Test 2 – Practice Problems for the Paper-and-Pencil portion

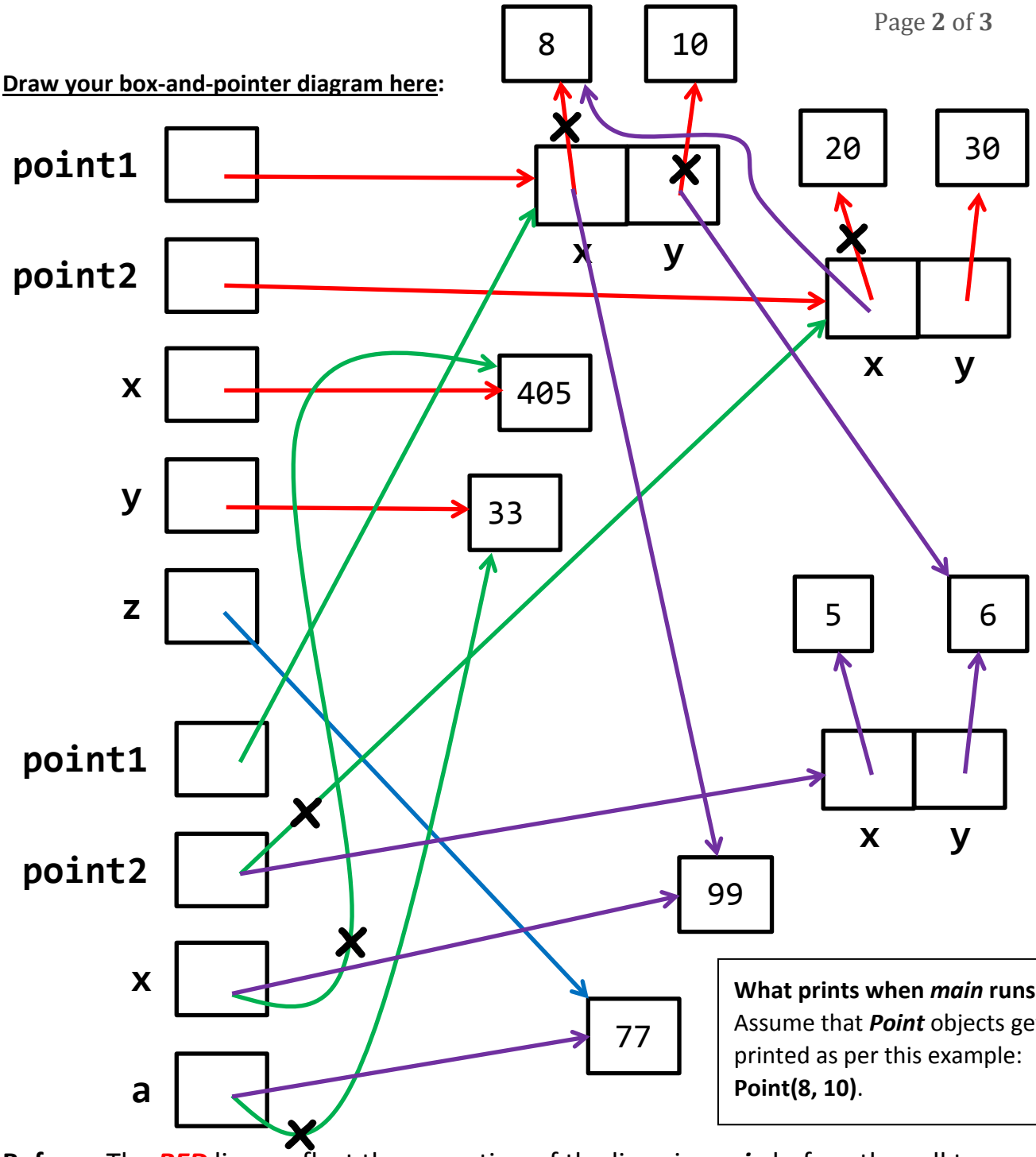
SOLUTION

1. In Session 8, you implemented a ***Point*** class. Recall that a ***Point*** object has instance variables ***x*** and ***y*** for its x and y coordinates.

