Exam Prep Section 5 Sols. - CS61A Spring 2018

Worksheet 4: Nonlocal and Object-Oriented Programming

Vulcans - Nonlocal Environment Diagram (Spring 2015 Midterm 2 Q2)
Video Walkthrough: https://www.youtube.com/watch?v=cBemCW_uV8g

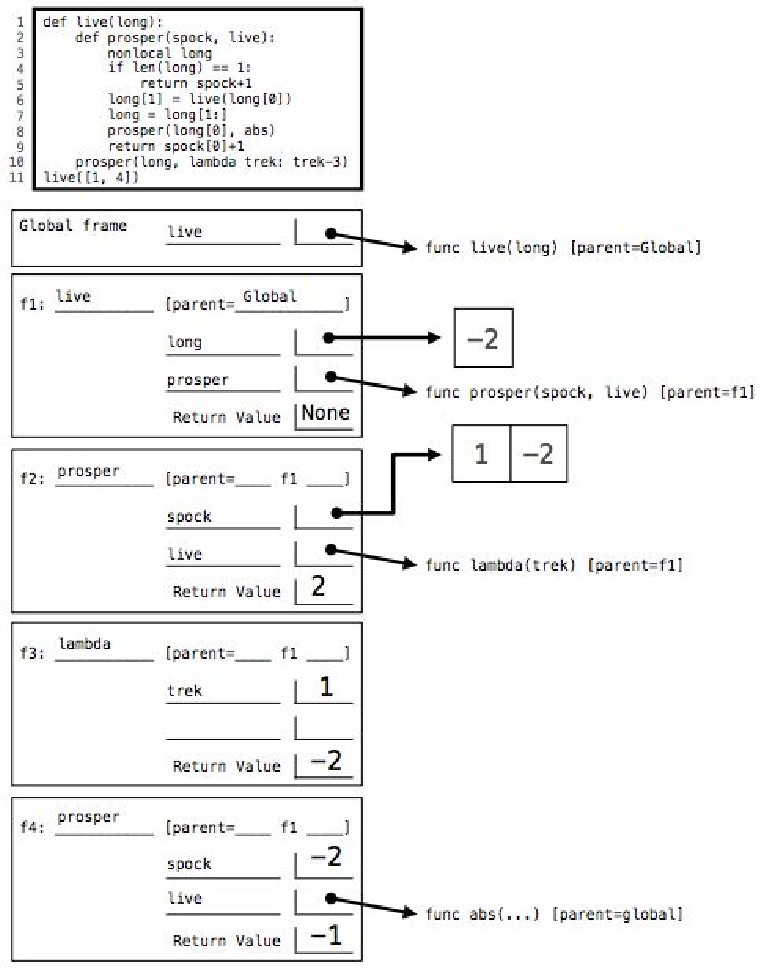
2. (12 points) Vulcans

(a) **(8 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.

Remember: Do not add a new frame when calling a built-in function (such as `abs`). The built-in `abs` function is always written as `func abs(...) [parent=Global]`.



```
def live(long):
    def prosper(spock, live):
        nonlocal long
        if len(long) == 1:
            return spock+1
        long[1] = live(long[0])
        long = long[1:]
        prosper(long[0], abs)
        return spock[0]+1
    prosper(long, lambda trek: trek-3)
live([1, 4])
```

Global frame
live
func live(long) [parent=Global]
f1: live [parent= Global]
long
-2
prosper
func prosper(spock, live) [parent=f1]
Return Value None
f2: prosper [parent= f1]
1 -2
spock
live
func lambda(trek) [parent=f1]
Return Value 2
f3: lambda [parent= f1]
trek 1
Return Value -2
f4: prosper [parent= f1]
spock -2
live
func abs(...) [parent=global]
Return Value -1

Class Hierarchy - What Would Python Print? (Fall 2014 Midterm 2 Q1)
Video Walkthrough: https://www.educreations.com/lesson/view/midterm-2-q1/26078314/

1. **(12 points) Class Hierarchy**

For each row below, write the output displayed by the interactive Python interpreter when the expression is evaluated. Expressions are evaluated in order, and **expressions may affect later expressions.**

