

INSTRUCTIONS

- You have 2 hours and 50 minutes 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
First name	Ng
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	
<i>All the work on this exam is my own.</i> (please sign)	

1. (6 points) Olympic Games

- (a) (4 pt) For each of the statements below, write the output displayed by the interactive Python interpreter when the statement is executed. The output may have multiple lines. **No answer requires more than four lines.** The first two have been provided as examples.

Assume that you have started python3 and executed the following statements:

```
gold = [0, 1]

def go(dream, team): print('Steps to success:')
    def medal(lion): return dream(team, lion)
    return medal

def win(big):
    print('Eat vegetables.') return big[-1] + big[-2]

def get(it, done): do = it while
    done < 2:
        print("Don't cheat!") do, done = do + [win(do)], done
        + 1
    return do
```

```
>>> print(4, 5) + 1
4 5
Error
```

```
>>> gold[-2]
0
```

```
>>> win(gold)
```

```
>>> riooo = go(get, gold)
Steps to success
```

```
>>> usa = riooo(0)
Don't cheat!
Eat vegetables
Don't cheat
Eat vegetables
```

```
>>> usa
0,1,1,2
```

(b) (2 pt) What would be the return value of `rio00(-2)`, after evaluating the expressions above?

[0, 1, 1, 2, 3, 5]

2. (6 points) The Name Game

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.
- Show the return value for each local frame.

```

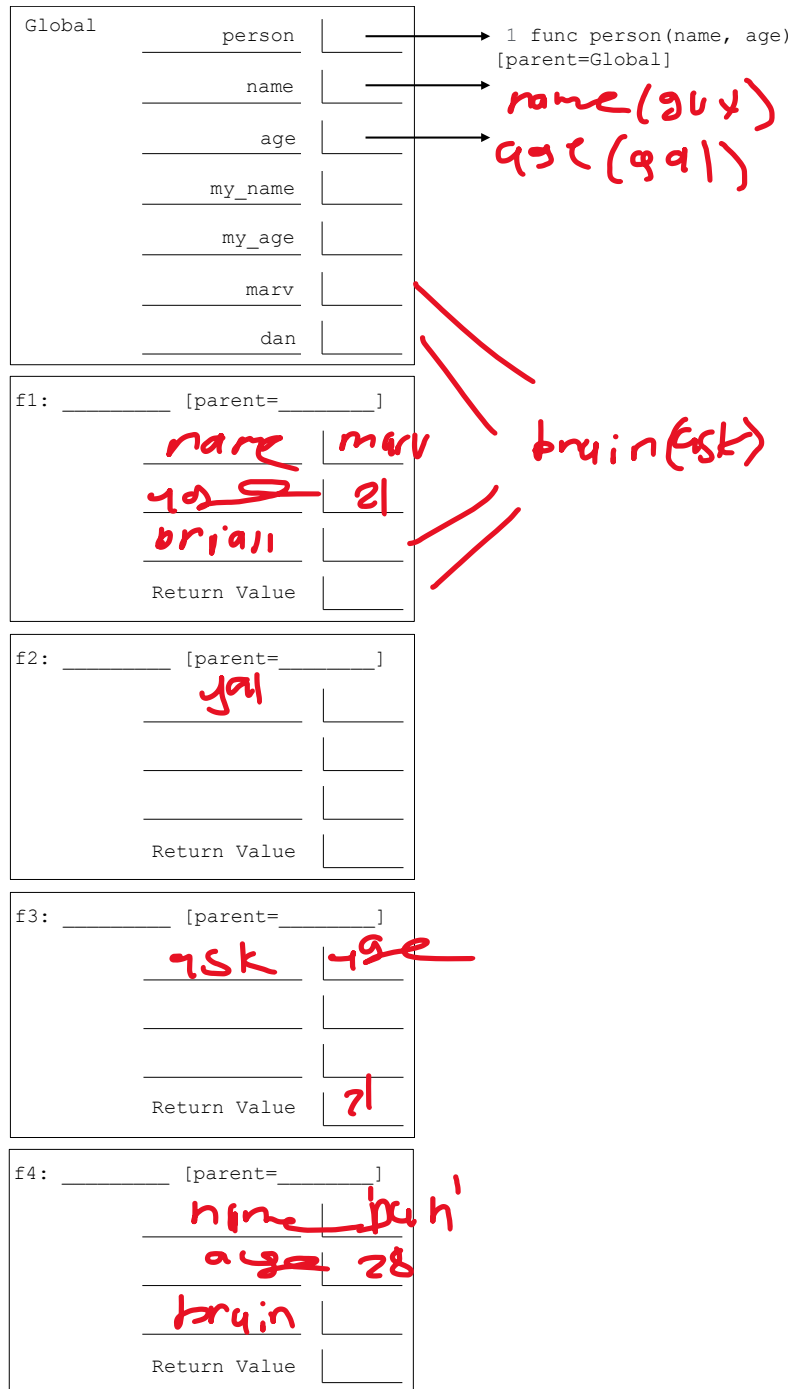
def person(name, age):
    def brain(ask):
        if ask == 'name':
            return name
        elif ask == 'age':
            return age
        return brain
    return brain

def name(guy):
    return guy('name')

def age(gal):
    return gal('age')

my_name, my_age = 'Marv', 21
marv = person(my_name, my_age)
dan = person('Dan', age(marv) + 7)

