row of "d" elements (d111, d112, d121, d122, d211, d212, d221, d222), your code should still work.**

Your code will allow you to call the function with 3 or greater elements and *your code will work with elements from different rows (so you could call it with C11, C12 and B2 and expect it to give you A as the common ancestor).*

2. Revisit your answer to assignment 4. Re-write it with **a list comprehension to chop up the dna sequence into codons.** Depending on your particular solution for assignment 4, you may require two list comprehensions. Other than that, your code should be the same as in assignment 4 (there is no need to do anything fancier – just the list comprehension substitution – unless you happen to want to make your code from assignment 4 more efficient in other ways).