CS 61A Structure and Interpretation of Computer Programs
Summer 2017

Quiz 10 Solutions

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**

Last name	
First name	
Student ID number	
CalCentral email (`_@berkeley.edu`)	
Teaching Assistant	◯ Alex Stennet ◯ Kelly Chen ◯ Angela Kwon ◯ Michael Gibbes ◯ Ashley Chien ◯ Michelle Hwang ◯ Joyce Luong ◯ Mitas Ray ◯ Karthik Bharathala ◯ Rocky Duan ◯ Kavi Gupta ◯ Samantha Wong
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) Run, Forrest, Run!

(a) (2 pt) For each of the expressions below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. If an error occurs, write "Error", but include all output displayed before the error. If a function value is displayed, write "Function".

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

    def __repr__(self):
        if self.branches:
            branches_str = ', ' + repr(self.branches)
        else:
            branches_str = ''
        return 'Tree(' + repr(self.root) + branches_str + ')'

    def is_leaf(self):
        return not self.branches

forrest = Tree(1)
gump = Tree(1, [forrest, forrest])
forrest.root = 2
forrest = Tree(forrest)
```

```
>>> run = Tree(forrest, gump.branches)
>>> run
Tree(Tree(Tree(2)), [Tree(2), Tree(2)])
>>> forrest.root = 1
>>> run
Tree(Tree(1), [Tree(2), Tree(2)])
```

(b) (3 pt) Implement `all_paths` which takes in a `Tree` and returns a Python list containing all the paths (represented as linked lists) from the root to the leaves. The `Tree` class definition is provided above.

```
class Link:
    def __init__(self, first, rest=empty):
        self.first = first
        self.rest = rest

def all_paths(t):

    if t.is_leaf():

        return [Link(t.root)]

    result = []

    for b in t.branches:

        result += [Link(t.root, path) for path in all_paths(b)]

    return result
```