```



2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

3. (3 points) High Scores

Write expressions involving the list `lst` that evaluate to the following values. Your expressions **must** use `lst`. For the second and third lines, you **must** use a list comprehension.

```
>>> lst = [[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]
```

```
>>> len(lst)
```

```
5
```

```
>>> [[11-x, 11-y] for x, y in lst]
```

```
[[10, 9], [8, 7], [6, 5], [4, 3], [2, 1]]
```

```
>>> [2*x for x in lst[0]]
```

```
[2, 4]
```

4. (4 points) Kerbal Space Program

Define the function `is_sorted` that takes in a non-negative integer `n` and returns `True` if the digits of `n` are in non-increasing order from left to right, and `False` otherwise. See the doctests for details.

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

```
def is_sorted(n):
```

```
    """Returns whether the digits in n are in non-increasing order from left to right.
```

```
    >>> is_sorted(4)
```

```
    True
```

```
    >>> is_sorted(55555)
```

```
    True
```

```
    >>> is_sorted(9876543210) # blastoff!
```

```
    True
```

```
    >>> is_sorted(9087654321)
```

```
    False """ if n<10: return true
```

```
    elif return false
```

```
    else: return is_sorted(n// 10)
```

5. (6 points) Katamari

- (a) (4 pt)** Define the function `aggregate` that takes in a two-argument function `fn`, a sequence `seq`, and a predicate function `pred` and returns the result of using `fn` to combine the elements in `seq` for which `pred` returns `True`. See the doctests for details. You may assume `fn` is commutative and never returns `None`.

You may only use the lines provided. You may not need to fill all the lines. You may not use an import statement.

```
def aggregate(fn, seq, pred):
```

```
    """Aggregates using fn the elements in seq that satisfy pred.
```

```

>>> def is_even(x):
...     return x % 2 == 0
>>> def sum_plus_one(x, y):
...     return x + y + 1
>>> aggregate(sum_plus_one, [2, 4, 6], is_even) # (2 + 4 + 1) + 6 + 1
14
>>> # If no elements satisfy pred, return None
>>> aggregate(sum_plus_one, [1, 3, 5, 7, 9], is_even)
>>> # If only one element satisfies pred, return that element
>>> aggregate(sum_plus_one, [1, 2, 3], is_even)
2 """ result = None

```

For elem in seq:

```

    If
pred(elem):

    If result ==
none:

Result== elem

```

return result

- (b) (2 pt)** Use the aggregate function above, which you can assume works correctly, to define the fact function that returns the factorial of non-negative integers. **Use only one line of code. If you need more space, you can continue your line of code on the second blank.** The add and mul functions, which may be useful, have been imported for you.

Hint: You may need the ternary operator <expr1> if <cond> else <expr2>.

```
>>> from operator import add, mul
```

```
>>> fact = lambda n: aggregate(mul, range(1, n+1), lambda x:true) if n > 0 else 1
```

```
>>> fact(0) 1
```

```
>>> fact(5)
```

```
120
```

6. (6 points) Legend of Zelda

This question uses the following linked list data abstraction. We have provided an example linked list.

```

link(3)))
>>> def link(first, rest=empty):
...     return [first, rest]

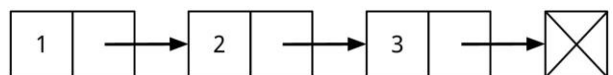
...
>>> def first(lnk):
...     return lnk[0]

>>> def rest(lnk):

```

```
>>> empty = 'empty'
```

```
>>> link(1, link(2,
```



```
...         return lnk[1]
```

