CS 61A Structure and Interpretation of Computer Programs

Summer 2013 Midterm 2

INSTRUCTIONS

- You have 2 hours to complete the exam.
- The exam is closed book, closed notes, closed computer, closed calculator, except one hand-written 8.5" × 11" crib sheet of your own creation and the official 61A midterm 1 study guide attached to the back of this exam.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation.

Last name	
First name	
SID	
Login	
TA & section time	
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. (9 points) Do you feel lucky, punk?

For each of the following expressions, write the value to which it evalutes in the interactive Python interpreter. The first two rows have been provided as examples.

- In the **Evaluates to** column, write the value to which the expression evaluates. If it evaluates to a function value, write FUNCTION. If evaluation causes an error, write ERROR.

Assume that you have started Python 3 and executed the following statements:

```
lucky = [3, 4, 5, 6, 7]
beginning = [lucky, [3, 4, 5, 6, 7]]
punk = beginning[1]
```

Expression	Evaluates to
`3 + 3`	6
`1 / 0`	ERROR
`lucky[1]`	
`beginning[1] == lucky`	
`beginning[1] is lucky`	
`punk[2:]`	
`punk == lucky`	

Now, assume the following line is executed *in the same interpreter session*:

```
beginning[0].append(10)
```

Expression	Evaluates to
`lucky`	
`beginning[0]`	
`beginning[1]`	
`punk == lucky`	

2. (6 points) Not with a fizzle, but with a bang

(a) (2 pt) Consider the following function definition.

```
def fizzle(n):
    if n <= 0:
        return n
    elif n % 23 == 0:
        return n
    return fizzle(n - 1)
```

Circle the correct order of growth for a call to `fizzle(n)`:

$\Theta(1)$ $\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n^2)$ $\Theta(n^3)$ $\Theta(2^n)$

(b) (2 pt) Now consider the following function definitions.

```
def boom(n):
    if n == 0:
        return "BOOM!"
    return boom(n - 1)

def explode(n):
    if n == 0:
        return boom(n)
    i = 0
    while i < n:
        boom(n)
        i += 1
    return boom(n)
```

Circle the correct order of growth for a call to `explode(n)`:

$\Theta(1)$ $\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n^2)$ $\Theta(n^3)$ $\Theta(2^n)$

(c) (2 pt) Now consider the following function definition.

```
def dreams(n):
    if n <= 0:
        return n
    if n > 0:
        return n + dreams(n // 2)
```

Circle the correct order of growth for a call to `dreams(n)`:

$\Theta(1)$ $\Theta(\log n)$ $\Theta(\sqrt{n})$ $\Theta(n)$ $\Theta(n^2)$ $\Theta(n^3)$ $\Theta(2^n)$

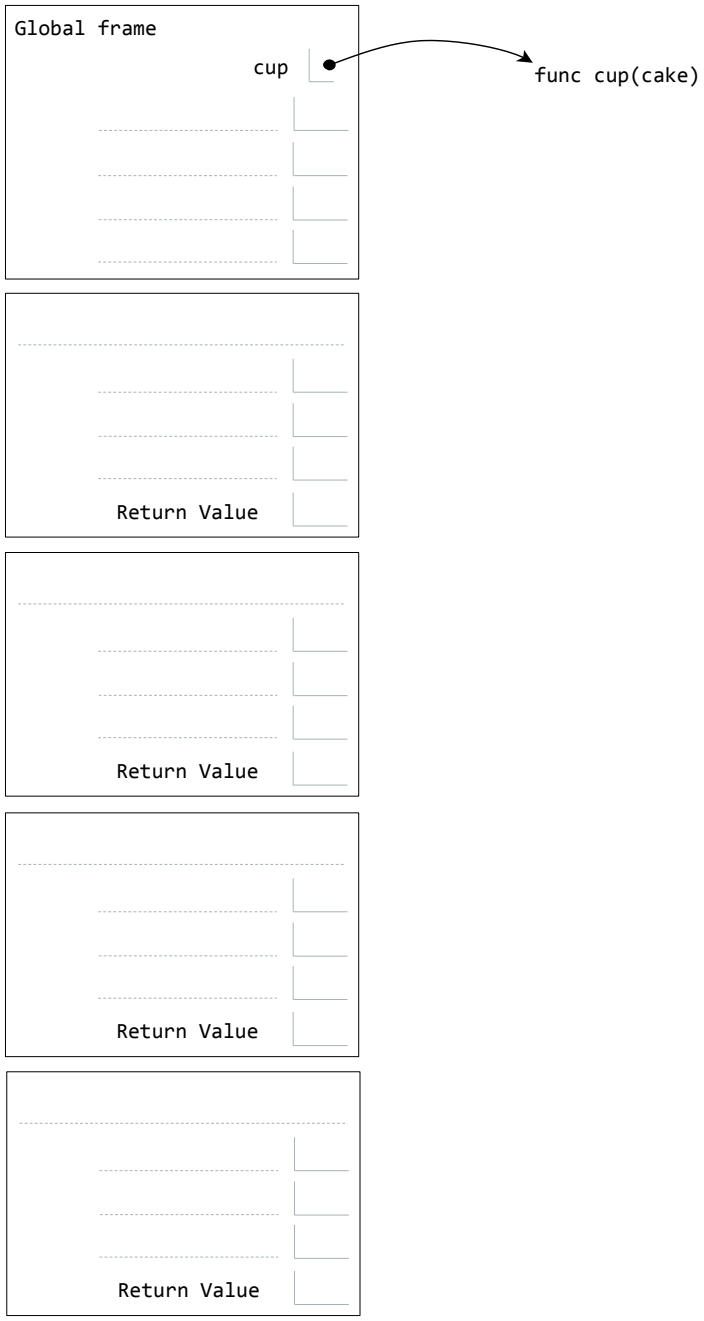
3. (12 points) Cupcakes cupcakes cupcakes cupcakes cupcakes

