

**15-112**  
**Spring 2019 Midterm 1**  
**February 21, 2019**

**Name:**

**Andrew ID:**

**Recitation 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 to receive credit.
- You may use the backs of pages as scratch paper.
- You may complete the problems in any order you'd like; you may wish to start with the free response problems, which are worth most of the credit.
- All code samples run without crashing unless we state otherwise. Assume any imports are already included as required.
- Do not use these concepts: sets/dictionaries, objects, recursion

Don't write anything in the table below.

Question	Points	Score
1	10	
2	10	
3	10	
4	25	
5	20	
6	25	
Total:	100	

## 1. Short Answer

Answer each of the following *very briefly*.

- (a) (3 points) The following snippet of code may crash on some inputs, even if the inputs have the correct types (list and int). In the three boxes below, give an example of an input pair with correct types that would cause the code to crash, explain why it crashes, and show how to fix the bug.

```
def mystery(lst, i):  
    if lst[i] == 0 and 0 <= i < len(lst):  
        lst[i] = 42  
    else:  
        return False  
    return True
```

**Input Pair:**

**Why does it crash?**

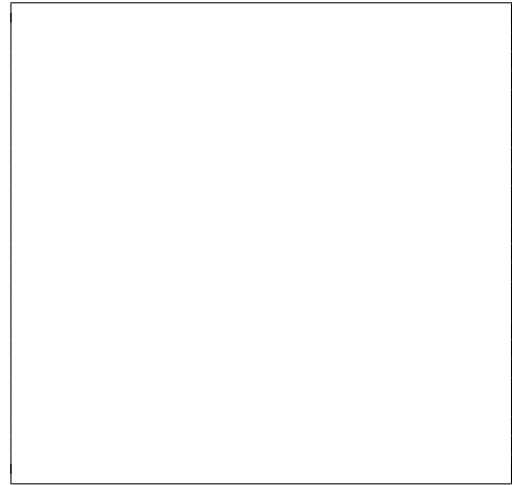
**How would you fix it?**

- (b) (1 point) In time-based animations, if we set `data.timerDelay = 1000`, how many times is `timerFired(data)` called per second?

- (c) (2 points) The following function checks if a string is a palindrome. The function works fine but has many style violations. List **two different style rules** from the 15-112 style guide that this code violates, and **circle where each rule is violated in the code**.

```
# Kermit Frog (andrew id: kfrog)
# Section 42

#Checks if word is the
# same forwards and back
def is_palindrome(s):
    a = True
    for i in range(len(s)):
        if (s[i] != s[-1-i]):
            a = False
        elif (s[-1-i] != s[i]):
            a = False
        else:
            continue
    return a
```



- (d) (2 points) Imagine the following animation: We draw a box that is larger than our canvas, and then add a bouncing ball inside. A score counter increases over time whenever the ball is visible, and stayed constant whenever the ball is offscreen. We can use the arrow keys to move (or scroll) our view, and clicking the ball pauses the animation. When the animation is paused, the ball changes to a different color. For the following components of this animation, mark whether these components should be considered as part of the Model, View, or Controller.

1. Clicking the ball pauses the animation
 

<input type="radio"/> Model	<input type="radio"/> View	<input type="radio"/> Controller
-----------------------------	----------------------------	----------------------------------
2. The paused state is stored as boolean value
 

<input type="radio"/> Model	<input type="radio"/> View	<input type="radio"/> Controller
-----------------------------	----------------------------	----------------------------------
3. A canvas-sized borderless rectangle is used to create a colored background
 

<input type="radio"/> Model	<input type="radio"/> View	<input type="radio"/> Controller
-----------------------------	----------------------------	----------------------------------
4. The position of the ball is stored as a list of two integers
 

<input type="radio"/> Model	<input type="radio"/> View	<input type="radio"/> Controller
-----------------------------	----------------------------	----------------------------------
5. Pressing the arrow keys lets us scroll across the animation
 

<input type="radio"/> Model	<input type="radio"/> View	<input type="radio"/> Controller
-----------------------------	----------------------------	----------------------------------

