Name: ______________________________

Please answer the following questions within the space provided on the following pages. Should you need more space, you can use scratch paper, but clearly label on the scratch paper what problem it corresponds to. While you are not required to document your code here, comments may help me to understand what you were trying to do and thus increase the likelihood of partial credit should something go wrong. If you get entirely stuck somewhere, explain in words as much as possible what you would try.

Each question clearly shows the number of points available and should serve as a rough metric to how much time you should expect to spend on each problem. You can assume that you can import any of the common libraries we have used throughout the semester thus far.

The exam is partially open, and thus you are free to utilize printed portions of:

- The text
- Your notes
- Slides
- Any past work you have done as part of sections, problem sets, or projects

Computers and internet capable devices are prohibited. *Your work must be your own on this exam, and under no conditions should you discuss the exam or ask questions to anyone but myself.* Failure to abide by these rules will be considered a breach of Willamette's Honor Code and will result in penalties as set forth by Willamette's academic honesty policy.

**Please sign and date the below lines to indicate that you have read and understand these instructions and agree to abide by them.** *Failure to abide by the rules will result in a 0 on the test.* Good luck!!

______________________________ Signature

______________________________ Date

| Question: | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|
| Points: | 8 | 10 | 15 | 20 | 53 |
| Score: | | | | | |

(8) 1. **Evaluating Python Expressions:** For each of the below expressions, write out **both** the value of the expression and what type of object that value is (`int`, `float`, `bool`, etc).

(a) `10 ** ((9 + 8) % 7) + 6 * 5 - 4 // 3 + 2 * 1`

**Solution:** `1031`, an `int`

(b) `4 % 7 == 0 or 10 // 4 == 2 and "A" == "a"`

**Solution:** `False`, a `bool`

(10) 2. **Program Tracing:** What output would the following program produce?

```
def mystery(w):

    def enigma(s, k):
        return s[4 - k % 3]

    s = ""
    for i in range(4):
        if i % 3 == 0:
            s += w[len(w) - 1]
        else:
            s += w[0]
        s += enigma(w, i)
    return s

if __name__ == '__main__':
    print(mystery("abcdefgh"))
```

Show your work and explain your reasoning for full credit.

**Solution:** The final output value would be `"headache"`. Remember that you need to compute the remainder in the `enigma` function before the subtraction!

(15) 3. Write a python function called `count_signs` that reads a sequence of integers entered in by a user, stopping when the user enters a blank line. After the blank line appears, your program should report the number of positive, negative, and zero values that were entered, as illustrated in the below sample run:

```
>>>count_signs()
Enter a blank line to stop.
 ? 4
 ? -9
 ? 5
 ? 0
 ? -6
 ? 9
 ? -7
 ? -8
 ? -2
 ? -4
 ? 4
 ?
 positive: 4, negative: 6, zero: 1
```

**Solution:** One possible solution:

```python
def count_signs():
    pos = 0
    neg = 0
    zero = 0
    finished = False
    while not finished:
        value = input(' ? ')
        if value == "":
            finished = True
        else:
            num = int(value)
            if num > 0:
                pos += 1
            elif num < 0:
                neg += 1
            else:
                zero += 1
    print(
        f"positive: {pos}, negative: {neg}, zero: {zero}"
    )
```

(20) 4. Write a graphics program that accomplishes the following:

1. Creates a filled red square that measures 50 pixels on each side
2. Adds said square to be centered in an 800 by 600 pixel window
3. Immediately starts a timer that moves the square 2 pixels every 20 milliseconds in a constant random direction. The random direction can be specified as an angle between 0 and 360, and would be suitable for use with the `move_polar` method.
4. Every time you click the mouse *inside* the square, the direction of the square changes to a new random angle. The speed of the square does not change. Clicking outside of the square has no effect.

If you were to actually write such a program, you would likely supply some means of bringing the program to a stop, especially if the square moves off of the screen, for instance. For the exam though, just have the program run continuously, without worrying about whether the square moves off the screen or how or when to stop the timer.

**Solution:**

```python
from pgl import GWindow, GRect
from random import uniform

WIDTH = 800
HEIGHT = 600
SQ_SIZE = 50

def step():
    sq.move_polar(2, gw.direction)

def click_action(e):
    mx = e.get_x()
    my = e.get_y()
    if sq.contains(mx,my):
        gw.direction = uniform(0,360)

gw = GWindow(WIDTH, HEIGHT)
gw.direction = uniform(0,360)
sq = GRect( WIDTH / 2 - SQ_SIZE / 2,
            HEIGHT / 2 - SQ_SIZE / 2,
            SQ_SIZE,
            SQ_SIZE)
sq.set_filled(True)
sq.set_color('red')
```

```
gw.add(sq)
gw.add_event_listener("click", click_action)
gw.set_interval(step, 20)
```