(a) (6 pt) Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You need only show the final state of each frame. *You may not need to use all of the spaces or frames.*

A complete answer will:

- Add all missing names, labels, and parent annotations to all local frames.
- Add all missing values created during execution.
- Show the return value for each local frame.

```
def cup(cake):
    if len(cake) != 1:
        frosting = cake.pop(0)
        cup(cake)
        cake.append(frosting)
frosting = [1, 2, 3]
cup(frosting)
```



(b) **(6 pt)** Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You need only show the final state of each frame. *You may not need to use all of the spaces or frames.*

A complete answer will:

- Add all missing names, labels, and parent annotations to all local frames.
- Add all missing values created during execution.
- Show the return value for each local frame.

```
n = 3

def follow():
    sunshine = 10
    def x(n):
        nonlocal sunshine
        def y(fn):
            nonlocal sunshine
            sunshine = fn(sunshine) + n
            return sunshine
        sunshine += n
        return y
    return x(5)

rain = follow()
fall = rain(lambda x: x * n)
```

4. (4 points) Good job, Mark

Mark is tired of having to dig around BearFacts to look up student information, so he wrote a script to create a dictionary of student info. However, when he did so, he was sleep-deprived. As such, he accidentally swapped the keys and the values! Now he's stuck with a dictionary that maps any single piece of student information to the student's login:

```
>>> info['Eric Tzeng'] # info is mark's faulty dictionary
'cs61a-ta'
>>> info['eric.tzeng@berkeley.edu']
'cs61a-ta'
```

Mark is too tired to fix this, so you'll have to do it for him. We'll do this by writing a `get_info` function. It should take a student login string and a faulty info dictionary as its two arguments and return a list containing all of that student's information. The returned list can contain a student's information in any order, but it must contain *all* of the information for a student found in the faulty dictionary.

```
>>> get_info('cs61a-ta', info)
['Eric Tzeng', 'eric.tzeng@berkeley.edu']
```

You may assume that no two students have duplicate information. However, you *may not* assume that names and emails are the only pieces of information stored in the dictionary.

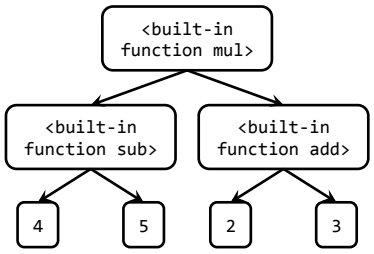
```
def get_info(login, info_dict):
    """Retrieve student information based on the student's login, using the
    provided (faulty) info_dict.

    >>> mark_dict = {'Andrew Huang': 'cs61a-tf', 'awesomeandrew@example.com': 'cs61a-tf',
    ...              'Robert Huang': 'cs61a-td', 'coolrob42@example.com': 'cs61a-td'}
    >>> get_info('cs61a-td', mark_dict)
    ['Robert Huang', 'coolrob42@example.com']  # order doesn't matter
    >>> get_info('cs61a-tf', mark_dict)
    ['awesomeandrew@example.com', 'Andrew Huang']  # order doesn't matter
    """
```

5. (5 points) Express yourself

In the first week of class, we learned that expressions like `mul(sub(4, 5), add(2, 3))` can be represented as expression trees. In this problem we use our `Tree` data structure to further explore this idea.

We propose representing expressions using a binary tree such as the `Tree` class you have seen in lecture. In this binary tree representation, leaves always represent atomic values such as `2` or `True`, and non-leaf nodes always represent function calls. For example, the expression `mul(sub(4, 5), add(2, 3))` corresponds to the following binary expression tree:



Write a function `evaluate`, which takes a binary expression `Tree` and returns what that expression would evaluate to. You may assume that all operators will take exactly 2 arguments, and that nodes always have either 0 or 2 children (never 1).

