Name: ______________________________ Recitation: ____ Andrew Id: ________________

15-112 Summer 2019 Quiz 3

Up to 50 minutes. No calculators, no notes, no books, no computers, no extra paper. Show your work!
Do not use sets, dictionaries, or recursion on this quiz.

1. (30 points) **Free Response:** The following code is part of an animation that draws a Dot (which should be drawn as a blue circle) with a radius of 25 pixels at a random location and random speed (chosen from 5 to 10 pixels/sec) every second. Every timer fired, the dots move in their direction, and when the edge of a dot hits the edge of the canvas, it should bounce off in the other direction. When a dot is clicked, it is deleted by removing it from the list. Write the Dot class (and nothing else!) so that the animation is complete. Read the code carefully so that you know how the Dot class should behave!

   Note: We have provided the helper function `distance(x1, y1, x2, y2)` if you wish to use it.

   Note: You should be writing at least 4 methods in the Dot class, although you can write more if you would like!

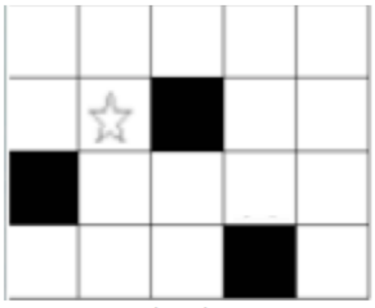
```python
import random

def distance(x1, y1, x2, y2):
    return ((x1-x2)**2 + (y1-y2)**2)**0.5

def init(data):
    data.dots = []
    data.timer = 0
    data.r = 25

def mousePressed(event, data):
    remainingDots = []
    for i in range(len(data.dots)):
        dot = data.dots[i]
        if not dot.clicked(event.x, event.y):
            remainingDots.append(dot)
    data.dots = remainingDots

def timerFired(data):
    data.timer += 1
    if data.timer % 10 == 0: # spawn a dot
        x = random.randint(data.r, data.width-data.r)
        y = random.randint(data.r, data.height-data.r)
        dx = random.choice([1, -1])
        dy = random.choice([1, -1])
        data.dots.append(Dot(x, y, dx, dy))

    for dot in data.dots: # move the dots
        # make sure the dots bounce off the edge of the canvas!
        dot.move(data.width, data.height)

def redrawAll(canvas, data):
    for dot in data.dots:
        dot.draw(canvas)
```

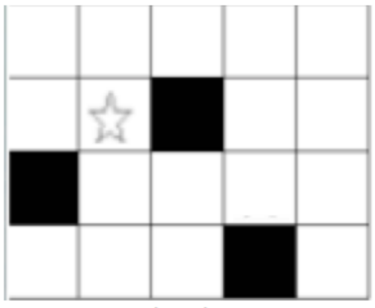
YOU MAY CONTINUE WRITING CODE ON THIS PAGE.

2. (30 points) **Free Response:** Background: an integer n is "powerful" if it is positive, and for every prime factor p of n, p^2 is also a factor of n. For example, 72 is a powerful number because the prime factors of 72 are 2 and 3, and 4 and 9 are also factors of 72.

   Without using any iteration, write the function `nthPowerfulNumber(n)` that returns the nth powerful number. `nthPowerfulNumber(0)` should return `4`. You may assume that `isPrime` is already written for you.

3. (10 points) **Short Answer:** Say we start with a 4x5 board and fill the black squares as shows below. Then we right click where the star is to start a floodFill from there. On the left, write the **depths** that result in each cell after floodFill completes. On the right, write the **ordinals** in each cell.


(a) depths


(b) ordinals

   1. The numbers of the left are depths, so some labels may occur more than once.
   2. The numbers on the right are ordinals, so labels must occur only once each.
   3. The first label is 0, not 1 (so a 0 should be where the star is).
   4. Our floodFill code recursively tries to go up, then down, then left, then right.

QUIZ IS ON BOTH SIDES OF THE PAGE

4. (15 points) **Code Tracing:** Indicate what the following program prints. Place your answer (and nothing else) in the box below the code.

```
class A(object):
    def __init__(self, x, y = 5):
        self.x = x
        self.y = y

    def __str__(self):
        return "A(%d, %d)" % (self.x, self.y)

    def bar(self):
        return self.x + self.y

class B(A):
    def __str__(self):
        return "B(%d)" % self.x

class C(A):
    def bar(self):
        return self.x*self.y


foo = A(1, 3)
for obj in [A(0, 2), B(2), C(3)]:
    print(type(obj) == type(foo), end = " ")
    print(isinstance(obj, type(foo)), end = " ")
    print(obj, obj.bar())
```

5. (15 points) **Reasoning Over Code**: Find an input value for `s` that makes `roc(a)` return `True`. Place your answer (and nothing else) in the box to the right of the code.

```
def f(L, d):
    if (len(L) == 0):
        return [ ]
    else:
        assert(d < min(L))
        return [L[0]%d] + f(L[1:], d)

def roc1(L):
    return (f(L[1:], L[0]) == [1, 4, 3])
```