CS 7 Introduction to Programming and Computer Science

SUMMER 2021	PRACTICE MIDTERM
EXAM VERSION	AIR NOMADS
DATE	MAY 29
TIME	6 TO 8 PM CAT

INSTRUCTIONS

- You have 2 hours to complete the exam.
- The exam is open book, open notes, closed computer. You may consult any books, notes, or other non-responsive objects available to you.
- There are 4 questions in this exam all worth 40 points. The midterm is worth 10 percent of the total grade.
- Answer on the separate answer sheet. You may use a scratch for your work but make sure to transfer the solutions to the answer sheet. Work not in the answer sheet will not be graded.
- After completing this exam, you will have 10 minutes to scan and upload your answer sheet to the midterm assignment on Gradescope

Be warned: Computer Science exams are known to cause panic. Fortunately, this reputation is entirely unjustified. Just read all the questions carefully to begin with and first try to answer those parts about which you feel most confident. Do not be alarmed if some of the answers are obvious. Should you feel an attack of anxiety coming on, feel free to jump up and run around the outside of your building once or twice.

1. (12 points) Fundemic 😷

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. If an error occurs, write `"Error"`.
Hint: No answer requires more than 5 lines. (It's possible that all of them require even fewer.)
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 corona(virus):
    if virus < mask :
        return virus * 3
    else :
        return corona(virus // 2) + 3

def mandate(put, on, mask=7):
    while put:
        put , mask = on(put) , print(put, mask)
    return lambda mask : print(put , mask)

virus , mask = 2 , 3
```

Question	Expression	Interactive Output
	`pow(3, 4)`	`81`
	`print(2, 0) + 21`	`2 0` `Error`
A	`print(None, print(None))`	
B	`corona(3)`	
C	`(lambda mask: corona(8))(9)`	
D	`mandate(2, lambda x: x -2)(4)`	
E	`mandate(5, print)(mask)`	
F	`mandate(6, lambda virus: virus - mask,2)(-1)`	

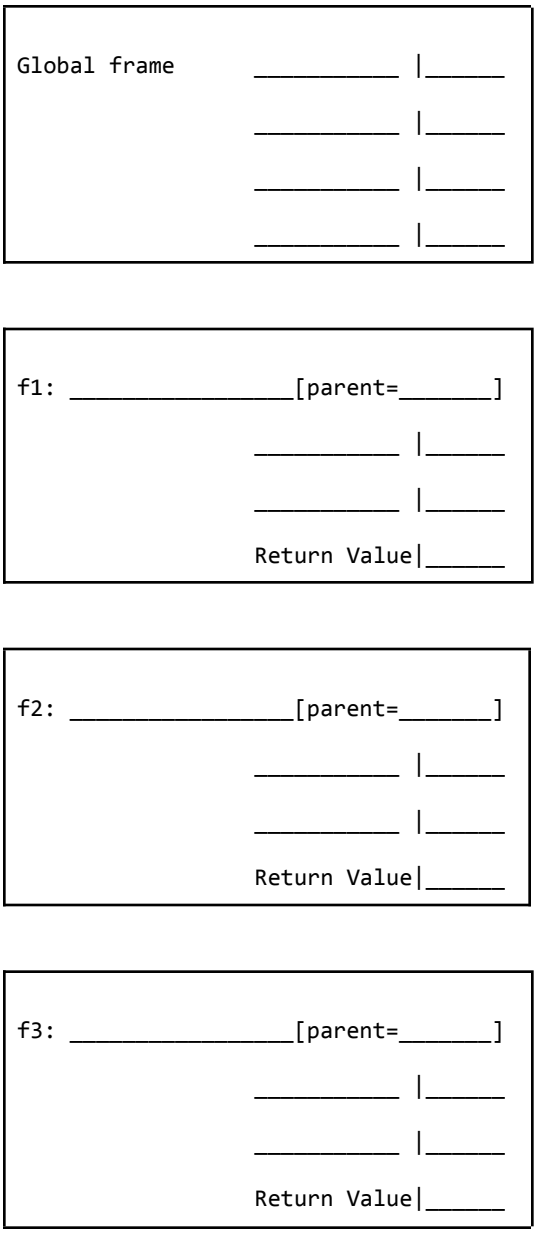
2. (12 points) Batlao Hurda 💔

a. **(6 pt)** 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:

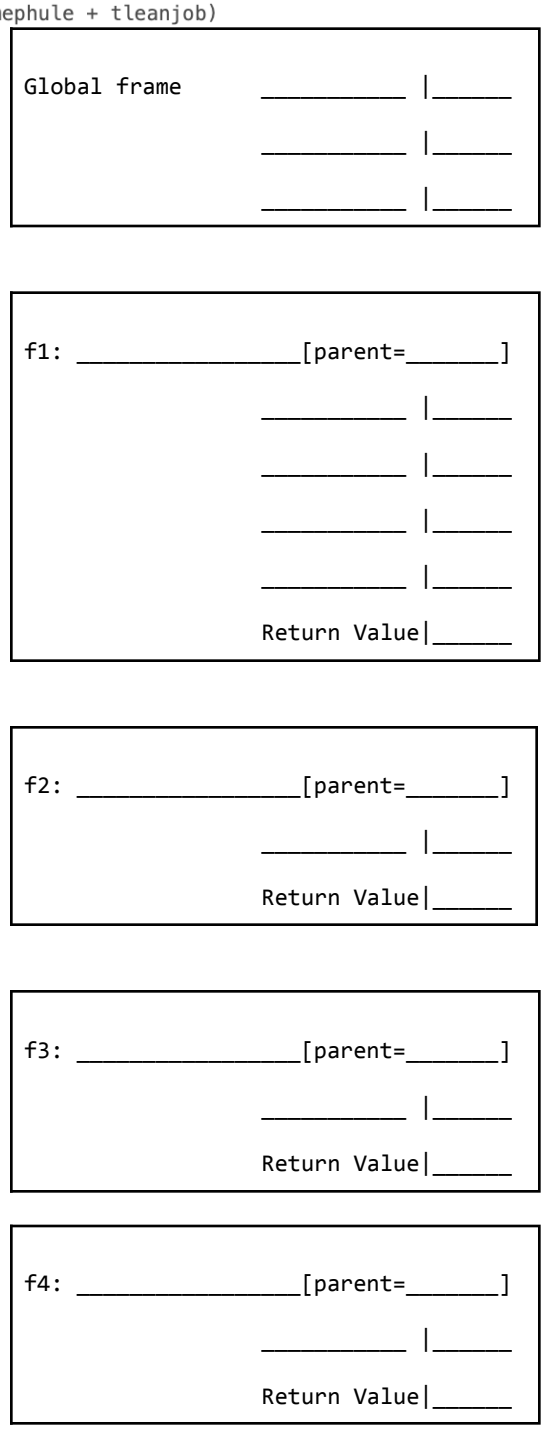
- Add all missing names and parent annotations to all local frames.
- Add all missing values created or referenced during execution.
- Show the return value for each local frame.

