CS 61A Fall 2017

Structure and Interpretation of Computer Programs

Quiz 4 Solutions

INSTRUCTIONS

- You have 10 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**

Last name	
First name	
Student ID number	
CalCentral email (`_@berkeley.edu`)	
Discussion Section	___ ___ ___
All the work on this exam is my own. **(please sign)**	

0. **Your thoughts?** How's life?

1. **Trie Recursion**

   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.

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

```
def tree(label, branches=[]):
    """Construct a tree with the given label value and a list of branches."""

def label(tree):
    """Return the label value of a tree."""

def branches(tree):
    """Return the list of branches of the given tree."""

def is_tree(tree):
    """Returns True if the given tree is a tree, and False otherwise."""

def is_leaf(tree):
    """Returns True if the given tree's list of branches is empty, and False otherwise."""
```

   Implement `collect_words`, which takes in a trie `t` and returns a Python list containing all the words contained in the trie.

```
>>> 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
```

```
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 is_leaf(t):

        return [label(t)]

    words = []

    for branch in branches(t):

        words += [label(t) + word for word in collect_words(branch)]

    return words
```