Midterm 1 Review Worksheet

1. Rappers from Dunder Mifflin

For each of the expressions in the tables below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. If an error occurs, write "Error". The first two rows have been provided as examples. Recall: The interactive interpreter displays the value of a successfully evaluated expression, unless it is None. Assume that you have started python3 and executed the following statements:

```
def desiigner(animal = 7):
    if animal == 4:
        return 6
    if animal > 0:
        print "panda"
        return desiigner(animal – 1)
    else:
        return "friends in atlanta"

def drake(sadboy):
    if sadboy > 5:
        return desiigner
    else:
        return sadboy or "love hurts"

def kanye(west):
    protege = desiigner
    if west > 10:
        for i in range(50):
            print("I love kanye")
    return protege(west)

bears, beets, battlestar_galactica = 2, 4, 61
```

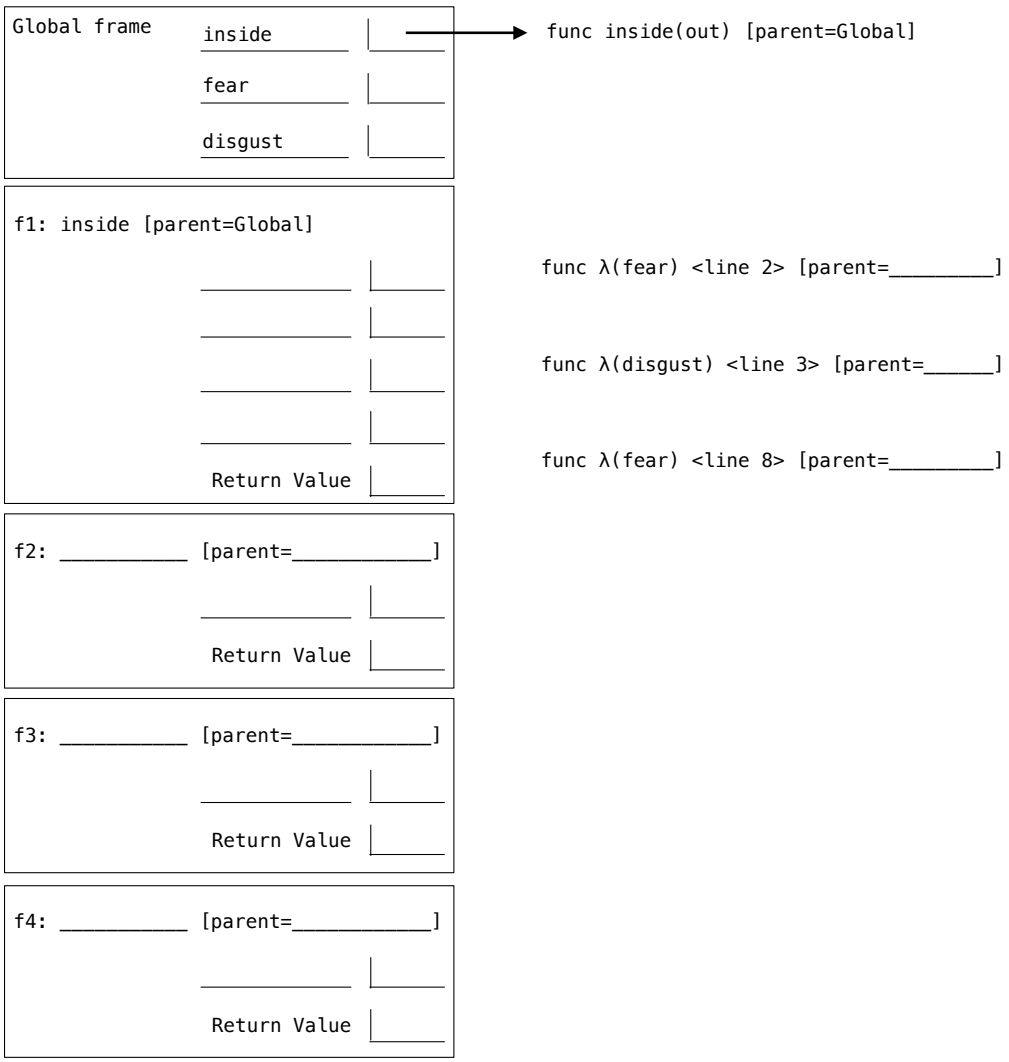
Expression	Interactive Output
`pow(2, 3)`	8
`print(4, 5) + 1`	4 5 Error
`print(None, print(None))`	
`kanye(bears)`	
`beets = desiigner(beets)`	
`drake(beets)(bears)`	
`drake(bears)`	

2. Copy Cat *(this is question 2b from Fall 2015 Midterm 1)*

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. The <line ...> annotation in a lambda value gives the line in the Python source of a lambda expression. A complete answer will:

- Add all missing names and parent annotations to all local frames
- Add all missing values created or referenced during execution
- Add all missing parents of function values
- Show the return value for each local frame

```
1 def inside(out):
2     anger = lambda fear: fear(disgust)
3     fear = lambda disgust: anger(out)
4     disgust = 3
5     fear(5)
6
7 fear, disgust = 2, 4
8 inside(lambda fear: fear + disgust)
```



3. ♪ ~ Mary had a little lambda whose fleece was list comprehension ~ ♪

For each blank, write a line of code that will produce the output shown. You may not explicitly multiply or square any numbers in the second blank, i.e., your solution for the second line should not contain * or **. (hint: for the first line, use a lambda function)

```
>>> lamb = ____________________________________________________________________
```

```
>>> [lamb(6), lamb(7)]
[36, 49]
```

```
>>> lst = [1, 2, 3, 4, 5]
```

```
>>> ____________________________________________________________________
[1, 4, 9, 16, 25]
```

4. Linked(in) Lists

We're right about halfway through this course, and some of you may be wondering how you can stay in touch with your awesome instructors. Maybe you can find them on linkedin, or maybe you can solve this linked list question and impress them with your newly developed CS skills.

For each blank, write a line of code that would produce the desired output shown below. Assume that the linked list constructors/selectors have been defined.

```
>>> net = link("marvin zheng", link("martin zhang", link("marvin zhang")))
```

```
>>> ____________________________________________________________________
'marvin zhang'
```

```
>>> work = link("brian house", link("brian hou", link("brian who")))
```

```
>>> ____________________________________________________________________
'brian hou'
>>> home = link("unit3", link("berkeley hou-zhang", link("brian house")))
```

```
>>> ____________________________________________________________________
'house'
```

5. Sometimes we like trees (unless they grow in Palo Alto)

Implement the function `tree_min`, which takes in a tree and a depth and returns the smallest node in the tree up to that depth.

```
def tree_min(t, d):

    """Return the smallest node in t, checking up to depth d.

    >>> my_tree = tree(5, [tree(2, [tree(4), tree(1)]), tree(3)])
    >>> tree_min(my_tree, 1)
    2
    >>> tree_min(my_tree, 0)
    5

    """

    if  ____________________________________________________________________ :

        return  _________________________

    else:

        smallest =  ____________________________________________________________

        return  _________________________
```