Name: _______ **SOLUTION** ______ CM: ____________ Section: _____ Grade: _______of 10

```
class SecretAgent(object):
    """ A secret agent w/ multiple identities. """
    def __init__(self, names):
        self.names = names
        self.current_name_index = 0

    def show_all_names(self):
        """ Prints all of this SecretAgent's names. """
        for k in range(len(self.names)):
            print(self.names[k])

    def change_to_next_name(self):
        # --- Location 1 ---
        self.current_name_index = (self.current_name_index + 1) % len(self.names)

    def give_current_name(self):
        """ Returns the name that this Secret Agent is currently using. """
        return self.names[self.current_index]


agent1 = SecretAgent(["Mary", "Jane", "Lola", "Terri"])
agent2 = SecretAgent(["Bob", "Niko", "Rick", "Zane", "Alec"])
agent1.change_to_next_name()
print(agent1.give_current_name())
print(agent2.give_current_name())
```

1. Fill in the code for `show_all_names`.

2. Fill in the code for `give_current_name`. It must use the `current_name_index`.

3. Which object is ***`self`*** when Location 1 is executed in the above code? ____agent1

4. (a) If you call `change_to_next_name()` repeatedly, it works for a while, but what error will you eventually get and why?

   Index out of bounds

   (b) Instead, once you get to the last name, it should cycle back to the first name. Add code to that method above to make that happen and so fix the bug.

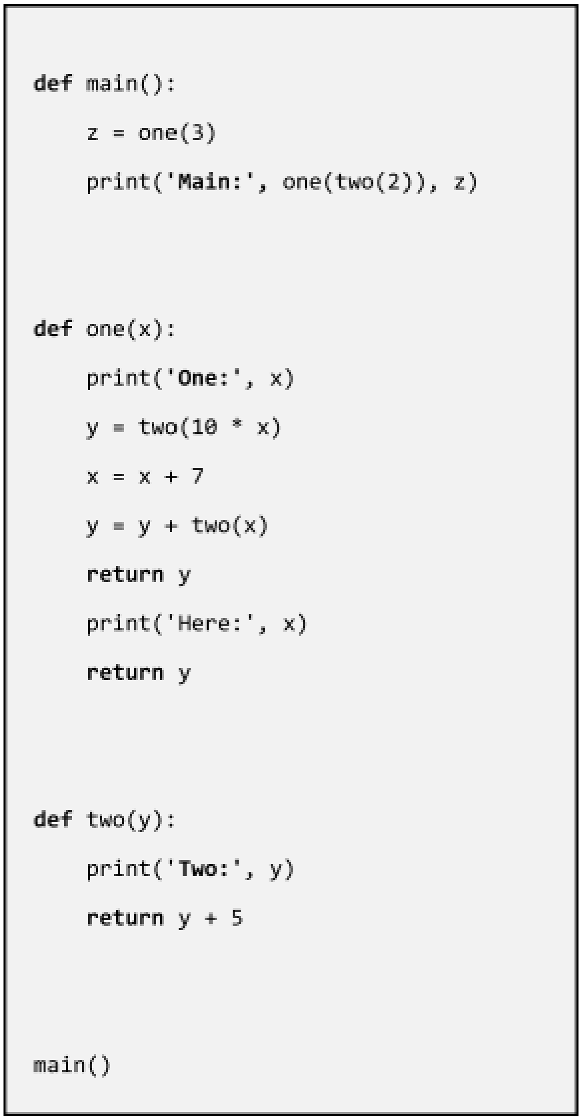
5. To the right of the code, give the output of running the completed code above.

   Sorry, no answer for this. Work out your own.

6. Consider the code snippet below. It is a contrived example with poor style, but it will run without errors. What does it print when it runs?

   Write your answer in the box to the right of the code.

   Showing your work by marking up the code to show its execution is the best way to allow for partial credit.



```
def main():
    z = one(3)
    print('Main:', one(two(2)), z)


def one(x):
    print('One:', x)
    y = two(10 * x)
    x = x + 7
    y = y + two(x)
    return y
    print('Here:', x)
    return y


def two(y):
    print('Two:', y)
    return y + 5


main()
```

Output: I have put extra spaces in the answer to make it easier to read.

```
One:    3
Two:   30
Two:   10
Two:    2
One:    7
Two:   70
Two:   14
Main:  94   50
```

