15-112 Midterm #1a – Fall 2016
80 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- **You may not use any books, notes, or electronic devices during this exam.**
- **You may not ask questions about the exam except for language clarifications.**
- **Show your work on the exam (not scratch paper) to receive credit.**
- **If you use scratch paper, you must submit it with your andrew id on it, and we will ignore it.**
- **All code samples run without crashing. Assume any imports are already included as required.**
- **Do not use these post-midterm1 topics/constructs: sets, maps, recursion, or classes/OOP.**

DO NOT WRITE IN THIS AREA		
Part 1 (CT)	15 points	
Part 2 (RC)	10 points	
Part 3 (SA)	10 points	
Part 4 (FR / zeddish)	20 points	
Part 5 (FR / dotBoom)	25 points	
Part 6 (FR / rotatePerimeter)	20 points	
Part 7/bonus	5 points bonus	
Total	100 points	

1. **[15 pts] Code Tracing**

Indicate what each will print:

Statement(s):	**Prints:**

`def ct1(s):`
`  (a, t1) = ( [ ], '' )`
`  for i in range(len(s)):`
`  t2 = s[i : i//2 : -1]`
`  if (len(t2) > len(t1)):`
`  a.append(t2)`
`  t1 = t2`
`  while (a != [ ]):`
`  print(a.pop())`
`ct1('abcdef') # prints 3 values`
	_____
	_____
	_____

`def ct2(L):`
`  n = len(L)`
`  for i in range(n):`
`  M = L[i]`
`  if (i == 2): L[i] = L[i-1]`
`  M.reverse()`
`  for i in range(n):`
`  print(L[i][i])`
`ct2([[  0,  1,  2,  3],`
`  [  4,  5,  6,  7],`
`  [  8,  9, 10, 11],`
`  [ 12, 13, 14, 15]]) # prints 4 values`
	_____
	_____
	_____
	_____

`def ct3(L, x):`
`  for f in L:`
`  if (f == ct3): break`
`  x = f(x)`
`  if (f == h): print("h")`
`  print(x)`
`def g(x): return x*10`
`def h(x): return x+g(x)`
`L = [g, h, g, ct3, h, g, h]`
`ct3(L, 2) # prints 4 values`
	_____
	_____
	_____
	_____

2. **[10 pts] Reasoning about code**
For each function, find values of the parameters so that the function will return True. Place your answers in the box on the right of each function.

```
def rc1(n):
    assert(isinstance(n, int) and
           (n >= 0) and (n < 1234))
    x = y = 0
    for i in range(2, 1234):
        if ((i % (i//2) > 0) and
            ((i//10) == (i%10))):
            (x, y) = (y, i)
    return (x == n)
```

n = _____

```
def rc2(M):
    assert(isinstance(M, list))
    (i, n) = (3, 1)
    for val in M:
        if (int(str(i)*n) != val):
            return False
        i -= 1
        if (i == 0): (i, n) = (3, n+1)
    return (len(M) == 7)
```

M = _____

3. [10 pts] Short Answer

Answer each of the following very briefly.

A. Give an example of 1-2 lines of Python code that demonstrates short-circuit evaluation.

B. Give an example of 1-2 lines of Python code that will exhibit a run-time error because strings are immutable.

C. Draw a picture that clearly shows where the anchor is located when drawing the text "ABC" in Tkinter with an anchor of E.

D. In our first implementation of Snake, we stored the location of the snake's head in data.headRow and data.headCol even though we technically did not need to do that. Very briefly (one or two well-chosen words is fine), state one advantage and one disadvantage of this approach.

advantage:

disadvantage:

E. Give an example of a common MVC violation -- that is, something that is explicitly disallowed by the rules for Model-View-Controller applications.

4. **[20 pts] Free Response: isZeddish(n) and nthZeddish(n)**
   Note: for full credit, you may not use strings for this problem, though lists are fine if you wish. For a 5-pt deduction, you may use strings.

   An integer m is zeddish (a coined term) if and only if:
   1) m is positive, and
   2) m is not a multiple of 10, and
   3) m contains non-leading 0's, and
   4) if n is the number formed by removing the 0's from m, n is a factor of m

   The first several zeddish numbers are 105, 108, 405, 1001, 1005, 1008, 2002, 2025,... For example, to verify that the number 105 is zeddish, we see that when we remove the 0's from 105 we get 15, and 15 is indeed a factor of 105. Similarly, 20025 is zeddish since 225 is a factor of 20025.

   With this in mind, write the functions isZeddish(n) that takes an integer value n and returns True if it is zeddish, and False otherwise, and also write the function nthZeddish(n), that takes a non-negative int n and returns the nth zeddish number, where nthZeddish(0) returns 105.

