CS 61A Spring 2018 — Structure and Interpretation of Computer Programs

MOCK EXAM

INSTRUCTIONS

- You have 3 hours to complete the exam.
- The exam is closed book, closed notes, closed computer, closed calculator, except three hand-written 8.5" × 11" crib sheet of your own creation and the official CS 61A study guides.
- Mark your answers **on the exam itself**. We will *not* grade answers written on scratch paper.

Last name	
First name	
Student ID number	
CalCentral email (`_@berkeley.edu`)	
TA	
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)**	

POLICIES & CLARIFICATIONS

- You may use built-in Python functions that do not require import, such as `min`, `max`, `pow`, `len`, and `abs`.
- You **may not** use example functions defined on your study guides unless clearly specified by the question.
- For fill-in-the blank coding problems, we will only grade work written in the provided blanks. You may only write one Python statement per blank line, and it must be indented to the level that the blank is indented. you may not need every blank, however.
- Unless otherwise specified, you are allowed to reference functions defined in previous parts of the same question.

1. (10 points) WWPD

For each of the expressions in the table below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. The interactive interpreter displays the repr string of the value of a successfully evaluated expression, unless it is `None`. Write "FUNC" to indicate a functional value.

The first two rows have been provided as an example.

Assume that you have started `python3` and executed all the code to the left of the table first.

```
class A:
    a = 5
    def __init__(self, lst, n):
        a = 1
        self.lst = lst
        self.n = n
    def update(self):
        for i in range(len(self.lst)):
            self.lst[i] = self.lst[i] * self.n
        self.a += 1
        print(self.a)

class A2(A):
    a = 3
    def update(self):
        for i in range(len(self.lst)):
            self.lst[i] = self.lst[i] - self.m
        self.a -= 1
        print(self.a)

class B:
    def __init__(self, a):
        self.a = a

c = [3, 5, 6]
a = A(c, 2)
b = A2(c, 3)
c = b
```

Expression	Interactive Output
`[2, 3]`	[2, 3]
`print((2, 3))`	(2, 3)
`a.a`	FUNC [FUNC, None]
`c.a`	FUNC 17 1 1
`a.update() \\ b.update() \\ A.a`	[0, -1, 1, -1]
`A2.a`	[2, [4, [9, 2]]]

`z=4`
`mx(z)`
`print(z)`
	4

`a.lst`	[0, -1, 1, -1]
`B(a).a.a`	[0, -1, 1, -1]
	[0, -1, 1, -1]

2. (10 points) Environment Diagram

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 may not need to use all of the spaces or frames.*

A complete answer will:

- Add all missing names and parent annotations to frames.
- Add all missing values created or referenced during execution.
- Show the return value for each local frame.
- Use box-and-pointer notation for list values. You do not need to write index numbers or the word "list".

3. (12 points) Clever Pun No. 5

Suppose Samo the dog needs to make his way across an n x n grid to get back to Professor DeNero. Samo is a very loyal dog and wants to reach the Professor in the fewest moves possibles, but at the same time, Samo is an opportunist, and notices treats scattered throughout the grid.

Suppose Samo starts at location (0, 0) on the grid G and Professor DeNero is at location (n, n) on the grid; that is, they are on opposite corners. Our input grid G tells us how many treats are at any location - for a location (x, y), the number of treats in that location can be found with G[x][y]. Given that Samo can move up, down, left, or right (no diagonals), and that Samo will eat all the treats in a location as he leaves it, fill in the function trail_of_treats to return the maximum amount of treats Samo can eat if he takes the minimum moves to get to Professor DeNero. (Samo will also eat all the treats at Professor DeNeros location when he reaches it.)

```
def trail_of_treats(G):
    return trail_helper(G, 0, 0)

def trail_helper(G, x, y):
    if x == len(G) or y == len(G):
        return G[x][y]
    else:
        a = trail_helper(G, x + 1, y)
        b = trail_helper(G, x, y + 1)
        return max(a, b) + G[x][y]
```