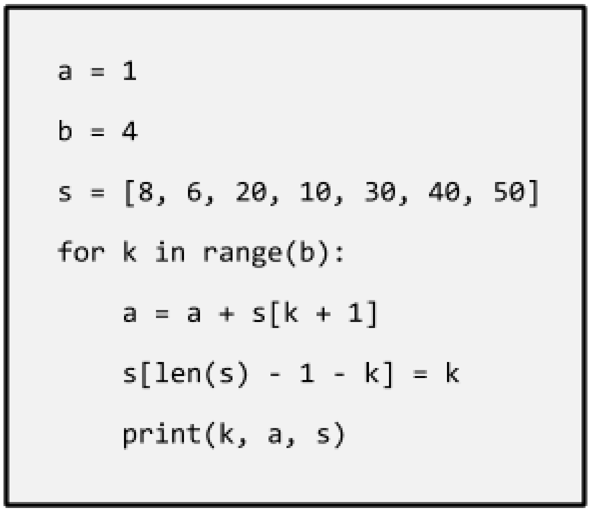
Subtract 1 point for each number that is wrong or missing. Note that the *Main* line counts 2 points.

Subtract 2 points if correct lines but in wrong order.

Ignore any small errors like omitting colons. Ignore any "slip of the pen" errors (where the student has the right idea).

7. Consider the code snippet to the right. It is a contrived example with poor style, but it will run without errors.

What does it print when it runs? Write your answer in the box below. We suggest that you use the space on a separate page to keep track of the values of **k**, **a**, **b** and **s** as you work.



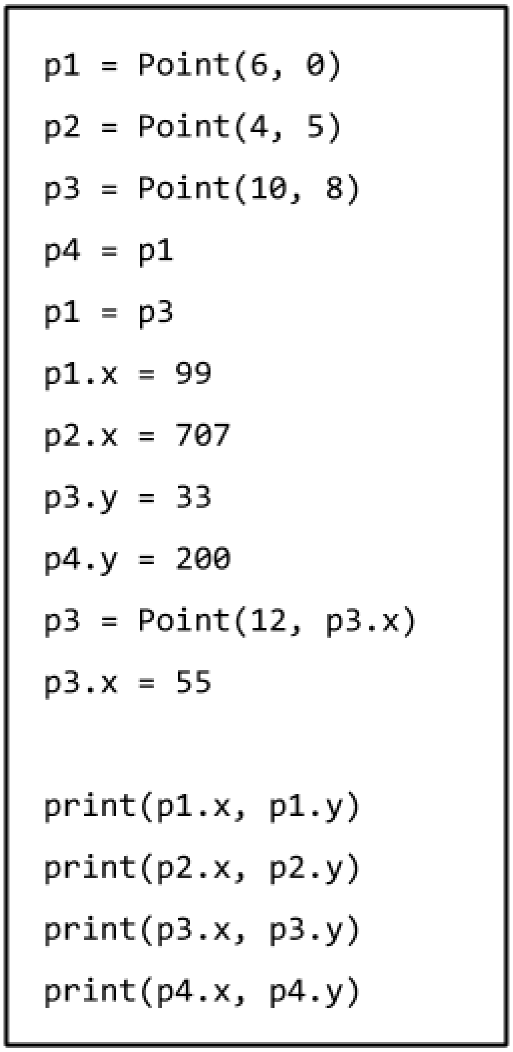
```
a = 1
b = 4
s = [8, 6, 20, 10, 30, 40, 50]
for k in range(b):
    a = a + s[k + 1]
    s[len(s) - 1 - k] = k
    print(k, a, s)
```

Output: I have put extra spaces in the answer to make it easier to read.

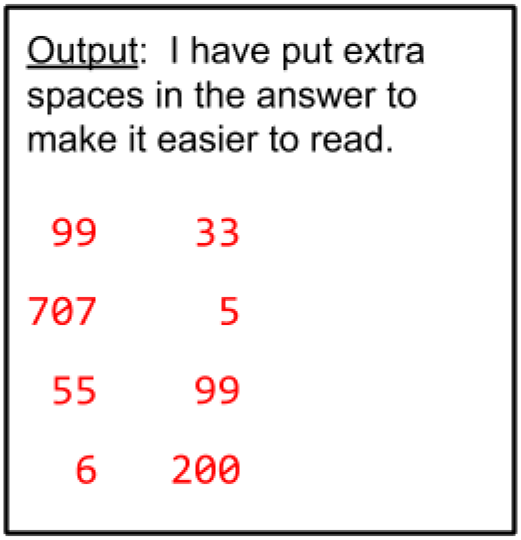
```
0     7     [8, 6, 20, 10, 30, 40, 0]

1    27     [8, 6, 20, 10, 30, 1, 0]

2    37     [8, 6, 20, 10, 2, 1, 0]

3    39     [8, 6, 20, 3, 2, 1, 0]
```