Whenever the interpreter would report an error, write ERROR. You *should* include any lines displayed before an error. *Reminder*: The interactive interpreter displays the `repr` string of the value of a successfully evaluated expression, unless it is `None`. Assume that you have started Python 3 and executed the following:

```
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return Bourgeoisie.greeting
class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'My job is to gather wealth'
class Proletariat(Worker):
    greeting = 'Comrade'
    def work(self, other):
        other.greeting = self.greeting + ' ' + other.greeting
        other.work() # for revolution
        return other
jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```

Expression	Interactive Output
5*5	25
1/0	ERROR
Worker().work()	'Sir, I work'
jack	Peon
jack.work()	'Maam, I work'

Expression	Interactive Output
john.work()[10:]	Peon, I work 'to gather wealth'
Proletariat().work(john)	Comrade Peon, I work Peon
john.elf.work(john)	'Comrade Peon, I work'

Code Writing - What color is it? (Spring 2015 Midterm 2 Q4a)
Video Walkthrough: http://youripark.github.io/midterm2sp15.html (Scroll to Question 4)

4. (12 points) What color is it?

(a) (6 pt) Implement the `look` method of the `Dress` class. The `look` method returns a `Dress` instance's current color when the number of times that the instance's `look` method has ever been invoked evenly divides the total number times that the `look` method of any `Dress` instance has ever been invoked. Otherwise, the instance's `color` changes to the most recently returned color from any call to `look`, and `None` is returned.

```
class Dress:
    """What color is the dress?

    >>> blue = Dress('blue')
    >>> blue.look()
    'blue'
    >>> gold = Dress('gold')
    >>> gold.look()
    'gold'
    >>> blue.look()  # 2 does not evenly divide 3; changes to gold
    >>> Dress('black').look()
    'black'
    >>> gold.look()  # 2 does not evenly divide 5; changes to black
    >>> gold.look()  # 3 evenly divides 6
    'black'
    >>> Dress('white').look()
    'white'
    >>> gold.look()  # 4 evenly divides 8
    'black'
    >>> blue.look()  # 3 evenly divides 9
    'gold'
    """
    seen = 0
    color = None

    def __init__(self, color):
        self.color = color
        self.seen = 0

    def look(self):

        Dress.seen += 1

        self.seen += 1

        if Dress.seen % self.seen == 0:

            Dress.color = self.color

            return self.color

        else:

            self.color = Dress.color
```

Code Writing - Cucumber (Fall 2015 Final Q4)
Video Walkthrough: https://www.youtube.com/watch?v=N3WWzfNCyf4&t=2664s (Starts 44:24)

4. (8 points) Cucumber

Cucumber is a card game. Cards are positive integers (no suits). Players are numbered from 0 up to `players` (0, 1, 2, 3 in a 4-player game). In each `Round`, the players each `play` one card, starting with the `starter` and in ascending order (player 0 follows player 3 in a 4-player game). If the `card` played is as high or higher than the `highest` card played so far, that player takes `control`. The winner is the last player who took control after every player has played once. Implement `Round` so that `play_round` behaves as described in the doctests below. Part of your score on this question will be assigned based on *composition* (don't repeat yourself).

```
def play_round(starter, cards):
    """Play a round and return all winners so far. Cards is a list of pairs.
    Each (who, card) pair in cards indicates who plays and what card they play.

    >>> play_round(3, [(3, 4), (0, 8), (1, 8), (2, 5)])
    [1]
    >>> play_round(1, [(3, 5), (1, 4), (2, 5), (0, 8), (3, 7), (0, 6), (1, 7)])
    It's not your turn, player 3
    It's not your turn, player 0
    The round is over, player 1
    [1, 3]
    >>> play_round(3, [(3, 7), (2, 5), (0, 9)]) # Round is never completed
    It's not your turn, player 2
    [1, 3]
    """
    r = Round(starter)
    for who, card in cards:
        try:
            r.play(who, card)
        except AssertionError as e:
            print(e)
    return Round.winners

class Round:
    players, winners = 4, []
    def __init__(self, starter):
        self.starter, self.player, self.highest = starter, starter, -1

    def play(self, who, card):

        assert not self.complete(), 'The round is over, player '+str(who)

        assert who == self.player, "It's not your turn, player "+str(who)

        self.player = (who + 1) % self.players

        if card >= self.highest:

            self.highest, self.control = card, who

        if self.complete():

            self.winners.append(self.control)

    def complete(self):

        return self.player == self.starter and self.highest > -1
```