Define the function `linked sum` that takes in a linked list of positive integers `lnk` and a non-negative integer `total` and returns the number of combinations of elements in `lnk` that sum up to `total`. You may use each element in `lnk` zero or more times. See the doctests for details. **Do not violate abstraction barriers.**

You may only use the lines provided. You may not need to fill all the lines.

```
def linked_sum(lnk, total):
```

```
    """Return the number of combinations of elements in lnk sum up to total.           that
```

```
    >>> # Four combinations: 1 1 1 1 , 1 1 2 , 1 3 , 2 2
```

```
    >>> linked_sum(link(1, link(2, link(3, link(5)))), 4)
```

```
    4
```

```
    >>> linked_sum(link(2, link(3, link(5))), 1)
```

```
    0
```

```
    >>> # One combination: 2 3
```

```
    >>> linked_sum(link(2, link(4, link(3))), 5)
```

```
    1
```

```
    """
```

```
    if total == 0: return 1 elif lnk == empty or total < 0 return 0 else:
```

```
        with_first = linked_sum(lnk, total - first(lnk)) without_first = without
```

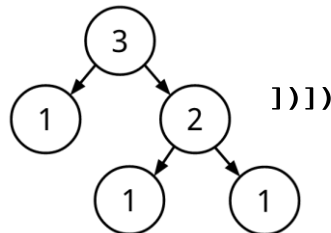
```
        first=linked_sum(rest(lnk),total return with_first + without_first
```

7. (9 points) Game of Thrones

This question uses the following tree data abstraction. We have provided an example tree.

```
>>> def tree(entry, children=[]):
...     return [entry, children]
...
...     tree(2, [
>>> def entry(tree):tree(1), tree(1)
...     return tree[0]
...
...
>>> def children(tree):
...     return tree[1]
```

```
>>> tree(3, [
...     tree(1),
```



- (a) (6 pt)** Define the function `track lineage` that takes in a tree of strings `family tree` and a string `name`. Assume that there is a unique node with entry name. `track lineage` returns a list with the entries of the parent and grandparent of that node.¹ If the node with entry name does not have a parent or grandparent, return `None` for that element in the list. See the doctests for details. **Do not violate abstraction barriers.**

You may only use the lines provided. You may not need to fill all the lines.

```
def track_lineage(family_tree, name):
```

```
    """Return the entries of the parent and grandparent of the node with entry name in
    family_tree.
```

```
    >>> t = tree('Tytos', [
```

¹ The grandparent of a node is the parent of the node's parent.

```

...         tree('Tywin', [
...             tree('Cersei'), tree('Jaime'), tree('Tyrion')
...         ]),
...         tree('Kevan', [
...             tree('Lancel'), tree('Martyn'), tree('Willem')
...         ]))

```

```

>>> track_lineage(t, 'Cersei')
['Tywin', 'Tytos']
>>> track_lineage(t, 'Tywin') ['Tytos', None]
>>> track_lineage(t, 'Tytos')
[None, None]

```

```

""" def tracker(t, p, gp):

```

```

    if name==entry(t):

```

```

    return (p, gp)

```

```

    for c in children(t):

```

```

    res = tracker(c, entry(t), p)

```

```

    if res:

```

```

    return res

```

```

    tracker(family_tree, None, None) (b) (3 pt) Assuming that track lineage works correctly,

```

define the function are_cousins that takes in a tree of strings family tree and two strings name1 and name2 and returns True if the node with entry name1 and the node with entry name2 are cousins in family tree. Assume that there are unique nodes with entries name1 and name2 in family tree. See the doctests for details.

Two nodes are cousins if they have the same grandparent but different parents. **You may only use the lines provided. You may not need to fill all the lines.**

```

def are_cousins(family_tree, name1, name2):

```

```

    """Return True if a node with entry name1 is a cousin of a node with entry name2 in family_tree.

```

```

    >>> are_cousins(t, 'Kevan', 'Tytos') # same tree as before

```

```

    False

```

```

    >>> are_cousins(t, 'Cersei', 'Lancel')

```

```

    True

```

```

    >>> are_cousins(t, 'Jaime', 'Lancel') True

```

```

    >>> are_cousins(t, 'Jaime', 'Tyrion') False

```

```

    """

```

```

    P1, gp1 = track_lineage(family_tree, name1)

```


Name: _____

9

P2,gp2=track_lineage(family_tree,name2)

8. (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.