
CS 61A Structure and Interpretation of Computer Programs

Fall 2014

PRACTICE FINAL

INSTRUCTIONS

- You have 3 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 three official 61A midterm study guides attached to the back of this exam (not included for the practice 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	Assignment Project Exam Help
First name	
SID	https://powcoder.com
Login	Add WeChat powcoder
TA & section time	
Name of the person to your left	
Name of the person to your right	
<i>All the work on this exam is my own. (please sign)</i>	

For staff use only

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Total
/20	/20	/14	/20	/6	/80

1. (20 points) Say What?

- (a) (6 pt) For each of the following call expressions, write would be output by the interactive Python interpreter. The first two rows have been provided as examples.

In the column labeled **Interactive Output**, write all output that would be displayed during an interactive session, after entering each call expression. This output may have multiple lines. Whenever the interpreter would report an error, write ERROR. You *should* include any lines displayed before an error.

Reminder: the interactive interpreter displays the value of a successfully evaluated expression, unless it is `None`.

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

```
cookies = [ "cookies", "monster"]
monster = ["cookies", cookies]
```

```
def more(monster, cookies):
    more = monster + cookies
    if monster[0][0] == cookies[0][0]:
        monster[1] = cookies[1:]
    return more
```

Expression	Interactive Output
square(5)	25
1 / 0	ERROR
[x % 3 for x in [1, 5, 6] if x % 2 != 0]	
[cookies[1], monster[2]]	
more(cookies, monster)	
monster	
cookies[1][0]	
cookies is monster[1]	

- (b) (8 pt) (Similar to Fall 2013 Final) Each of the following expressions evaluates to a **Stream** instance. For each one, write the values of the three elements in the stream. The first value of the first stream is filled in for you.

Assume that you have started Python 3 and executed the following statements, in addition to the **Stream** class statement on your final study guide.

```
def q(t):
    def compute_rest():
        return q(t.rest.rest)
    return Stream(t.rest.first + t.first, compute_rest)

s = lambda t: Stream(t, lambda: s(t-1))
t = s(5)
u = q(t)
```

Stream	Has the first three elements
t	5, _____, _____
u	_____, _____, _____
q(u)	_____, _____, _____

- (c) (6 pt) (Similar to Fall 2013 Final) For each of the following Scheme expressions, write the Scheme value to which it evaluates. The first three rows are completed for you. If evaluation causes an error, write ERROR. If evaluation never completes, write FOREVER. **Hint:** No dot should appear in a well-formed list.

Assume that you have started the Project 4 Scheme interpreter and evaluated the following definitions.

```
(define b (lambda () (b)))
```

Expression	Evaluates to
(* 5 5)	25
'(1 2 3)	(1 2 3)
(/ 1 0)	ERROR
'((1 . (2 3 . (4))) . 5)	
(cons 1 (list 2 3 '(car (4 5))))	
((lambda () (b)))	
((lambda (x) '(x 2)) 4)	

2. (20 points) Such Enviroment Diagram

- (a) (10 pt) (From CS61A Practice Problems - <http://cs61a.org/problems>) Draw 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.

A complete answer will:

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

```
wow = 6
def much(wow):
    if much == wow:
        such = lambda wow: 5
        def wow():
            return such
        return wow
    such = lambda wow: 4
    return wow()
wow = much(much(much))(wow)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

(b) (10 pt) (Based off Lecture 15) Draw the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled.

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

```
def f(x):
    x = 4
    w = [x, 5]
    def g(y):
        def h(z):
            nonlocal x
            x = x + y
            w[0] = x
            return x + y + z
        return h
    return g
a = f(1)
b = a(2)
c = b(3) + b(4)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

