Work with your neighbor. (This will be graded for participation only.)

1. Below is a recursive function `zip(a,b)` where `a` and `b` are lists of any type. The function `zip(a,b)` returns a list of 2-tuples where the first tuple is `(a[0],b[0]))`, the second is `(a[1],b[1])`, etc. Zipping stops when the shorter list runs out. For example, if `len(a)` is `3` and `len(b)` is `2`, then `len(zip(a,b))` is `2`.

```
def zip(a,b):
    if a == [] or b == []:
        return []
    else:
        return list((a[0], b[0])) + zip(a[1:], b[1:])
```

Now let's rewrite `zip` using a helper function. We will call `zip(a,b)` as before, but the new version will call a helper function `zip_helper(a,b,result)` that takes an additional list argument called `result`.

Before each recursive call, `zip_helper(a,b,result)` will modify `result` by concatenating the new tuple of `(a[0], b[0])` to `result` and then pass the modified `result` list to the recursive call.

```
def zip(a,b):
    # call the helper function with an empty list
    return zip_helper(a,b,[])

def zip_helper(a,b,result):
```

2. **`    # your code goes here`**

ANS:

```
def zip(a,b):
    if a == [] or b == []:
        return []
    else:
        return list((a[0], b[0])) + zip(a[1:], b[1:])
```

3. Tree terminology:

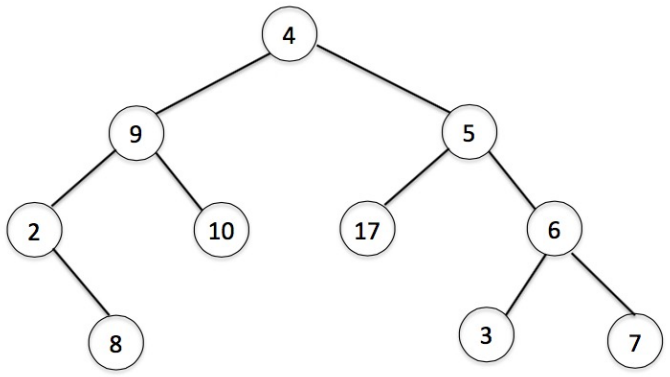
sibling – a set of nodes that all share the same parent

degree – the number of children a node has

edge/path – the line that connects a parent to its child

level – the number of "edges" from the root to a node

height – the maximum of the levels of the nodes in the tree (or, the length of the longest path in the tree)



ANS:

What are the leaves of this tree? 8, 10, 17, 8, 3, 7

Which nodes have multiple children? 4, 9, 5, 6

Which nodes have multiple parents? No node has multiple children

What is the root of this tree? 4

Which nodes are siblings of 10? 2

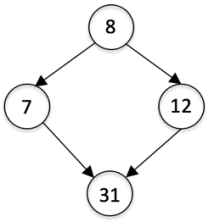
What is the level of 6? 2

What is the height of this tree? 3

4. Draw the tree that represents the main folders (directories) and files on your computer.

   ANS: various answers

5. Is the following a tree? Discuss with your neighbors. Explain why or why not.

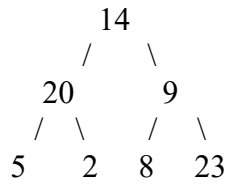


ANS:

This is not a tree. A node in a tree cannot have more than one parent. The node with value 31 has two parents.

6. Draw a binary tree with 7 nodes. Make it as wide as you can, so that the tree is as short as possible.

ANS:



How many nodes are at each level?

ANS:

Level 0: 1

Level 1: 2

Level 2: 4

How many nodes would you need to add in order to fill up another level?

ANS:

8

(**Note:** these are powers of 2.)

7. Using the same 7 node values as the tree above, draw another tree but make it has tall as possible.

ANS: (Given the tree above in 4. The values in your nodes will be different.)