WARNING: THE TOPICS AND EMPHASIS ARE LIKELY SLIGHTLY DIFFERENT FROM THIS YEAR – THIS IS JUST AN EXAMPLE! ALSO, THIS CLASS DIDN'T HAVE A CHEAT-SHEET, SO YOUR QUESTIONS, MIGHT BE SLIGHTLY DIFFERENT IN TONE / DIFFICULTY.

Name: _____

There are 106 points possible. NO CALCULATORS!!

1. (**16 points**) What is the DEFINITION of each of these (I'm *not* looking for a number answer):
   a. π

   b. Sin (Θ)

   c. Cos^{-1} (h)

   d. 1 radian

2. (**4 points**) How would you convert 137 degrees to the equivalent radian angle? I'm not asking for a number answer (since you don't have a calculator – instead I'm looking for an equation / expression that we *could* enter in python or a calculator).

3. (**8 points**) Suppose you are currently storing these python variables:
   a. **x**: the character's current x position
   b. **y**: the character's current y position
   c. **degrees**: the orientation of the character (in degrees)
   d. **speed**: The speed at which the character moves (in pixels / s)
   e. **dt**: the time (in seconds) since the last update.

   Show a python code snippet to show how to update the character's position if they were to move forward at the given rate after dt seconds have passed.

4. (**8 points**) Suppose you are currently storing these python variables:
    a. **x**: the character's current x position
    b. **y**: the character's current y position
    c. **tx**: The x position of a target
    d. **ty**: The y position of a target

    Show a python code snippet to create two additional variables:
    dist: the distance between the character and the target
    degrees: The angle (in degrees) towards the target (from the character's point-of-view)

5. (**20 points**) Show me what must be in the foo module for this main program to produce the output shown in comments:

```
import foo

f = foo.Blah(4, 15)         # Yarg! 19
g = foo.Blah(3, 2)          # Yarg! 5
print(f)                    # 15-15-15-15
print(g)                    # 2-2-2
f.toot()                    # Toot!
print(f)                    # 15-15-15-15-15
f.toot()                    # Toot!
print(f)                    # 15-15-15-15-15-15
g.toot()                    # Toot!
print(g)                    # 2-2-2-2
```

6. (**10 points**) Write a function called **move** that takes a parameter **L**, which is a list of lists, where each sub-list is of this form [**x**, **y**, **velX**, **velY**, **rad**]. The function also takes a parameter **dt** (the change in time, in seconds) and **accX**, **accY**, which is the acceleration these objects are being subjected to. The function should apply acceleration on all the objects and change their position. Any objects that go completely off-screen (in an 800x600 window) should be removed from the list. The function should return a new list of any objects that were removed. **(Note: We didn't do acceleration in the fall2017 semester. Basically, you do velX += accX * dt every frame and also posX += velX * dt)**

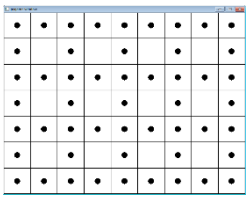
7. (**8 points**) Write a function named **ptInRect** which takes two arguments: **p** (a position tuple) and **r** (a pygame rectangle) and returns a Boolean indicating whether the point is within the rectangle.

8. (**8 points**) Suppose we have a sprite sheet being stored in a python (pygame.Surface) variable called **img**. Each frame of the animation is 100x100 pixels. Further suppose we have these variables:
    a. **frame**: an integer (0 – 7) that indicates what frame of animation we're on
    b. **direction**: a string (one of "w", "ne", "n", "nw", "w", "sw", "s", "se")
    c. **x**, **y**: a position
    d. **rows**: a dictionary which contains {"w":4, "se":7, and so on}

    Show a python blit command to center the *current* frame of animation at the given position on the screen. To get full points, use the dictionary.

9. (**13 points**) Write a python code snippet which will draw a grid using black lines. For even rows *or* even columns (or both), draw a small circle in the middle of the square. You have 3 variables to work with:
    a. **screen**: The surface to draw to (you can assume it is 800 x 600)
    b. **numCols**: The number of circles to draw horizontally
    c. **numRows**: The number of circles to draw vertically

    For example: if numCols were 9 and numRows were 7, the screen would look like:



10. (**5 points)** Convert this while loop to an equivalent for loop (using range). Use / create the same variables

```
i = 0
while i < 20:
      c = data[i]
      print("i = " + str(c))
      i += 1
```

11. (**6 points**) What is the output of this python snippet? If there errors, indicate them, but continue with the rest of the program.

```
G = (4, 7.2, "abc", [9, 3, 11], [14, 18, 19], "qrs")
print(len(G))
print(G[1])
print(G[-1])
print(G[2:4])
print(G[2][1])
```

Have a great break! I know this has been a tough semester – thank you for all your hard work!