- (e) (2 points) The test function for `isPrime(n)` shown below is missing some important cases. Write two important and different test cases (as assert statements) that need to be added for the test function to work properly. Note that only number inputs are allowed (so `isPrime("foo")` is not a valid answer).

```
def testIsPrime():  
    print("Testing isPrime()...", end="")  
    assert(isPrime(2) == True)  
    assert(isPrime(7) == True)  
    assert(isPrime(131) == True)  
    print("...done!")
```

```
assert(isPrime(      ) ==      )
```

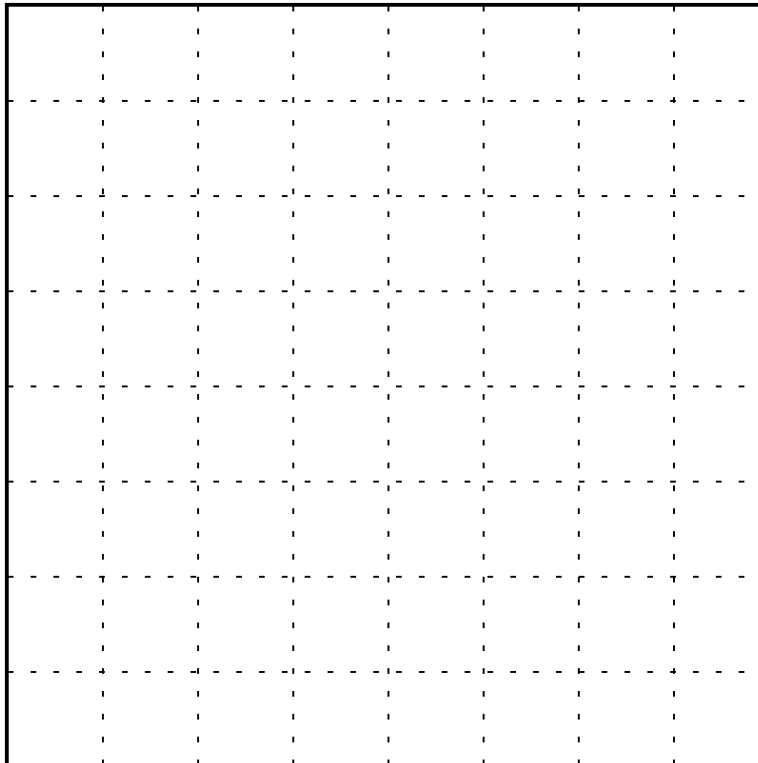
```
assert(isPrime(      ) ==      )
```

## 2. Code Tracing

Indicate what each piece of code will print. Place your answer (and nothing else) in the box below each piece of code.

- (a) (3 points) Note that the box below is the canvas, with a width and height of 400px, and each of the small boxes is 50px by 50px. In the code, `w` and `h` are the width and height. You may assume the code to create a tkinter canvas has already been run.

```
def drawCt1(canvas, w, h):  
    a = 300  
    canvas.create_polygon(0, h/2, w, h/2, w/4, h/4,  
                          fill = "", outline = "black")  
    canvas.create_oval(w/2, h/2, a, a)  
    canvas.create_text(a/3, a, text = "Yay", anchor = "w")  
  
drawCt1(canvas, 400, 400)
```



