INSTRUCTIONS

- You have 5 minutes to complete this quiz.
- The exam is closed book, closed notes, closed computer, closed calculator.
- Mark your answers on the exam itself. We will not grade answers written on scratch paper.
- For multiple choice questions, fill in each option or choice completely.
  - □ means mark all options that apply
  - ○ means mark a single choice

<table>
<thead>
<tr>
<th>Last name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>First name</td>
<td></td>
</tr>
<tr>
<td>Student ID number</td>
<td></td>
</tr>
<tr>
<td>CalCentral email (@berkeley.edu)</td>
<td></td>
</tr>
<tr>
<td>Discussion Section</td>
<td>__ __ __</td>
</tr>
</tbody>
</table>

All the work on this exam is my own. (please sign)

0. Your thoughts? If Cal’s mascot was a turkey, what would be its name?
1. **Tree Time**

Fill in the square to the left of each line if *removing the line* would help pass the doctests. In the space to the right, briefly describe why each line should be removed. *Remove as many lines as possible.*

*Recall*: The special method `__iter__` is called by the built-in `iter()` and should return an iterator. `IterableTree.__iter__` is a generator that yields the root value of the tree and then each value in its branches.

```python
class Tree:
    def __init__(self, label, branches=()):
        self.label = label
        self.branches = list(branches)

    def is_leaf(self):
        return not self.branches

class IterableTree:
    class __init__(self, label, branches=()):
        Tree.__init__(label, branches)

    def __iter__(self):
        """Yield the entries of this tree."

        >>> T = IterableTree
        >>> t = T('A', [T(2, [T('C'), T(4)]), ...
                   T('E', [T(6)])])
        >>> list(t)
        ['A', 2, 'C', 4, 'E', 6]
        """

        yield self.label

        yield label

        for branch in self.branches:
            branch = iter(branch)

            for label in branch:
                for label in branch():
                    yield self.label

        yield label

        yield self.label
```

- [ ] class Tree:
- [ ] def __init__(self, label, branches=()):
- [ ] self.label = label
- [ ] self.branches = list(branches)
- [ ] def is_leaf(self):
- [ ] return not self.branches
- [ ] class IterableTree:
- [ ] class __init__(self, label, branches=()):
- [ ] Tree.__init__(label, branches)
- [ ] def __iter__(self):
- [ ] """Yield the entries of this tree."
- [ ] >>> T = IterableTree
- [ ] >>> t = T('A', [T(2, [T('C'), T(4)]), ...
- [ ] >>> T('E', [T(6)])])
- [ ] >>> list(t)
- [ ] ['A', 2, 'C', 4, 'E', 6]
- [ ] """
- [ ] yield self.label
- [ ] yield label
- [ ] for branch in self.branches:
- [ ] branch = iter(branch)
- [ ] for label in branch:
- [ ] for label in branch():
- [ ] yield self.label
- [ ] yield label
- [ ] yield self.label
- [ ] yield label