Part of REQUIRED Admission Ticket for Exam 1

Exam 1 – Paper and Pencil part (Spring, 2020-21)

Name: ______________________________ **Section:** _______

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. You may also use a calculator if you like (but only for calculating).

Problem	Points Possible	Points Earned	Comments
1	10		
2	4		
3	4		
4	6		
5	6		
6	10		
7	10		
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 delegates, 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.

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 to the right of the code. Show your work by making notations in the code or by using the empty space below or on another sheet of paper, as desired.

```
def main():
    g = snowflake(4, 2)
    print("Main:", g)


def snowflake(x, y):
    print("Snowflake 1:", x, y)
    print(fang(x) + fang(y))
    print("Snowflake 2:", x, y)
    return fang(x + y)
    print("Done!")


def fang(x):
    print("Fang:", x)
    x = x + 1
    return x * 10


print("Winter:", fang(53))
main()
```

Output:

2. Assume that you have a name (i.e., variable) **`my_circle`** that refers to an **`rg.Circle`** object that has already been made in the code and that
   `import rosegraphics as rg`
   already appears in the code.

   Write statements that would:

   - construct a ***new*** **`rg.Point`** named **`p`** located at **`(400, 200)`**;
   - and cause the ***center*** of **`my_circle`** to become that newly constructed **`rg.Point`**.

3. Assume that you have names (i.e., variables) **`snowflake`** and **`fang`** that refer to **`Gerbil`** objects that have already been made in the code. Assume further that **`Gerbil`** objects have a **`squeak_at`** method that takes three arguments:

   - a **duration,** that is a number for how long to squeak in seconds
   - a **volume**, that is a number for how loud to squeak (1 to 10, 10 is loudest)
   - and a **`gerbil`** argument, of type **Gerbil**, to whom the first **`Gerbil`** (the one calling the **`squeak_at`** method) is to squeak at.

   (a) **Write a statement** (below) that would make **`snowflake`** squeak at **`fang`** for a minute at the maximum (loudest) volume.

   (b) **Write a statement** (below) that would make **`fang`** squeak at **`snowflake`** for 20 seconds at the minimum (least loud) volume.

4. Assume that you have names (i.e, variables) **a** and **b** that are positive integers where **b > a**. In the box below, ***fill in the blanks*** with code that would print every integer from **b - a** to **b + a** including the **b + a** value. For example, if **a = 4** and **b = 11**, your code would print every integer from **7** to **15**, thus producing the output shown to the right.

   Your code must be for the ***generic case*** for **a** and **b**. That is, use the names (i.e., variables) **a** and **b** in filling in the blanks, as appropriate. As in all problems throughout Exam 1, you must use the ***single-argument*** form of **range**, as in **range(blah)**. You may NOT use the multiple-argument form of **range**, as in **range(r, s)**.

```
7
8
9
10
11
12
13
14
15
```

```
for k in range(________________________________________):

    print(________________________________________)
```

5. 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 and to the right.

   Suggestion: Use a table on scratch paper to keep track of the values of **k**, **a** and **b** as you trace through the code.

```
a = 2
b = 8
for k in range(5):
    a = a + k
    b = a + b
    print(k, a, b)

print("ok", a, b)
```

Output:

6. 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 name (i.e., variable)** that is ***defined*** at that Location.
    2. ***WRITE*** the ***VALUE*** of each name (i.e., variable) that you ***circled*** directly **BELOW** the circle.

    Note that you fill in the table in the order that Locations are encountered, NOT from top to bottom. Ask for help if you do not understand these instructions.

```
def main():
    #### Location 1
    a = 2
    b = 4
    c = 10
    d = 505
    y = 100
    #### Location 2

    y = cat(a, c, d)
    #### Location 3


def cat(a, d, c):
    #### Location 4
    b = 99
    r = a + 1
    y = 20
    c = 42
    #### Location 5
    return a + 5

main()

#### Location 6
```

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

7. Consider a function whose name is ***return_sum*** that takes two arguments, a positive integer `m` first, then a number `r`. The function computes and returns the first ***m*** terms of the series:

```
[3 * (r ** 1)]  +  [4 * (r ** 2)]  +  [5 * (r ** 3)]  +  [6 * (r ** 4)] + ...
```

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

```
(3 * 2)  +  (4 * 4)  +  (5 * 8)  +  (6 * 16)  +  (7 * 32)  +  (8 * 64)
```

(which is **894**) while **`return_sum(3, 0.5)`** returns

```
(3 * 0.5) + (4 * 0.25) + (5 * 0.125)
```

(which is **3.125**)

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