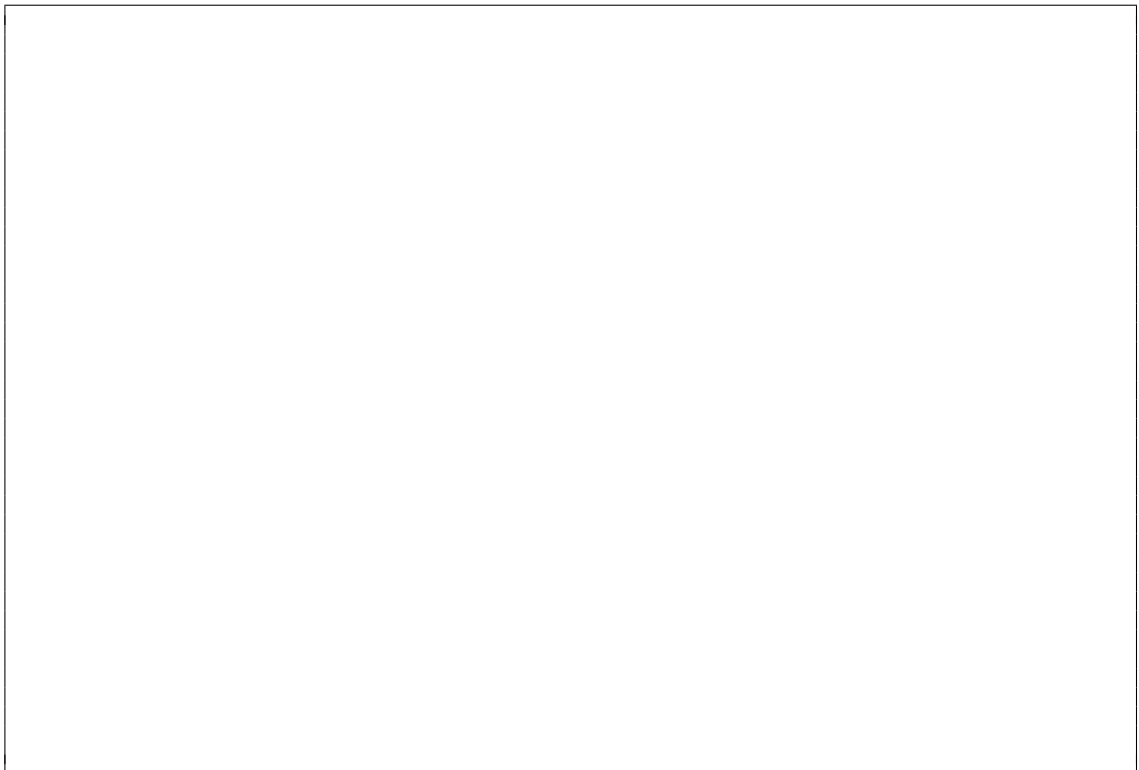
(b) (7 points) CT 2

```
import copy
def ct2(a):
    b = a
    c = copy.copy(a)
    d = copy.deepcopy(a)

    a.append("go")
    b[2] = "day"
    c.insert(1, "Q")
    d.extend(d[0])
    b[1][0] = 42
    c[0].pop()
    d[0][0] = 9

    print("a", a)
    print("b", b)
    print("c", c)
    print("d", d)

ct2([ [1, 2], [3, 4], "snow" ])
```



### 3. Reasoning Over Code

For each function, find parameter values that will make the function return True. Place your answer (and nothing else) in the box below each block of code.

**Note:** in both of these problems, you can get partial credit if you can't figure out the correct answer. See the note above each problem to determine how. In each case, if your answer in the left box is correct, you'll get full points; if it's wrong, we'll look at the partial credit box.

(a) (5 points) ROC 1

**Partial Credit:** Write in the right-side box the values that will be printed when the function `roc` is run on the correct input, in the correct order.

```
def f(x):
    print("A", x)
    x = h(x)
    y = 3
    print("B", x)
    return x + g(x, 1)

def g(x, y):
    print("C")
    return x ** y

def h(x):
    print("D")
    x = x + 4
    return x // 2

def roc(x):
    return f(g(h(x), 2)) == 40
```

**Answer:**

**Partial Credit:**

(b) (5 points) ROC 2

**Partial Credit:** Write in the right-side box what is printed when the function is run on the correct input.

```
def roc2(s):
    assert('n' not in s)
    a = ''
    for i in range(len(s)):
        b = len(s) - 2
        if s[i] == str(b):
            a = a + 'n'
        else:
            a = a + s[i]

    print(a)

s = a[2:0:-1] + a

s = s[::-1]
return s == 'banAna'
```

**Answer:**

**Partial Credit:**



4. (25 points) **Free Response:** `nthFactorMatch(x, n)`

Write the function `nthFactorMatch(x, n)` which takes two ints, `x` and `n`, and returns the `n`th number after `x` which has the same number of prime factors as `x`. A prime factor is a prime number that divides `x` with no remainder. A number can have multiple prime factors if it can be divided by the factor multiple times- for example, 12 has three prime factors, 2, 2, and 3. You may assume that `x` is an integer greater than 1, and that `n` is a positive integer.