3. (14 points) Dressing Classy

- (a) (9 pt) For the following class definition, cross out any incorrect or unnecessary **lines** in the following code so that the doctests pass. **Do not cross out class declarations or doctests.** You can cross out any other line of code, including method declarations, and your final code should be as short as possible.

```
class Wardrobe:
    """
    >>> shirts = [Shirt("green"), Shirt("blue")]
    >>> winter = Wardrobe(shirts)
    >>> winter.count
    2
    >>> winter.wear()
    'green'
    >>> shirts[0].stains
    1
    >>> winter.count
    1
    """
    def __init__(self, shirtsList):
        self.shirts = shirtsList
        shirts = shirtsList
    @Attribute
    @property
    def count(self):
        sum = 0
        for i in shirts:
        for i in self.shirts:
            if i.washed:
                sum += 1
        return sum
    def wear():
    def wear(self):
        for s in self.shirts:
        for s in shirts:
            if s.washed:
            if self.washed:
                s.washed = False
                s.stains += 1
                return s.color
        return s.color
class Cloth:
    def __init__(color):
    def __init__(self, color):
        self.color = color
        washed = True
        self.washed = True
class Shirt(Cloth):
    def __init__(color):
    def __init__(self, color):
        self.color = color
        self.stains = 0
        Cloth.__init__(self, color)
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- (b) (5 pt) Implement an iterator for the wardrobe that only iterates over the shirts that have been washed. Fill in the blanks in the implementation of the `__iter__` method for the **Wardrobe** class. For this part, you may assume that the **Wardrobe** class stores the list of shirts in an instance attribute called **shirts** and that the **wear** method of the **Wardrobe** class has been correctly implemented. Each instance of a shirt has an boolean attribute **washed** that indicates whether it has been washed.

```
Class Wardrobe:
    """
    >>> shirts = [Shirt("yellow"), Shirt("green"), Shirt("blue")]
    >>> winter = Wardrobe(shirts)
    >>> winter.wear()
    'yellow'
    >>> for option in winter:
    ...     print(option)
    ...
    'green'
    'blue'
    >>> len(winter.shirts)
    3
    """
    """ Assume the methods from Part A are implemented """
    def __iter__(self):
        for
```

Assignment Project Exam Help

-----:

<https://powcoder.com>

-----:

Add WeChat powcoder

- ```
STk> (deep-reverse '(foo bar baz))
(baz bar foo)
STk> (deep-reverse '(1 (2 3) (4 (5 6) 7)))
((7 (6 5) 4) (3 2) 1)
```

(cond \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

Assignment Project Exam Help \_\_\_\_\_))

- ```
create table airports as
select "SFO" as orig, "NYC" as dest, 500 as price union
select "SFO" , "MAD", 3000 union
select "NYC" , "LHR" , 600 union
select "LHR" , "FRA" , 600 union
select "LHR" , "MAD", 400;
```

start	end	cost	hops
NYC	MAD	1000	2
SFO	LHR	1100	2
NYC	FRA	1200	2
SFO	MAD	1500	3
SFO	FRA	1700	3

```

----- as (
select ----- union
    select -----
    -----
)
select -----;
```

5. (6 points) Interpretation

(From Summer 2014 MT2) Select which function(s) you would have to modify in order to add the new syntax features in Calculator. **For full credit, you must justify your answers with at most two sentences.**

(a) (1 pt) = (equality checker) – e.g. (= 3 1) returns False

calc_eval calc_apply Both Neither

Justification:

(b) (1 pt) or – e.g. (or (= 5 2) (= 2 2) (\ 1 0)) returns True

calc_eval calc_apply Both Neither

Justification:

(c) (1 pt) Creating and calling lambdas (Assume define has been implemented.) – e.g.

```
(define square (lambda (x) (* x x)))
(square 4)
```

calc_eval calc_apply Both Neither

Justification:

(d) (1 pt) Which of the following is not a benefit of Client-Server Architecture

Creates a Abstraction Barrier Server Reuses Computation Both Neither

Justification:

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

```
def g(n):
    if n % 2 == 0 and g(n + 1) == 0:
        return 0
    return 5
```

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

$\Theta(1)$ $\Theta(\log n)$ $\Theta(n)$ $\Theta(n^2)$ $\Theta(b^n)$

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder