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

1. Here is the example use of filter that filters out the words that start with a vowel:

```
list(filter(lambda w: w[0] in "aeiou", alist)
```

   For comparison, write this using a list comprehension below:

2. Iterators) Below is the example user-defined iterator `Reverse` discussed in lecture:

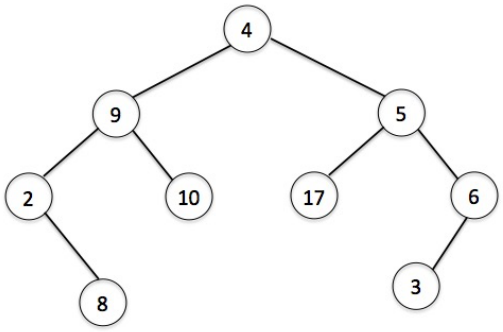
```
class Reverse:
    def __init__(self, data):
        self._data = data
        # start the index at the end of the list
        # next() will decrement it before use
        self._index = len(data)

    def __iter__(self):
        return self

    def __next__(self):
        if self._index == 0:
            raise StopIteration
        self._index = self._index - 1
        return self._data[self._index]
```

   Using this as a template, write an iterator called `EveryOther` that produces every other element of a list. The first element produced is the element at index 0, the second element produced is the element at index 2, and so on.

3. (Generators) Now write a generator called `every_other()` that is a generator version of `EveryOther`.

Final Exam Review

4. (Trees) Use the tree below to answer the following questions:



a) What is the height of the tree?

b) Write the inorder traversal:

c) Write the postorder traversal:

5. **(ADT)** For this problem, you are to implement an abstract data type named `CharStack` that represents a stack of characters (strings of length 1).

   Implement the following **methods** for the `CharStack` class:
   - `__init__`: initializes an empty stack
   - `push`: puts its argument, a character, on the top of the stack; returns `None`
   - `pop`: removes the top character from the stack and returns it; returns `None` if the stack is empty
   - `swap`: swaps the top two characters of the stack; always returns `None`; has no effect if the stack has less than two elements
   - `is_empty`: returns `True` if the stack is empty and `False` otherwise

Restriction: The `CharStack` class has only one **attribute** which is of type string.

Here is an example of usage:

```
>>> cs = CharStack()
>>> cs.push('p')
>>> cs.push('a')
>>> cs.push('n')
>>> cs.push('s')
>>> print(cs)
snap
>>> cs.swap()
>>> cs.pop()
'n'
>>> print(cs)
sap
>>> cs.is_empty()
False
>>>
```