**Example:** `nthFactorMatch(22, 1)` would return 25. We can derive this because 22 has two prime factors (2 and 11); of the following numbers, 23 has one prime factor (23), 24 has four prime factors (2, 2, 2, and 3), and 25 has two prime factors (5 and 5). Since 25 matches the factor count and is the 1st matching number we found, it is returned. `nthFactorMatch(22, 2)` would return 26, because 26 also has two prime factors (2 and 13) and is the 2nd match found (after 25).

**Note:** some solutions may use `isPrime(n)`. If yours does, you may call the function without writing it; assume the code has already been provided.

Additional Space for Answer to Question 4

5. (20 points) **Free Response:** `minHand(1st)`

Write the function `minHand(1st)`, where `1st` is a list of strings such as `['cat', 'dog', 'axolotl']`. The function should return the **shortest possible list of letters** such that any **single** word in `1st` can be spelled.

Recall the scrabble problem, where a ‘hand’ (represented by a list of single-letter strings) may contain the necessary characters to spell one or more words from the word list.

**Example 1:** If `1st==['axolotl', 'cat']`, a working function might return:

`['a', 'x', 'o', 'l', 'o', 't', 'l', 'c']`. **The order of the letters does not matter.**

- For our first word, ‘axolotl’, `1st` must include ‘o’ and ‘l’ twice, because we need two of each of those characters to spell ‘axolotl’. Each other character in ‘axolotl’ should appear exactly once.
- For ‘cat’, the second element of the list, we’ll need to add ‘c’ to our result, but ‘a’ and ‘t’ are already present in sufficient quantity for us to spell ‘cat’ so we don’t add those.

**Example 2:** If `1st==['bat', 'cat', 'dad', 'add', 'baa']`, a working function might return: `['b', 'a', 't', 'c', 'd', 'd', 'a']`.

- Note that ‘dad’ and ‘add’ require two ‘d’ characters, and ‘baa’ requires two ‘a’ characters.
- **We shouldn’t have multiple ‘t’ characters** or any other duplicates though, because we only need to be able to spell one word at a time.

**Note:** You may assume that the strings in `1st` only contain lowercase letters, and `1st` will not be empty.

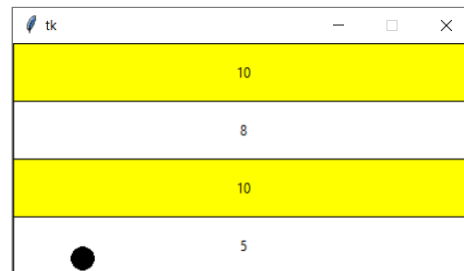
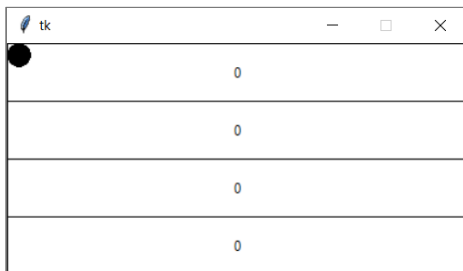
Additional Space for Answer to Question 5

6. (25 points) **Free Response: Animation**

Assuming the `run()` function is already written, write `init(d)`, `keyPressed(e, d)`, `mousePressed(e, d)`, and `redrawAll(c, d)` so that when the animation is first run, the screen is split into four horizontal rectangles that each have a number in the center and a black circle with a radius of 10px is placed in the top left corner. The animation proceeds as follows:

- When you click on the screen, the circle moves to be centered where you clicked.
- When the user presses the up arrow key or down arrow key, the circle moves 10 pixels up or down, respectively. The circle may move offscreen this way.
- Whenever the circle moves (either through mouse clicking or arrow movement), find the row that the center of the circle is located inside. The number in that row has one added to it.
- As soon as there is a non-zero number in at least one of the rows, the row(s) with the highest number is highlighted yellow. This highlighting should update appropriately as the row numbers update.

Make reasonable assumptions for anything not specified here. Do not hardcode values for `d.width` or `d.height`. We recommend that, to save time writing, you abbreviate `canvas`, `event`, and `data`: use `c`, `e` and `d`, respectively. **You must follow the MVC framework in your solution**; a solution with an MVC violation will receive an automatic 5 point deduction. To provide clarity, the image shown below on the left shows the starting screen; the image on the right shows a possible screen after user interaction.



Additional Space for Answer to Question 6

Additional Space for Answer to Question 6