   Consider the code in the box below. On the **next** page, draw the ***box-and-pointer diagram*** for what happens when ***main*** runs. Also on the next page, show what the code would ***print*** when ***main*** runs.

```
def main():
    point1 = Point(8, 10)
    point2 = Point(20, 30)
    x = 405
    y = 33

    print('Before:', point1, point2, x, y)

    z = change(point1, point2, x, y)

    print('After:', point1, point2, x, y, z)


def change(point1, point2, x, a):
    point2.x = point1.x
    point2 = Point(5, 6)
    point1.y = point2.y
    x = 99
    point1.x = x
    a = 77

    print('Within:', point1, point2, x, a)

    return a
```

Draw your box-and-pointer diagram here:



What prints when *main* runs? Assume that ***Point*** objects get printed as per this example: **Point(8, 10)**.

Before: The ***RED*** lines reflect the execution of the lines in ***main*** before the call to function ***change***. Therefore, what gets printed BEFORE the call to ***change*** is:
`Point(8, 10)  Point(20, 30)  405  33`

Within: The ***GREEN*** lines reflect the execution of the call to function ***change***. The ***PURPLE*** lines reflect the execution of the lines in ***change***. Therefore, what gets printed WITHIN the call to ***change*** (at the end of that function) is:
`Point(99, 6)  Point(5, 6)  99  77`

After: The ***BLUE*** line reflects the execution of the return from ***change*** and the assignment to ***z*** in function ***main***. Therefore, what gets printed AFTER the call to ***change*** is: `Point(99, 6)  Point(8, 30)  405  33  77`

2. In Session 8, you implemented a ***Point*** class. Recall that a ***Point*** object has instance variables ***x*** and ***y*** for its x and y coordinates.

Here, you will implement a portion of a class called ***Triangle***, described as follows:

- The ***Triangle*** constructor takes 3 arguments, each a ***Point*** object: one for each corner of the ***Triangle***.
- The ***Triangle*** class has a method called ***spin*(x, y)**. It changes the first of the Triangle's corners to **(x, y)**, where ***x*** and ***y*** are arguments of the ***spin*** method. It also swaps the other two of the Triangle's corners.

  For purposes of this problem, you can treat whichever corner you wish as the "first" of the Triangle's corners. Also, we don't care in this problem whether you construct new Points or make variables refer to existing ones.

Write the implementation of the class here:

```
class Triangle(object):

    def __init__(self, point1, point2, point3):
        self.corner1 = point1
        self.corner2 = point2
        self.corner3 = point3

    def spin(self, x, y):
        self.corner1 = Point(x, y)

        temp = self.corner2
        self.corner2 = self.corner3
        self.corner3 = temp
```

Note: It would be fine to replace
self.corner1 = Point(x, y)
by
self.corner1.x = x
self.corner1.y = y
since the problem statement is ambiguous as to whether ***spin*** is to make a ***new*** **Point** at **(x, y)** or modify the existing **Point's x** and **y**.

For each of the following, write one TEST of:

- The ***construction*** of Triangle objects.

```
p1 = Point(10, 20)
p2 = Point(30, 40)
p3 = Point(0, 50)
print(p1, p2, p3)
tri = Triangle(p1, p2, p3)
print(tri.corner1, tri.corner2, tri.corner3)
```

You MUST ***construct*** a Triangle object to test ***construction*** of Triangle objects. But there are other alternatives to PRINTING its corners as the test (e.g. you could draw them).

- The execution of the ***spin*** method.

The above plus:

```
tri.spin(300, 200)
print(tri.corner1, tri.corner2, tri.corner3)
```