Exam 3 – Paper and Pencil part (Winter, 2018-19)

Name: ____ Section: _______

Rules and Expectations

At the beginning of this exam, you will receive the **Expectations about Academic Integrity** for this exam -- it is the same as what you were given to read previously. Re-read that document as needed. ***Sign it and turn it in when you finish this exam (both parts).***

Two parts (this is Part 1, Paper-and-Pencil)

For this part, the ONLY external resource you may use is three (3) one-sided 8½ by 11-inch sheets of paper, with whatever you want on them, typed or handwritten or a combination of the two. (Or, you may use one two-sided sheet and another single-sided sheet.) You must have prepared the sheets *before* beginning this exam. You may also use a calculator if you like (but only for calculating), as well as additional blank paper. **You may NOT use your computer.**

Problem	Points Possible	Points Earned	Comments
1	**6**		
2	**6**		
3	**6**		
4	**6**		
5	**8**		
6	**8**		
7	**10**		
Total (of 50 on the exam)	**50**		

Communication: For both parts of the exam, ***you must not communicate with anyone*** except your instructors and her assistants, if any. In particular:

- You must not talk with anyone else or exchange information with them during this exam.
- After this exam, you must not talk about the exam with anyone who has not yet taken it.

You must NOT use email, *chat* or the like during this exam. **Close all such applications before you start the exam.**

1. 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 below the code.

```
def mysteryFunction(input):
    a = []
    for k in range(len(input)):
        if input[k] > 0 and input[k] < 10:
            a = a + [input[k]]
    print(a)


mysteryFunction([1, -4, 5, 9, -6, 19, 10, 8, 11, -9, 6])
```

2. 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 (in any way you wish) is the best way to allow for partial credit.

```
def main():
    value = 0
    total = 0
    while True:
        value = value + 2
        if value > 8:
            break
        total = (2 * total) + value + 1
        print(value, total)
    print('End:', value, total)


main()
```

3. 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 in the space below (in any way you wish) is the best way to allow for partial credit.

```
def main():
    for r in range(5, 2, -1):
        print('RED:', r)
        for s in range(1, r + 1):
            print(r, s)
        print(GREEN:', r)
    print('BLUE')



main()
```

4. 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 below the code.

```
def mystery(input):
    a = []
    for k in range(len(input)):
        sublist = input[k]
        for j in range(1, len(sublist)):
             if sublist[j] > sublist[j-1]:
                 a = a + [sublist[j]]
    print(a)

mystery([[1, 2, 3], [4, 2, 10, 5, 8], [7, 3, 4, 9, 1]])
```

5. In the box on the next page, implement the **`to_the_right`** function whose specification is:

What comes in:
- A sequence of subsequences of rg.Points, and
- An integer ***z***

What goes out: ***Returns*** a list that contains the **y-values** of the points that meet the following criteria: the **x-value** of that point is greater than the argument **z.**

Side effects: ***Prints*** all of the points in the subsequences whose x-value is greater than the argument **z.**

Example: If the given the sequence of subsequences of rg.Points is as shown below with **`z` as `3`**:

```
seq = [(rg.Point(3, 1), rg.Point(5, 4), rg.Point(15, 99)),
       [rg.Point(0, 0)],
       [],
       (rg.Point(6, -20), rg.Point(-20, -10), rg.Point(3.1, 0))
      ]
```

then the function ***prints*** the points:

```
Point(5, 4)
Point(15, 99)
Point(6, -20)
Point(3.1, 0)
```

and ***returns*** the list

```
[4, 99, -20, 0].
```

```
def to_the_right(seq, z):
```

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

   Draw a ***box-and-pointer diagram*** below that shows the execution of the code. In the Output box on the next page, show ***what gets printed***.

```
def main():
    a = 5
    s = [44, 10, 3]
    t = [61, s[0]]
    foo(a, s)
    print(a)
    print(s)
    print(t)

def foo(a, s):
    a = 92
    s[0] = 355
    s = [7, 8, 9]
```

Output:

7. Consider the code snippet below. It is a contrived example with poor style but will run without errors. In the space below, draw a box-and-pointer diagram that shows the execution of the code. AFTER drawing your box-and-pointer diagram, in the box to the right show what the output would be (from the PRINT statements).

```
a = [1, 2, 3]
b = a
b[2] = 9
p = Point(50, 38)
p2 = p
p2.x = 5
p.y = 44
p = Point(33, 402)
p3 = p
p4 = p3
p.x = 505
p3.x = 25
c = 15
d = c
c = 371

print(a[2])
print(p2.x, p2.y)
print(p4.x, p4.y)
print(d)
```

Output: