fullName:_____andrewID:_____ recitationLetter:_____

15-112 N23

Quiz2 version A (40 min)

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- You may not ask questions during the quiz, except for English-language clarifications. Read problem statements carefully and if you are unsure of how to interpret a problem, take your best guess.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in week 2.
- You may not use lists, list indexing, tuples, dictionaries, sets, or recursion.
- You may use string methods described in the notes that produce lists, but you may only loop over their return values as shown in the notes. (i.e. you may not store any lists in variables, index into them, use list methods etc.)
- We may test your code using additional test cases. Do not hardcode.
- Assume almostEqual(x, y) and rounded(n) are both supplied for you. You must write all other helper functions you wish to use, unless we specify otherwise.

Multiple Choice [10pts total]

Consider the following app code for questions MC1 and MC2:

```
from cmu_graphics import *

def onAppStart(app):
    resetShape(app)

def onKeyPress(app, key):
    shrinkShape(app)

def onStep(app):
    growShape(app)

def redrawAll(app):
    createShape(app)

def resetShape(app):
    app.size = 100

def createShape(app):
    drawRect(0, 0, app.size, app.size)

def growShape(app):
    app.size *= 1.05

def shrinkShape(app):
    app.size *= 0.9
```

MC1. [1pt each] Mark each of the following as part of the model, view, or controller. Select the best answer for each (fill in one circle per row).

createShape	○ Model	○ View	○ Controller
growShape	○ Model	○ View	○ Controller
shrinkShape	○ Model	○ View	○ Controller
onStep	○ Model	○ View	○ Controller
redrawAll	○ Model	○ View	○ Controller
app.size	○ Model	○ View	○ Controller

MC2. [1pt] Which of the following could be called from redrawAll without causing an MVC violation?
Select ALL that apply (fill in at least one square).

☐ resetShape ☐ shrinkShape ☐ growShape ☐ None of the above

MC3. [1pt] What is the output of the following code? Select the best answer (fill in one circle).

```
a = 'abcdef'
b = a[6:]
print(repr(b))
```

◯ a blank line

◯ ''

◯ It will crash with a syntax error

◯ It will crash with a runtime error

MC4. [1pt] How many newline characters (`'\n'`) does the following code print? Select the best answer (fill in one circle).

```
a = '''
a,b,c
d,e,f
'''
print(repr(a))
```

◯ 0

◯ 2

◯ 3

◯ 4

◯ 5

MC5. [1pt] What is the output of the following code? Select the best answer (fill in one circle).

```
if '' in 'abcdef':
    print('yes')
else:
    print('no')
```

- ○ yes
- ○ no
- ○ It will crash with a syntax error
- ○ It will crash with a runtime error

CT1: Code Tracing [10pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(s):
    if s.endswith('d'):
        s.replace('bad','good')
    print(s)
    s = s[:-3] + 'fun'
    t = ''
    n = 0
    for c in s:
        if not c.isdigit():
            t += c
        else:
            n += int(c)
    print(n,t)
    return s

ct1('programming15is112bad')
```

CT2: Code Tracing [12pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct2(s):
    t = s
    s = s + 'b'
    t = t + 'c'
    print(s + '-' + t)
    s = 'abcdba'
    t = ''
    v = chr(ord('b')-1)
    c = 0
    for w in s.split('b'):
        c += 1
        t = t + str(ord(w[0]) - ord(v))
    print(c)
    return t

print(ct2('a'))
```

Free Response 1: getWorstCities(data) [34 points]

Write the function getWorstCities(data) that takes a multiline string of daily temperatures and and returns a string stating the city with the highest temperature and the city with the lowest temperature The input data looks like this:

```
data = '''
Pittsburgh, 30
New York, 60
Anchorage, -10
'''
```

and should return the output string:

```
'The warmest city is New York and the coldest city is Anchorage.'
```

Important notes:

- The data collection is not perfect so some of the data may contain blank lines.
- You can assume the input always has at least one non-empty line of data.
- You can assume that the temperatures are always integers and not floats.
- To break ties, return the city that appears first in the data.

Here are some test cases:

```
def testGetWorstCities():
    data='''
    Pittsburgh, 30
    New York, 60
    Anchorage, -10
    '''
    assert(getWorstCities(data) ==
           'The warmest city is New York and the coldest city is Anchorage.')
    data='''
    Pittsburgh, 30
    New York, 60

    Anchorage, -10
    New Mexico, 67

    Boston, -11
    '''
    assert(getWorstCities(data) ==
           'The warmest city is New Mexico and the coldest city is Boston.')
    data='''
    Albany, 50
    '''
    assert(getWorstCities(data) ==
           'The warmest city is Albany and the coldest city is Albany.')
```

Begin your FR1 answer here.

Continue your FR1 answer here.

Free Response 2: Dot Animation [34pts]

Write an animation with the following features:

1. A large blue circle with radius 70 is located in the center of the canvas.
2. A small falling circle with radius 20 is initially located with its center at the top of the canvas, horizontally centered, and colored red.
3. Every second, the small falling circle moves down 10 pixels.
4. The color of the small falling circle becomes green whenever it is touching or overlapping the large blue circle. Whenever the circles are not touching or overlapping, the small falling circle is red.
5. Note that the small circle should remain visible when it passes in front of the large circle.
6. Whenever the center of the small circle reaches the bottom of the canvas, it resets to its original position with its center at the top of the canvas.
7. Whenever the large blue circle is clicked with the mouse, the small falling circle immediately moves to the top of the canvas. Other mouse presses should be ignored.

Note: You should not assume the size of the canvas. We provide the graphics function headers for you, but you may write any helper functions you would like.

Begin your FR2 answer here:

```
from cmu_graphics import *

def onAppStart(app):
```

Continue your FR2 answer here

```
def redrawAll(app):

def onMousePress(app, mouseX, mouseY):
```

Continue your FR2 answer here.

```
def onStep(app):
```

Continue your FR2 answer here. **You may write any helper functions here.**

```
runApp()
```

bonusCT: Code Tracing [1pt]

This question is optional, and intentionally quite difficult. Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def bonusCT(s):
        ans = ''
        while s:
            for i in range(0, len(s), 2):
                j = i % len(s)
                ans += s[j]
                s = s[: j] + s[j + 1:]
        return(ans)
    print(bonusCT('IloveAnimation!'))
```