```
1  def clumsy(heart):
2      return heart - 7
3
4  def batlao(hurda):
5      def clumsy(they):
6          return heart * 3
7      def hurda(wena):
8          heart = wena + 12
9          return clumsy(4) - 3
10     return hurda
11
12 heart, flee = 2, batlao(clumsy)
13 flee = flee(8)
```



b. **(6 pt)** 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.

```
1  def mjolo(fm):
2      ambulance = lambda mephule: mephule(tleanjob)
3      mephule = lambda tleanjob: ambulance(fm)
4      tleanjob = 3
5      mephule(5)
6
7  mephule, tleanjob = 2, 4
8  mjolo(lambda mephule: mephule + tleanjob)
```



3. (9 points) Don't take off your jacket! Math's not hot (Never hot) 🔥

a. **(6 pt)** Implement `counter`, which takes a non-negative single-digit integer `d`. It returns a function count that takes a non-negative integer `n` and returns the number of times that `d` appears as a digit in `n`. You may not use recursive calls or any features of Python not yet covered in the course.

```
def counter(d):
    """Return a function of N that returns the number of times D appears in N.
    >>> counter(8)(8018)
    2
    >>> counter(0)(2016)
    1
    >>> counter(0)(0)
    0
    """
    def count(_____________________________A_____________________________):

        k = 0

        while _____________________________B_____________________________:


           __________C___________, last = _____________D___________, n % 10


            if _____________________________E_____________________________:


                _____________________________F_____________________________


        _____________________________________G_____________________________


    _____________________________________H_____________________________
```

b. **(3 pt)** Implement `significant`, which takes positive integers `n` and `k`. It returns the `k` most significant digits of `n` as an integer. These are the first k digits of `n`, starting from the left. If `n` has fewer than `k` digits, it returns `n`. You **may not** use `round, int, str`, or any functions from the math module. You may use `pow`, which raises its first argument to the power of its second: `pow(9, 2)` is 81 and `pow(9, 0.5)` is 3.0.

```
def significant(n, k):
    """Return the K most significant digits of N.
    >>> significant(12345, 3)
    123
    >>> significant(12345, 7)
    12345
    """
    if _____________________________A_____________________________:

        return n

    return significant(____________B___________, _________C_________ )
```

4. (7 points) S'poko tha Ghost 👻

****IMPORTANT**** In this question, assume that all of f, g, and h are functions that take one non-negative integer argument and return a non-negative integer. You do not need to consider negative or fractional numbers

a. **(4 pt)** Implement the higher-order function `decompose1`, which takes two functions `f` and `h` as arguments. It returns a function `g` that relates `f` to `h` in the following way: For any non-negative integer `x`, `h(x)` equals `f(g(x))`. Assume that `decompose1` will be called only on arguments for which such a function `g` exists. Furthermore, assume that there is no recursion depth limit in Python.

```
def decompose1 (f , h ):
    """ Return g such that h(x) equals f(g(x)) for any non - negative integer x.
    >>> add_one = lambda x: x + 1
    >>> square_then_add_one = lambda x: x * x + 1
    >>> g = decompose1 ( add_one , square_then_add_one )
    >>> g (5)
    25
    >>> g (10)
    100
    """
    def g ( x ):

        def r ( y ):

            if _____________________________________A_____________________________________:


                return _____________________________________B_____________________________________

            else:

                return _____________________________________C_____________________________________

        return r (0)


_____________________________________________D_____________________________________________
```

b. **(3 pt)** Write a number in the blank so that the final expression below evaluates to 2021. Assume decompose1 is implemented correctly. The `make_adder` and `compose1` functions appear on the left column of page 2 of your study guide.

```
e , square = make_adder (1) , lambda x : x * x

decompose1 (e , compose1 ( square , e ))(3) + ____________________A____________________
```

c. **(1 pt)** What Does Momo Give Wan Shi Tong in Order to Enter His Library?

_____