8. Consider the code snippet to the right. It is a contrived example with poor style, but it will run without errors.

   ***Draw** a box-and-pointer diagram* **for the execution of the code. Then show, in the box below, what the code *prints* when it runs.**



```
p1 = Point(6, 0)
p2 = Point(4, 5)
p3 = Point(10, 8)
p4 = p1
p1 = p3
p1.x = 99
p2.x = 707
p3.y = 33
p4.y = 200
p3 = Point(12, p3.x)
p3.x = 55


print(p1.x, p1.y)
print(p2.x, p2.y)
print(p3.x, p3.y)
print(p4.x, p4.y)
```



Output: I have put extra spaces in the answer to make it easier to read.

```
 99    33
707     5
 55    99
  6   200
```

Draw your *box-and-pointer diagram* below here, or use a separate sheet (your choice):

Rubric: Throughout, a maximum of -5.

Subtract 1 point for each of the 8 numbers that is wrong.

Subtract 1 point if there is NO box-and-pointer diagram or if the box-and-pointer diagram is clearly NONSENSE. If it simply has errors, that is OK.

1. (15 points). Consider the code on the next page. It is a contrived example with poor style but will run without errors. In this problem, you will trace the execution of the code. As each location is encountered during the run: ***CIRCLE* each variable** that is ***defined*** at that location; ***WRITE*** the ***VALUE*** of each variable that you *circled* directly ***BELOW*** the circle.

Location 1 (1st time)	a 4	z 100	self.a	self.b	t1.a	t1.b	t2.a	t2.b	r.a	r.b	s.a	s.b
Location 2 (1st time)	a 99	z 100	self.a 40	self.b 100	t1.a	t1.b	t2.a	t2.b	r.a	r.b	s.a	s.b
Location 1 (2nd time)	a 22	z 505	self.a	self.b	t1.a	t1.b	t2.a	t2.b	r.a	r.b	s.a	s.b
Location 2 (2nd time)	a 99	z 505	self.a 220	self.b 505	t1.a	t1.b	t2.a	t2.b	r.a	r.b	s.a	s.b
Location 3	a	z	self.a 40	self.b 100					r.a 40	r.b 100	s.a 220	s.b 505
Location 4	a	z	self.a 13	self.b 70					r.a 13	r.b 70	s.a 220	s.b 505
Location 5	a	z 44	self.a 91	self.b 70					r.a 91	r.b 70	s.a 220	s.b 11
Location 6	a 3	z 100	self.a	self.b	t1.a 40	t1.b 100	t2.a	t2.b	r.a	r.b	s.a	s.b
Location 7	a 3	z 100	self.a	self.b	t1.a 40	t1.b 100	t2.a 220	t2.b 505	r.a	r.b	s.a	s.b
Location 8	a 3	z 88	self.a	self.b	t1.a 91	t1.b 70	t2.a 220	t2.b 11	r.a	r.b	s.a	s.b

Name: ______________________________________ **Section:** _______

ASK FOR HELP IF YOU DO NOT UNDERSTAND WHAT THIS PROBLEM ASKS YOU TO DO.

Use the space to the right or a separate sheet to show your work in any way that you find helpful. We suggest a Box and Pointer diagram.

```
class Thing(object):
    def __init__(self, a, z):
        #### ---  Location 1  ---
        self.a = a * 10
        self.b = z
        a = 99
        #### ---  Location 2  ---

    def go(self, r, s):
        #### ---  Location 3  ---
        self.a = 13
        self.b = 70
        #### ---  Location 4  ---
        r.a = 91
        s.b = 11
        z = 44
        #### ---  Location 5  ---
        return 88


def main():
    a = 3
    z = 100
    t1 = Thing(z, 4)
    #### ---  Location 6  ---
    t2 = Thing(505, 22)
    #### ---  Location 7  ---
    z = t1.go(t1, t2)
    #### ---  Location 8  ---



main()
```

```
First time for:
Location 1:                        1 point
Location 2, a and z:               1 point
Location 2, self.a and self.b:     1 point

Location 6:                        1 point

Second time for:
Location 1 (2nd time):             1 point
Location 2, a and z:               1 point
Location 2, self.a and self.b:     1 point

Location 7, t2.a and t2.b:         1 point
Rest of location 7:         Anything is OK

Location 3, self.a and self.b:     1 point
Location 3, r.a, r.b, s.a, s.b:    1 point

Location 4, self.a and self.b:     1 point
Rest of Location 4:         Anything is OK

Location 5, self.a and self.b: Any is OK
Location 5, r.a and r.b:           1 point
Location 5, s.a and s.b:           1 point

Location 8, a and z:               1 point
Location 8, t1.a and t1.b:         1 point
Location 8, t2.a and t2.b:         1 point

This is  16  points total,
so a score of 16 of 15 is possible.
```