6.101 Recitation 18: Week 10 Autocomplete Wrap-up **4/22/24**

This sheet is yours to keep!

Question 1: Review the code below, which creates a small PrefixTree `t`. Draw a diagram to represent the structure of `t`.

```
t = PrefixTree()
t['bar'] = 1
t['bark'] = 2
t['bat'] = 3
t['cart'] = 4
t['cats'] = 5
t['at'] = 6
```

Question 2: The implementation of __getitem__ and __setitem__ below is correct but has many similarities. Refactor these functions to decrease repetition while maintaining correctness. For an added bonus, try writing the code iteratively to increase efficiency.

```
class PrefixTree:
    def __init__(self):
        self.value = None
        self.children = {}

    def __setitem__(self, key, value):
        if not isinstance(key, str):
            raise TypeError
        elif not key:
            self.value = value
        else:
            if key[0] not in self.children:
                self.children[key[0]] = PrefixTree()
            self.children[key[0]][key[1:]] = value

    def __getitem__(self, key):
        if not isinstance(key, str):
            raise TypeError
        elif not key:
            if self.value is None:
                raise KeyError
            return self.value
        elif key[0] not in self.children:
            raise KeyError
        else:
            return self.children[key[0]][key[1:]]
```

Question 4: Look at the correct implementation of autocomplete provided below. How could we refactor this code to increase efficiency?

```
def autocomplete(ptree, prefix, max_count=None):
    if not isinstance(prefix, str):
        raise TypeError

    all_words = [i for i in ptree if i[0].startswith(prefix)]

    if max_count is None:
        max_count = len(all_words)

    out_words = []
    for _ in range(max_count):
        best = (None, float("-inf"))
        for i in all_words:
            if i[1] > best[1] and i not in out_words:
                best = i
        if best != (None, float("-inf")):
            out_words.append(best)

    return [i[0] for i in out_words]
```

R18 Participation Credit **Kerberos : _____@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 3: For each buggy implementation of the iter function below:
- What is going wrong?
- What changes do we need to make to fix the code, while keeping the same structure?

```
class PrefixTree:
    # other code …


    def __iter__(self):  # version A
        def helper(self, prefix):
            if self.value is not None:
                yield (prefix, self.value)
            for letter, child in self.children.items():
                helper(child, prefix + letter)
        self.helper(self, '')




    def __iter__(self):  # version B
        for letter, subtree in self.children.items():
            if subtree.value:
                yield (letter, self.value)


            yield from [(word+letter, val) for word, val in subtree]
```