Exam 1 – Paper and Pencil part

Name: _____ Section: _____

Honesty pledge

At the beginning of the exam, you will receive an ***Honesty Pledge*** that is exactly the same as the one which you should have read before this exam. Re-read the Honesty Pledge at the beginning of this exam. ***Sign and turn in that pledge 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 a single 8½ by 11-inch sheet of paper, with whatever you want on it, typed or handwritten or a combination of the two. You may use only ONE side of the sheet. You must have prepared the sheet *before* beginning this exam.

Problem	Points Possible	Points Earned
1	8	
2	8	
3	8	
4	10	
5	10	
6	6	
Total (of 100 on the exam)	50	

Communication

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

- You must not talk with anyone else or exchange information with them during this exam.
- ***You must NOT use email****, chat* or the like during this exam. **Close all such applications before you start the exam.**

1. Objects and methods

a. Assume that you have a variable **`my_circle`** that refers to an **`rg.Circle`** object. Write a statement (or a sequence of statements if you prefer) that would cause the ***center*** of **`my_circle`** to become a new **`rg.Point`** at **`(100, 150)`**.

b. Assume that you have variables **`jack`** and **`jill`** that refer to **`Person`** objects. Assume further that **`Person`** objects have a **`say_phrase_to`** method that takes two arguments:

- a phrase ***S, that is a string,*** to say to another **`Person`**.
- a **`Person`** ***P*** to whom the first **`Person`** (the one calling the **`say_phrase_to`** method) is to say the phrase.

Write a statement that would make **`jill`** say **`"roll down the hill"`** to **`jack`**.

2. Assume that you have variables **`r`** and **`s`** that are positive integers with **`s > r`**. In the space below, write code that would print every integer from **`(s - r)`** to **`s`** (but NOT including **`s`** itself). For example, if **`r = 8`** and **`s = 20`**, your code would produce the output shown to the right.

```
12
13
14
15
16
17
18
19
```

Write code for the generic case for **`r`** and **`s`**. That is, use the variables **`r`** and **`s`** in your code. You should use a **`range`** statement in your solution, but you may NOT use the multiple-argument form of **`range.`** It must have only a ***single*** argument.

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. Use the spare space to keep track of variables as you work.

```
t1 = 4
t2 = 0
for k in range(5):
    t1 = t1 + k
    t2 = t2 + t1
    print(k, t1, t2)

print('ok')
print(t1, t2)
```

Output:

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 to the right of the code. Use the page to the right to show your work (if needed).

```
def main():
    z = red(5, 2)
    print('Main:', z)


def red(x, y):
    print('Red:', x, y)
    print(blue(x) + blue(y))
    return blue(x + y)
    print('Green')


def blue(x):
    print('Blue:', x)
    x = x + 4
    print('After:', x)
    return x * 10


main()
```

Output:

Space to show your work for problem 4.

5. Consider the code below. 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:

1. ***CIRCLE* each variable** that is ***defined*** at that location.
2. ***WRITE* the *VALUE*** of each variable that you ***circled*** directly ***BELOW*** the circle.

```
#### Location 1

def main():
    #### Location 2
    a = 3
    b = 5
    c = 44
    d = 50
    y = 123
    #### Location 3

    y = red(a, c, d)
    #### Location 4


def red(a, d, c):
    #### Location 5
    a = 88
    b = 99
    r = d + 1
    #### Location 6
    return a


main()

#### Location 7
```

Location 1	a	b	c	d	r	y
Location 2	a	b	c	d	r	y
Location 3	a	b	c	d	r	y
Location 4	a	b	c	d	r	y
Location 5	a	b	c	d	r	y
Location 6	a	b	c	d	r	y
Location 7	a	b	c	d	r	y

Note that you fill out the table **in the order that the locations are encountered, *NOT from top to bottom.***

Ask for help if you do not understand how this problem works.

Space to show your work for problem 5 (if you want).

6. Consider a function whose name is ***return_sum*** that takes two arguments, a positive integer **m** and a number **r**. The function computes and returns the sum:

```
[1 * r] + [2 * (r ** 2)] + [3 * (r ** 3)] + ... + [m * (r ** m)]
```

For example, **return_sum(6, 2)** returns

```
(1 * 2) + (2 * 4) + (3 * 8) + (4 * 16) + (5 * 32) + (6 * 64)
```

(which is **642**) while **return_sum(3, 0.5)** returns

```
0.5 + (2 * 0.25) + (3 * 0.125)
```

(which is **1.375**)

Write (in the box below) a complete implementation, including the header (***def***) line, of the above ***return_sum*** function.