CS 7 Introduction to Programming and Computer Science

SUMMER 2021	MIDTERM
EXAM VERSION	FIRE NATION
DATE	MAY 29
TIME	4 TO 6 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) Skhosana Buhlungu 🎸

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:

```
from operator import add

def ra(sta):
    return print(sta , sta)

def isomi(so):
    lithi, tshayina = ra , print
    tshayina(lithi(so))
    return vara(so)

def vara(vara):
    if vara:
        return vara + vara
    elif lithi(vara)(print)(print):
        return 1000
    else :
        return tshayina(3)

tshayina = vara
lithi = lambda v : lambda a : lambda r : r (5 , a (v))
```

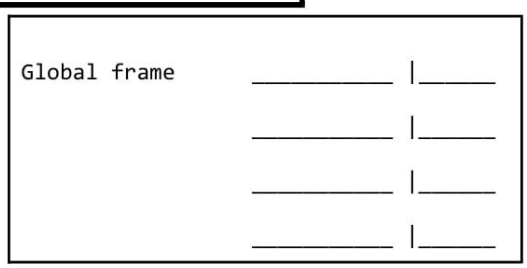
Question	Expression	Interactive Output
	`pow(3, 4)`	`81`
	`print(2, 0)`	`2 0`
A	`print(ra(1+2), print(4))`	
B	`isomi(3)`	
C	`isomi(tshayina(2))`	
D	`lithi(1)(vara)(pow)`	
E	`vara(print(1))`	
F	`lithi(0)(tshayina)(add)`	

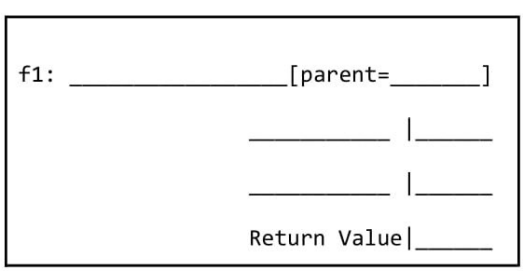
2. (12 points) Khuza Gogo 💃

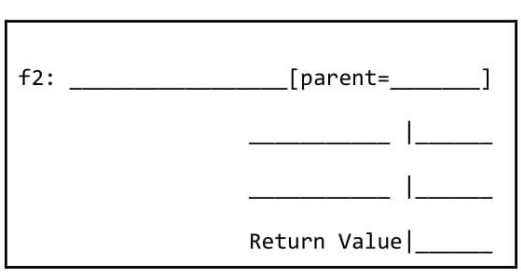
a. **(6 pt)** Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You may not need to use all of the spaces or frames.
A complete answer will:

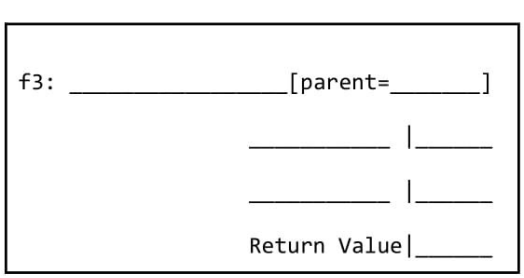
- Add all missing names and parent annotations to all local frames.
- Add all missing values created or referenced during execution.
- Show the return value for each local frame.

```
1  def go(go):
2      ye = 20
3      if go(2) > ye:
4          return go
5  nandi = lambda happy: (lambda birth: day+3)(ye+1)
6  ye, day = 19, 18
7  ye = go(nandi)
```



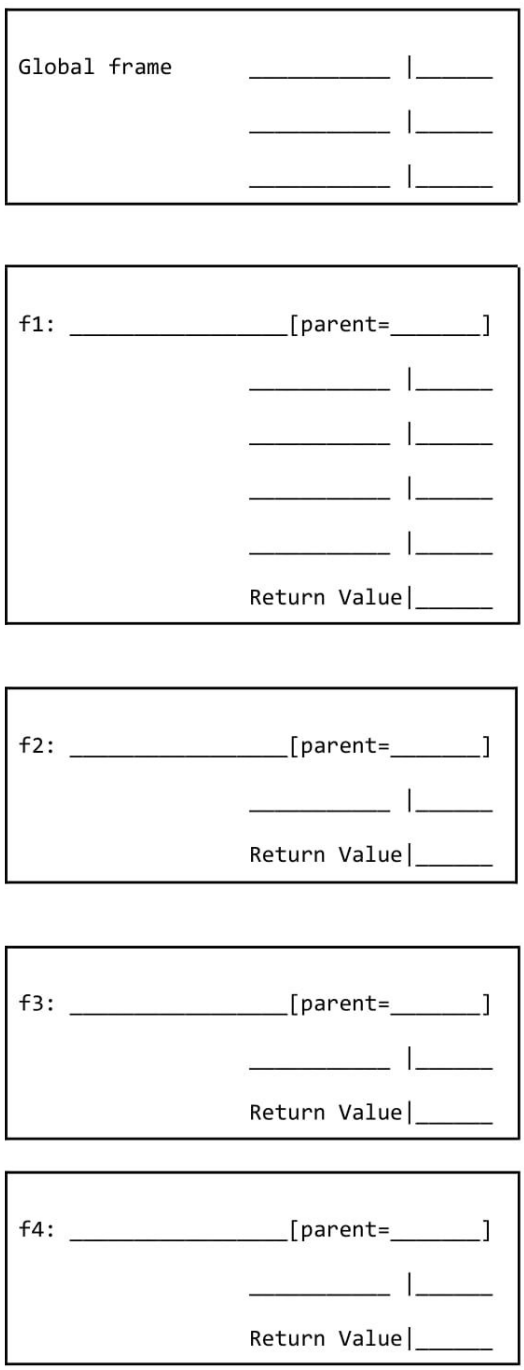






b. **(6 pt)** Fill in the environment diagram that results from executing the code below until the entire program is finished, an error occurs, or all frames are filled. You may not need to use all of the spaces or frames. The<line ...> annotation in a lambda value gives the line in the Python source of a lambda expression.

```
1  def khuza(gogo):
2      def gogo(gogo):
3          return lambda x: gogo
4      return gogo(5)
5
6  def jaiva(g):
7      return g(ogo)(4)
8
9  ogo = 3
10 jaiva(khuza)
```



3. (9 points) New General Mathematics Book Two 🧮

a. **(5 pt)** Implement `rect`, which takes two positive integer arguments, `perimeter` and `area`. It returns the integer length of the longest side of a rectangle with integer side lengths ***w*** and ***h*** which has the given `perimeter` and `area`. If no such rectangle exists, it returns `False`.

The perimeter of a rectangle with sides ***w*** and ***h*** is ***2w + 2h***. The area is ***w· h***.

Hint: The built-in function `round` takes a number as its argument and returns the nearest integer. For example, `round(2.0)` evaluates to 2, and `round(2.5)` evaluates to 3.

```
def rect(area, perimeter):
    """Return the longest side of a rectangle with AREA and PERIMETER that has integer sides.
    >>> rect(10, 14) # A 2 x 5 rectangle
    5
    >>> rect(5, 12) # A 1 x 5 rectangle
    5
    >>> rect(25, 20) # A 5 x 5 rectangle
    5
    >>> rect(25, 25) # A 2.5 x 10 rectangle doesn't count because sides are not integers
    False
    >>> rect(25, 29) # A 2 x 12.5 rectangle doesn't count because sides are not integers
    False
    >>> rect(100, 50) # A 5 x 20 rectangle
    20
    >>> rect(5, 11)
    False
    >>> rect(4, 11)
    False
    """
    side = 1

    while side * side _____A_____ area:

        other = round(_____B_____)

        if _____C_____:

            _____D_____

        side = side + 1

    return False
```

b. **(4 pt)** The `if_fn` returns a two-argument function that can be used to select among alternatives, similar to an if statement. Fill in the return expression of `factorial` so that it is defined correctly for non-negative arguments. **You may only use the names** `if_fn, condition, a, b, n, factorial, base,` **and** `recursive` **and parentheses in your expression (no numbers, operators, etc.).**

```
def if_fn(condition):
    if condition:
        return lambda a, b: a
    else:
        return lambda a, b: b



def factorial(n):
    """ Compute N! for non - negative N. N! = 1 * 2 * 3 * ... * N.
    >>> factorial (3)
    6
    >>> factorial (5)
    120
    >>> factorial (0)
    1
    """
    def base():

        return 1

    def recursive():

        return n * factorial(n-1)


    return _____A_____
```

4. (7 points) Mlung' omnyama 🙏🏾

a. **(4 pt)** Implement the `longest_increasing_suffix` function, which returns the longest suffix (end) of a positive integer that consists of strictly increasing digits.

```
def longest_increasing_suffix(n):
    """ Return the longest increasing suffix of a positive integer n.
    >>> longest_increasing_suffix(63134)
    134
    >>> longest_increasing_suffix(233)
    3
    >>> longest_increasing_suffix(5689)
    5689
    >>> longest_increasing_suffix(568901) # 01 is the suffix , displayed as 1
    1
    """
    m , suffix , k = 10 , 0 , 1
    while n :
        ____________________A____________________ , last = n // 10 , n % 10

        if ____________________________________B____________________________________:

            m , suffix , k = _______C_________ , __________D_________ , 10 * k
        else :
            return suffix
    return suffix
```

b. **(3 pt)** Implement the `combine` function, which takes a non-negative integer `n`, a two-argument function `f`, and a number `result`. It applies f to the first digit of `n` and the result of combining the rest of the digits of n by repeatedly applying `f` (see the doctests). If `n` has no digits (because it is zero), combine returns result..

```
from operator import add , mul

def combine(n, f, result):
    """ Combine the digits in non - negative integer n using f.
    >>> combine(3, mul, 2) # mul(3, 2)
    6
    >>> combine(43, mul, 2) # mul(4, mul(3, 2))
    24
    >>> combine(6502, add, 3) # add(6, add(5 ,add(0, add(2, 3)))
    16
    >>> combine(239, pow, 0) # pow(2, pow(3 ,pow(9, 0)))
    8
    """
    if n == 0:
        return result
    else :
        return combine(_________A_________, ________B________, __________C_________)
```

c. **(1 pt)** What does Emzini weCode mean?