[this page is blank (for isZeddish and nthZeddish]

5. **[25 pts] Free Response: dotBoom Animation**

Assuming the run() function is already written for you, write init, keyPressed, mousePressed, redrawAll, and timerFired so that when the animation is first run:

A. a red circle of radius 5 starts centered along the top edge of the canvas; and

B. a blue circle of radius 10 starts centered along the left edge of the canvas; and

C. a score of 0 is shown in the bottom-left corner.

Game play proceeds as such:

D. the red circle sweeps top-to-bottom, and wraps around to the top when it hits the bottom;

E. the blue circle bounces left-and-right, reversing direction when it hits the left or right edge;

F. if the circles collide after they move, score +1 point and reset the circles to their original positions.

G. Each up-arrow or right-arrow speeds up only the red circle.

H. Each down-arrow or left-arrow slows down only the red circle unless it is already not moving, so K up-arrows should be undone by K down-arrows, with no net change of speed.

I. For mouse presses, if the press is inside the blue circle, score +1 point, otherwise, -1 point.

Make reasonable assumptions for anything not specified here, and in any case avoid hardcoding values (such as data.width, data.height, or data.timerDelay). Also, while the two circles move at different speeds, you must only use one timer and only one timerFired function.

[this page is blank (for dotBoom))

[this page is blank (for dotBoom))

6. **[20 pts] Free Response: getPerimeterCells and rotatePerimeter**

First, write the function getPerimeterCells(rows, cols), that you can abbreviate as gpc(rows, cols), that takes the dimensions of a rectangular 2d list of integers and returns a list of the (row,col) tuples of the cells on the perimeter of that list, starting at (0,0), and sweeping left-to-right across the top row, then top-to-bottom down the rightmost column, then right-to-left across the bottom row, and finally bottom-to-top along the leftmost column. An example should help make this clear:

```
L = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
(rows, cols) = (len(L), len(L[0]))
M = getPerimeterCells(rows, cols)
assert(M == [(0,0), (0,1), (0,2), (0,3),     # top left-to-right
             (1, 3),                         # right top-to-bottom
             (2, 3), (2, 2), (2, 1), (2, 0), # bottom right-to-left
             (1, 0),                         # left bottom-to-top
            ])
```

Note that getPerimeterCells only takes the dimensions of L, and not L itself, so the actual values in L do not matter here. Also, be sure not to double-count the corner cells!

After you write getPerimeterCells(rows, cols), and using that function as a useful helper function, write the function rotatePerimeter(L, n), that you can abbreviate as rp(L, n), that takes a rectangular 2d list of integers L and a possibly-negative int n, and destructively modifies L by rotating the values along the perimeter of L in a clockwise direction by n cells (if n is negative, this rotation is counter-clockwise). For example:

```
L = [[1,  2,  3,  4],
     [5,  6,  7,  8],
     [9, 10, 11, 12]]
rotatePerimeter(L, 1)
assert(L == [[ 5,  1,  2, 3],
             [ 9,  6,  7, 4],
             [10, 11, 12, 8]])
```

Notice the values on the perimeter are rotated 1 cell clockwise, and that the two interior cells (where the 6 and 7 are) are unchanged. Here is another example:

```
L = [[1,  2,  3],
     [4,  5,  6],
     [7,  8,  9]]
rotatePerimeter(L, -2)
assert(L == [[ 3,  6,  9],
             [ 2,  5,  8],
             [ 1,  4,  7]])
```

Here, the perimeter is rotated 2 cells counter-clockwise, and the interior cells (just the 5) are left unchanged.

For full credit, you must use getPerimeterCells(rows, cols) in a meaningful way here. For example, you could use it to help create a 1d list of the perimeter values in clockwise order. Then after rotating that 1d list, you could use the result of getPerimeterCells(rows, cols) a second time to assign the rotated values back into the original 2d list.

Write your answer on the following page.

[this page is blank (for rotatePerimeter)]

[the top of this page is blank

Bonus/Optional: **[2.5 pts] What will this print?**

```
def bonusCt1(m = 0):
    for x in range(1000):
        try: m = int(str(x), base=3)
        except: pass
    return m
print(bonusCt1())
```

Bonus/Optional: **[2.5 pts] What will this print?**

```
def bonusCt2(a, b, d=0):
    if (a > b): return [ ]
    elif (a == b): return [ d ]
    else:
        m = (a+b)//2
        return bonusCt2(a, m-1, d+1) + [d] + bonusCt2(m+1, b, d+1)
print(bonusCt2(0, 6))
```