CS 61A Structure and Interpretation of Computer Programs

Spring 2017 Mock Midterm 1

INSTRUCTIONS

- You have 1 hour to complete the exam.
- The exam is closed book, closed notes, closed computer, closed calculator, except one 8.5” × 11” cheat sheet of your own creation.
- Mark your answers **on the exam itself.** We will *not* grade answers written on scratch paper.

Last name	Evan Ng
First name	
Student ID number	
Instructional account (cs61a-_)	
BearFacts 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)**	

1. (10 points) Send Help

```
b = 5 def x(x, help):
                def send(help):
             print(help) return
        nope def nope(send):
            print(send) return
                            help
       a = send(help) def help():
                        return x("help")

       print("More") return a
```

Expression	**Interactive Output**
print(print(print("hi")))	Hi None none
print((lambda x: print("which")) (print("is")), "first")	Is Which None first
a = print("hi")()	Hi error
b = a	error
print(b)	5
x = x(lambda x: print("oh dear"), print("which print"))	Which print None more
x = x("almost")	almost
print(x())	Oh dear error

Name: ______________________________

2. (10 points) **Stuk In Lambda's**

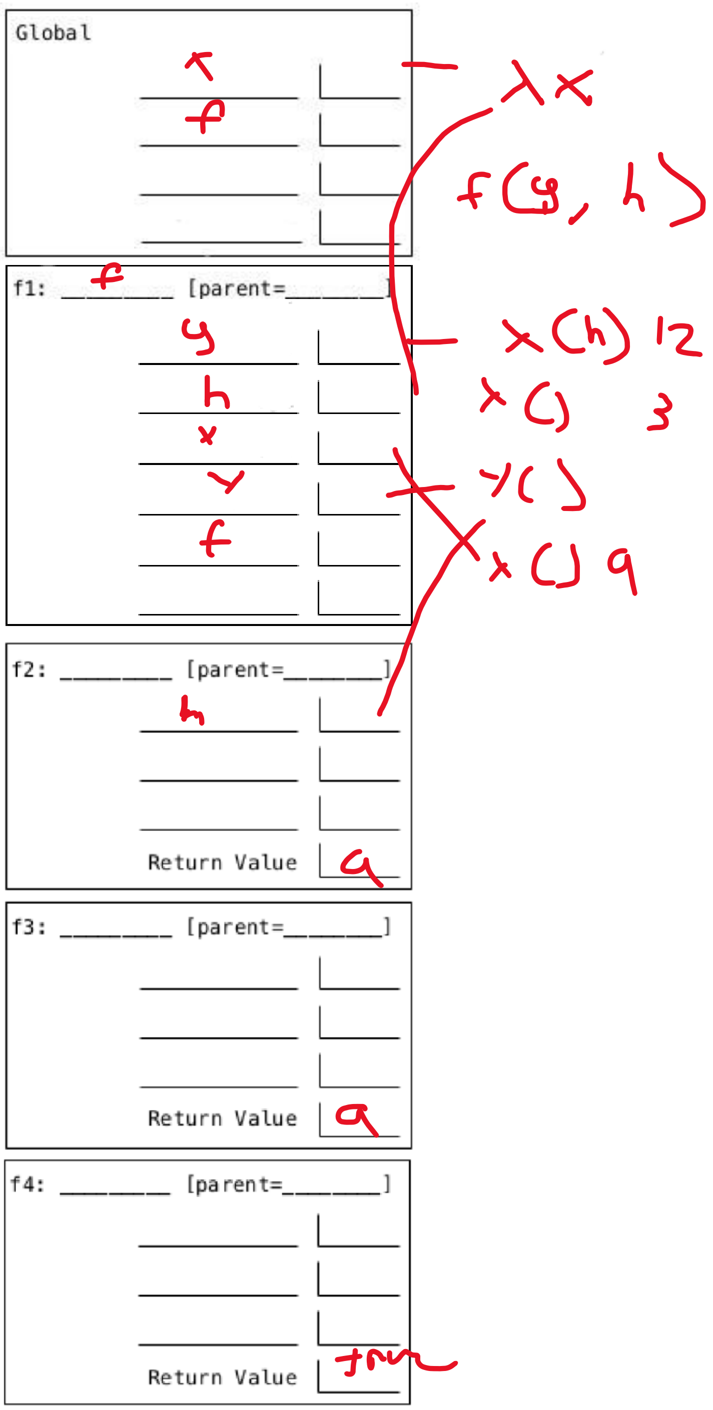
Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. *You may not need to use all of the spaces or frames.*

