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) `5 * 5 + 5 // 5 + 5 * 5 * 5`

Solution: `151`, an `int`

(b) `"abcde"[2] + str(2 != 2.0)[len(str(6 / 3))]`

Solution: `cs`, a `str`

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

```
def mystery(x):

    def puzzle(x, y=5):
        return x * y

    def enigma(y):
        return y ** x

    return enigma(puzzle(2)) + enigma(puzzle(3,x))

if __name__ == '__main__':
    print(mystery(3))
```

Show your work and explain your reasoning for full credit.

Solution: The final output value would be 1729.

(15) 3. The German mathematician Gottfried Wilhelm von Leibniz discovered the rather remarkable fact that the mathematical constant π can be computed using the following mathematical relationship:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots$$

where the formula to the right of the equals sign represents an infinite series; each fraction represents a term in that series. If you start with 1, subtract one-third, add one-fifth, and so on for each of the odd integers, you get a number that gets closer and closer to the value of $\frac{\pi}{4}$ as you go along.

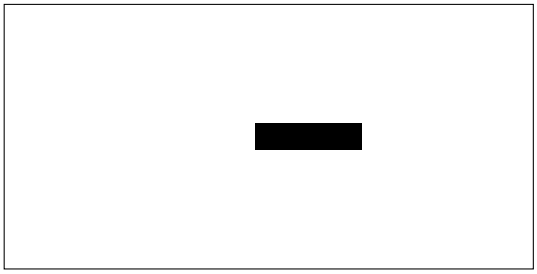
Write a program that calculates an approximation of π consisting of the first 10,000 terms in Leibniz's series. Your program should keep track of the running total and then go through a loop that alternately adds and subtracts the next term to the total. Once you have counted up to 10,000 terms, all you need to do is print out the total multiplied by 4.

Solution: One possible solution:

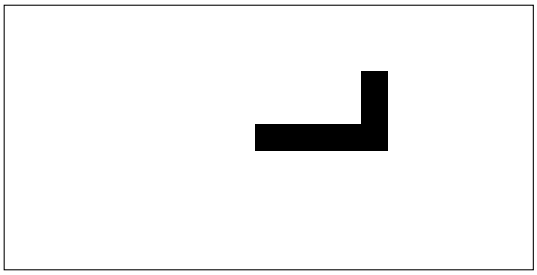
```
TERMS = 10000
total = 0
denom = 1
sign = 1
for i in range(TERMS):
    total += sign * 1 / denom
    sign *= -1
    denom += 2
print(f"The value of pi is approximately {4 * total}")
```

(20) 4. In this problem, your mission is to write a program that plays a very simplified version of the graphical game called "Snake", which was popular on the first generation of Nokia phones. When the program begins, there is nothing on the graphics window, but there is an invisible snake head at the center of the window. In each time step of the animation, the snake head moves 15 pixels in some direction, leaving behind a 15×15 filled square at the position it just left.

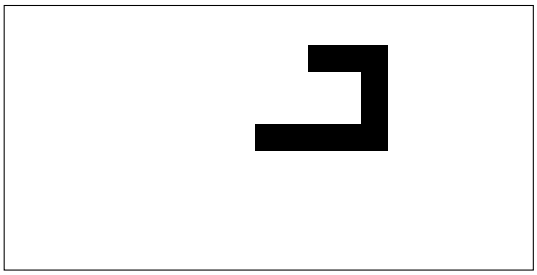
At the beginning of the game, the snake head is moving eastward, so after four time steps, it will have generated a trail of four squares (which run together on the screen), like this.



In this implementation of the game, you turn the snake by clicking the mouse. In this initial situation, with the snake moving horizontally, clicking the mouse above the current y position of the snake head sends it northward, and clicking below the current y position would send it south. In this particular example, let's assume that the player clicked above the snake head, so that its direction changes to the north. After three more time steps, the window would look something like this:



When the snake is moving vertically, clicking to the left of the snake head sends it westward, and clicking to the right sends it eastward. Clicking to the left would therefore send the snake off to the west, as follows:



In the actual game, the player loses if the path of the snake moves outside the window or closses its own path. For this problem, you can ignore the problem of stopping and just get the motion working.

While the program may seem complicated, there are not that many pieces which you need to get it working. Here are a few general hints or things to keep in mind as you get started:

- You need to keep track of three things: the x and y positions of the snake head, and some indication of what direction the snake is currently moving (North, South, East or West).
- The function that executes each time step must create a new black square whose center is at the current position, and *then* move the head on to the next square by updating the coordinates as appropriate to the current direction.
- The listener method that responds to mouse clicks has to look at the current position and direction and then use those together with the mouse click location to determine how to update the direction.
- Remember that you don't need to check whether the snake remains on the window or crosses its own path. You can implement those features on your own time after the exam if you really want!

Solution: One possible solution:

```
SIZE = 15
WIDTH = 20*SIZE
HEIGHT = 10*SIZE

def step():
    # Create rect at current position
    rect = GRect(gw.head_x - SIZE/2,
                 gw.head_y - SIZE/2,
                 SIZE, SIZE)
    rect.set_filled(True)
    gw.add(rect)
    # Advance head
    if gw.direction == 'E':
        gw.head_x += SIZE
    elif gw.direction == 'W':
        gw.head_x -= SIZE
    elif gw.direction == 'N':
        gw.head_y -= SIZE
    elif gw.direction == 'S':
        gw.head_y += SIZE
```

```
def click_action(e):
    mx = e.get_x()
    my = e.get_y()
    if gw.direction in 'NS':
        if mx < gw.head_x:
            gw.direction = 'W'
        else:
            gw.direction = 'E'
    else:
        if my < gw.head_y:
            gw.direction = 'N'
        else:
            gw.direction = 'S'

gw = GWindow(WIDTH, HEIGHT)
gw.head_x = WIDTH / 2
gw.head_y = HEIGHT / 2
gw.direction = 'E'
gw.set_interval(step, 200)
gw.add_event_listener("click", click_action)
```