INSTRUCTIONS

- You have 10 minutes to complete this quiz.
- The exam is closed book, closed notes, closed computer, closed calculator.
- The final score for this quiz will be assigned based on effort rather than correctness.
- Mark your answers on the exam itself. We will not grade answers written on scratch paper.
- For multiple choice questions,
  - □ 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>Teaching Assistant</td>
<td>Alex Stennet</td>
</tr>
<tr>
<td></td>
<td>○ Angela Kwon</td>
</tr>
<tr>
<td></td>
<td>○ Ashley Chien</td>
</tr>
<tr>
<td></td>
<td>○ Joyce Luong</td>
</tr>
<tr>
<td></td>
<td>○ Karthik Bharathala</td>
</tr>
<tr>
<td></td>
<td>○ Kavi Gupta</td>
</tr>
<tr>
<td></td>
<td>○ Kelly Chen</td>
</tr>
<tr>
<td></td>
<td>○ Michael Gibbes</td>
</tr>
<tr>
<td></td>
<td>○ Michelle Hwang</td>
</tr>
<tr>
<td></td>
<td>○ Mitas Ray</td>
</tr>
<tr>
<td></td>
<td>○ Rocky Duan</td>
</tr>
<tr>
<td></td>
<td>○ Samantha Wong</td>
</tr>
</tbody>
</table>

Name of the person to your left

Name of the person to your right

All the work on this exam is my own. (please sign)
1. (5 points)  Trieing to Get All the Points

A trie is a type of tree where the values of each node are letters representing part of a larger word. A valid word is a string containing the letters along any path from root to leaf. For simplicity, assume that our trie is represented with the tree abstract data type and where the value of each node contains just a single letter.

```
>>> greetings = tree('h', [tree('i'),
...     tree('e', [tree('l', [tree('l', [tree('o')])]),
...     tree('y')])])
```

```
>>> print_tree(greetings)
   h
     i
     e
     l
       l
       o
     y
```

Recall: The tree abstract data type is defined with the following constructors and selectors.

```python
def tree(root, branches=[]):
    return [root] + list(branches)

def root(tree):
    return tree[0]

def branches(tree):
    return tree[1:]

def is_leaf(tree):
    return not branches(tree)
```

(a) (5 pt) Define a function, collect_words, that takes in a trie t and returns a Python list with all the words contained in the trie.

```python
def collect_words(t):
    """Return a list of all the words contained in the tree where the value of each node in the tree is an individual letter. Words terminate at the leaf of a tree."

    >>> collect_words(greetings)
    ['hi', 'hello', 'hey']
    ""

    if _________________________________________________________:
        _________________________________________________________
        _________________________________________________________
        _________________________________________________________
        _________________________________________________________

    words = __________________________________________________________
    __________________________________________________________
    __________________________________________________________
    __________________________________________________________
    __________________________________________________________

    words += _________________________________________________________
    __________________________________________________________
    __________________________________________________________
    __________________________________________________________
    __________________________________________________________

    return words
```
>>> greetings = tree('h', [tree('i'),
    tree('e', [tree('l', [tree('l', [tree('o')])])]),
    tree('y')])

>>> print_tree(greetings)
  h
    i
e
    l
    o
    y

(b) (0 pt) Extra Practice
Define a function, has_path, that takes in a trie t and a string word. It returns True if there is a path that starts from the root where the letters along the path spell out the word, and False otherwise.

```python
def has_word(t, word):
    """Return whether there is a path spelling out word in the trie t."
    >>> has_word(greetings, 'h')
    True
    >>> has_word(greetings, 'i')
    False
    >>> has_word(greetings, 'hi')
    True
    >>> has_word(greetings, 'hello')
    True
    >>> has_word(greetings, 'hey')
    True
    >>> has_word(greetings, 'bye')
    False
    """
    if ________________________________:
        return ____________________________
    elif ________________________________:
        return ____________________________
    ________________________________
    ________________________________
    ________________________________
    return ____________________________