The class definition for `Tree` *can be found in the midterm study guide.*

```
def evaluate(exp):
    """Evaluates exp, which is an expression tree.

    >>> from operator import add, sub, mul
    >>> exp = Tree(3)  # 3
    >>> evaluate(exp)
    3
    >>> exp = Tree(add, Tree(3), Tree(4))  # add(3, 4)
    >>> evaluate(exp)
    7
    >>> exp = Tree(mul,
    ...            Tree(add, Tree(3), Tree(5)),
    ...            Tree(sub, Tree(5), Tree(2)))  # mul(add(3, 5), sub(5, 2))
    ...
    >>> evaluate(exp)
    24
    """
```

6. (8 points) The private life of Albert Wu

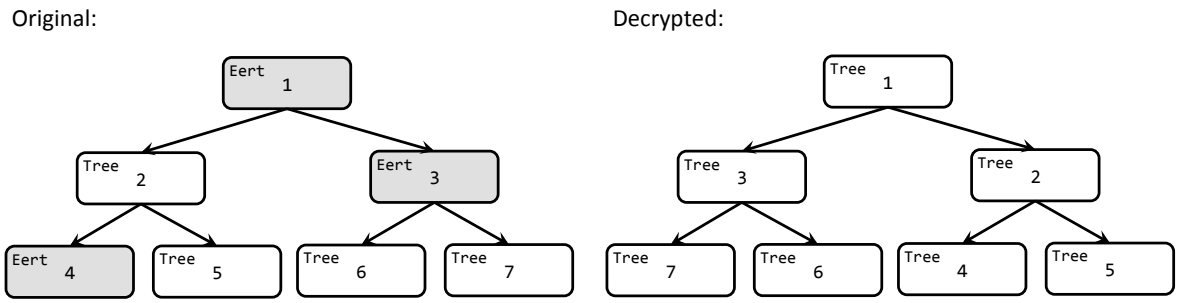
Albert keeps all of his top secret information in a binary tree. This prevents the layperson from reading his data. However, well trained computer scientists (such as you) can still access his information.

As a further layer of protection, Albert turns some of the nodes in his trees into `Eert` nodes. `Eert` nodes, which have `Tree` as their base class, are like normal `Tree` nodes, except they swap their left and right branches. (Albert settles for nothing less than the most advanced encryption techniques known to man.)

(a) (3 pt) Complete the `__init__` method for the `Eert` class **on the next page**. Make sure to use inheritance as much as possible. The `Eert` class should work as follows:

```
>>> e = Eert("61A account info",
...          Tree("Username: cs61a-te"),
...          Tree("Password: imsocool"))
>>> e.entry  # unchanged
"61A account info"
>>> e.left.entry  # swapped with right
"Password: imsocool"
>>> e.right.entry  # swapped with left
"Username: cs61a-te"
```

(b) (5 pt) Complete the definitions of the `decrypt` methods for both the `Tree` and `Eert` classes **on the next page**. When the `decrypt` method is invoked on a binary tree containing `Tree` and `Eert` nodes, it returns a copy of the binary tree, but with all `Eert` nodes replaced with `Tree` nodes. During this replacement, you should also swap the left and the right back to their proper positions! Here is a graphical representation of the process:



In terms of actual Python code, it should work as follows:

```
>>> old = Tree("League of Legends accounts",
...            Eert("Main account",  # swaps username and password positions
...                 Tree("Username: citrusdrink"),
...                 Tree("Password: imdiamond1")),
...            Tree("Smurf account",
...                 Tree("Username: yummycitrusdrink"),
...                 Eert("Password: ishouldbeapro")))  # no children, so no swapping
>>> new = old.decrypt()
>>> old.left.left.entry  # this got swapped because of the Eert
"Password: imdiamond1"
>>> new.left.left.entry  # no longer swapped
"Username: citrusdrink"
>>> old.right.right.entry
"Password: ishouldbeapro"
>>> new.right.right.entry
"Password: ishouldbeapro"
```

```
class Tree(object):

    def __init__(self, entry, left=None, right=None):
        self.entry = entry
        self.left = left
        self.right = right

    def decrypt(self):
        "*** PART B ***"
```

```
class Eert(Tree):

    def __init__(self, entry, left=None, right=None):
        "*** PART A ***"

    def decrypt(self):
        "*** PART B ***"
```

7. (6 points) kp it

Complete the definition of the `skip` procedure *in Scheme*. The `skip` procedure should take a list as its only argument and return a copy of the list with every other element removed, beginning with the first element:

```
STk> (skip '(1 2 3 4 5))
(2 4)
STk> (skip '(a (b c) d e))
((b c) e)
STk> (skip '(50))
()
```

```
(define (skip lst)
```

8. (0 points) Express yourself (v2)

Express your feelings in the space below through your choice of creative medium, such as poetry or illustration.