Name:_____ Section:___ Andrew Id: _____ a

15-112 Fall 2019 Quiz 9a
*** Up to 20 minutes. No calculators, no notes, no books, no computers. * Show your work!**
*** No recursion**

1. Multiple Choice: Animations [20 pts]
Completely fill in the bubble for **each** correct answer (there may be more than one)!

a) Which of these describe how we use a spritestrip?

- ○ We use a loop to crop out frames of the spritestrip image and store the new images in a list
- ○ Spritestrip files come with a dictionary that allows us to specify which frame we want
- ○ Spritestrips are the same as animated gifs, and we just specify the refresh rate we want with a method
- ○ We create a method that visits a URL, and every time we get data from the URL we get a new frame

b) Which of these correctly describe sidescrolling?

- ○ Our model keeps track of most objects in world coordinates, and our view uses scrollX and/or scrollY to draw the world in canvas coordinates
- ○ Our animation framework has a special sidescrolling subclass that allows us to have a larger world than our canvas
- ○ Sidescrolling only works in ModalApps
- ○ We simulate moving right by drawing everything farther to the left

c) Which of these correctly describe the modes in ModalApp?

- ○ Modes describe each frame of a spritestrip
- ○ Modes in the sidescrolling ModalApp assignment should include classes like Player(mode), Monster(mode), and Item(mode)
- ○ Modes in the sidescrolling ModalApp assignment should include classes like splashScreenMode(mode), helpMode(mode), and gameMode(mode)
- ○ Each mode of a ModalApp has the components of a stand-alone animation, and could run by itself with a few small changes

d) If a character moves beyond the right margin in a typical horizontal sidescroller:

- ○ scrollX should increase
- ○ scrollX should decrease
- ○ Everything in world coordinates gets drawn farther to the left of the canvas
- ○ Everything in world coordinates gets drawn farther to the right of the canvas

2. Free Response: OOP with Marbles [80 pts]
Write the classes Marble, ConstantMarble, and DarkeningMarble so that the following test code passes. For full points, **you must use inheritance properly.**

```
# A Marble takes a string (not a list) of comma-separated color names
m1 = Marble('Pink,Cyan')
assert(m1.colorCount() == 2) # pink and cyan
assert(Marble.getMarbleCount() == 1) # we have created 1 marble so far

# When converted to a string, the Marble includes the color names,
# each separated by a comma and a space, and all lower-case, and listed
# in alphabetical order:
assert(str(m1) == '<Marble with colors: cyan, pink>')

m2 = Marble('Red,Orange,yellow,GREEN')
assert(str(m2) == '<Marble with colors: green, orange, red, yellow>')
assert(m2.colorCount() == 4)
assert(Marble.getMarbleCount() == 2) # we have created 2 marbles so far
# This also works in a list:
assert(str([m1]) == '[<Marble with colors: cyan, pink>]')

# Equality works as expected:
m3 = Marble('red,blue')
m4 = Marble('BLUE,RED')
m5 = Marble('red,green,blue')
assert((m3 == m4) and (m3 != m5) and (m3 != "Don't crash here!"))
assert(Marble.getMarbleCount() == 5) # we have created 5 marbles so far

# As do sets that contain marbles:
s = { m3 }
assert((m4 in s) and (m5 not in s))

# You can add colors, which only change the marble if they are not present:
assert(m3.addColor('Red') == False) # False means the color was not added,
                                    # because it was already there
# and no changes here:
assert(m3.colorCount() == 2)
assert(str(m3) == '<Marble with colors: blue, red>')
assert((m3 == m4) and (m3 != m5))

# Once more, but with a new color:
assert(m3.addColor('green') == True) # True means the color was added!
# and so these all change:
assert(m3.colorCount() == 3)
assert(str(m3) == '<Marble with colors: blue, green, red>')
assert((m3 != m4) and (m3 == m5))
```

```
# A ConstantMarble is a marble that never changes its color:
m6 = ConstantMarble('red,blue')
assert(isinstance(m6, Marble))
assert(str(m6) == '<Marble with colors: blue, red>')
assert(m6.addColor('green') == False) # constant marbles never change!
assert(str(m6) == '<Marble with colors: blue, red>')
assert(Marble.getMarbleCount() == 6) # we have created 6 marbles so far

# A DarkeningMarble is a marble that prefixes 'dark' to any colors
# that are added after it is first created.
# Note: for full credit, you must use super() properly here!
m7 = DarkeningMarble('red,blue')
assert(isinstance(m7, Marble))
assert(str(m7) == '<Marble with colors: blue, red>') # not darkened
assert(m7.addColor('green') == True) # but green will become darkgreen
assert(str(m7) == '<Marble with colors: blue, darkgreen, red>')
assert(Marble.getMarbleCount() == 7) # we have created 7 marbles so far
print('Passed.')
```

3. Bonus/Optional: Code Tracing [2 pts]
Indicate what this prints. Very clearly circle your answer (and nothing else):

```
class A(object):
    def __init__(self, x): self.x = x
    def B(self, x): self.x += self.B(x); return x + self.x
    A = B
    def B(self, x): self.x += 2*x; return x + 2*self.x
class B(A):
    def A(self, x): return super().A(x) + super().B(x)
def f(a): return (a.A(3), a.x)
print(f(A(4)))
print(f(B(4)))
```