A complete answer will:

- Add all missing names and parent annotations to all frames.
- Add all missing values created or referenced during execution.

```
x = lambda x: x
def f(g, h):
    x = lambda: True
    def y():
        if f():
            return 'a'
        else:
            return 'b'
    f, x = x, lambda: False
    return g(y)

f(lambda h: h(), x)
```



- Show the return value for each local frame.

3. (10 points) Gib-me the Words!

Did you know that in Python, you can access elements in a string exactly like you access elements in a list? For example, if we were given the string, Gibbes:

(a) Assigning to the variable g, allows us to access the ’G’ with g[0].

(b) Slicing and concatenation are valid: g[3:] + ’t’ evaluates to ’best’.

Write a function that follows the specs below by creating a list of words out of a string, where a word has no spaces (’ ’). DO NOT USE LEN.

You may only use the lines provided. You may not use any Python built-in sorting functions.

```
def wordify(s):
    """ Takes a string s and divides it into a list of words.
    Assume that the last element of the string is a whitespace. Duplicate words allowed.
              >>> wordify( sum total of human knowledge                    )
    [’sum’, ’total’, ’of’, ’human’, ’knowledge’]
    >>> wordify(’one should never use exclamation points in writing! ’)
    [’one’, ’should’, ’never’, ’use’, ’exclamation’, ’points’, ’in’, ’writing!’]
    """ start, end, lst = 0, 0 for letter in ___________________:

        if letter != ’ ’:
            end = end +1

        elif start == end:
            start, end = start +1, end+1
        else: lst += [s] start, end = end,+1 , end + 1 return lst
```

4. (10 points) Walt is Relevant

We have a list of numeric data points *l*, and we want to see if a list of relevant numbers *s* is found in the data. The catch is, we want to see if the numbers in *s* occur in the same order within the data of *l*, though not necessarily one after the other. If so, then *s* is a subsequence of *l*.

Write a predicate function subseq that takes two lists *l* and *s* as arguments, and determines if *s* is a subsequence of *l*. If so, the function should return True; otherwise, it should return False. We have provided a few doctests to demonstrate the definition and usage.

```
def subsequence(l, s):
    """
                  Returns true if s is   a subsequence of l.

    >>> subsequence([9,              1, 4, 5, 6], [4, 5, 6])
    True
    >>> subsequence([3,              5, 0, 3, 4, 3, 7, 9, 3,                          2],     [3,     3, 9,       2])
    True

    >>> # Below, the numbers in seq2 >>> # but not in appear      in seq1,
    the same order.
    >>> subsequence([3, 5, 5, 8, 3],                             [8, 5,         3])
    False
    >>> # Below, not all the numbers                             in seq2        are present          in     seq1.
    >>> subsequence([3, 5, 5, 8, 3],                             [3, 2,         8])
    False
    >>> subsequence([3, 2, 57, 8], [3, 5, 7])
    False
    """
    if s == []:
        return true  elif l == []:

        return false elif l[0] ==s[0]

        return subsequence(1[1:], s[1:])

    else: return subsequence(1[1:], s)
```

5. (0 points)   Games of Berkeley

In the box below, write a positive integer. The student who writes the lowest unique integer will receive one extra credit point. In other words, write the smallest positive integer that you think no one else will write.