15-112 Spring 2019
Practice Midterm 2
80 Minutes

Name:

Andrew ID:

Recitation Section:

- **This exam was written by TAs and may not reflect the true difficulty or length of the actual midterm.**
- 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. Nothing written on the back of any pages will be graded.
- 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.

Don't write anything in the table below.

Question	Points	Score
1	10	
2	10	
3	15	
4	15	
5	15	
6	20	
7	15	
Total:	100	

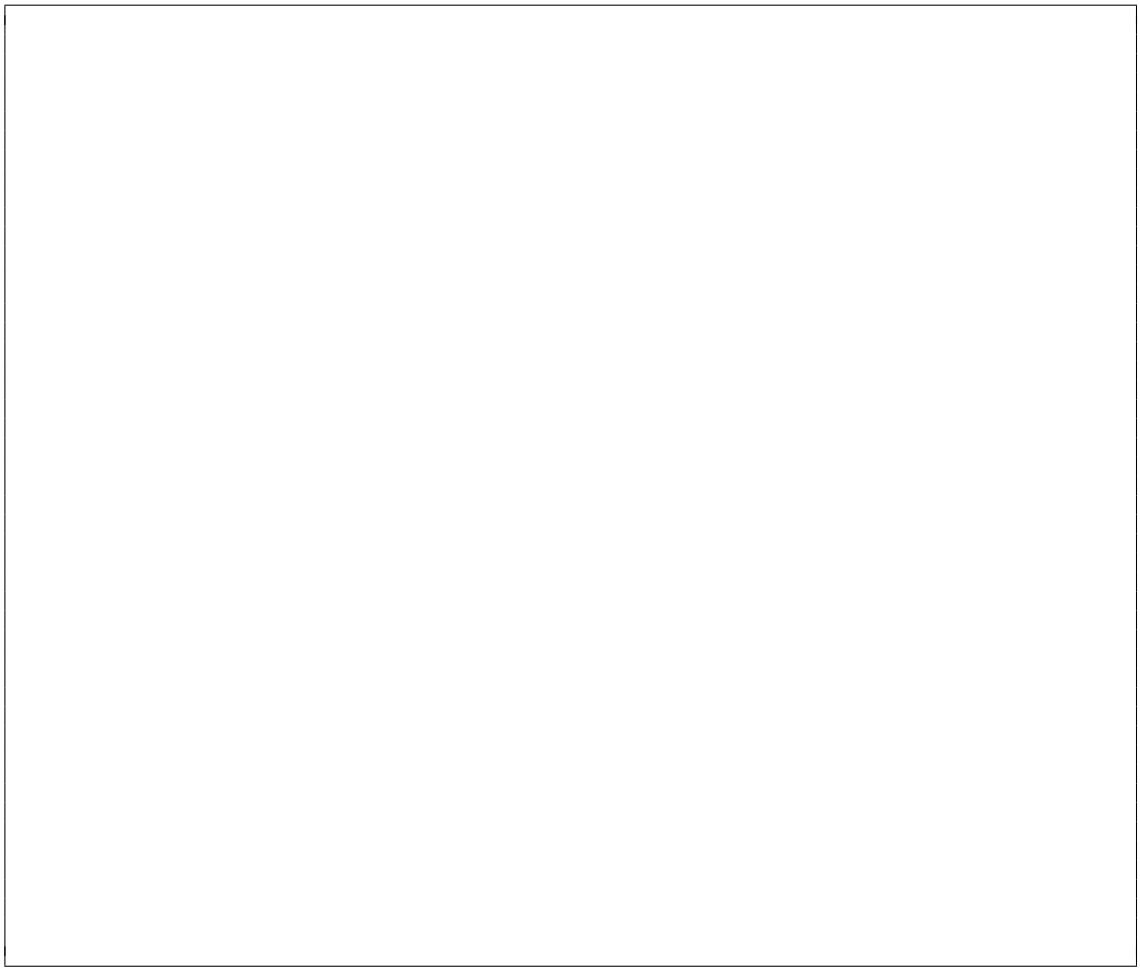
1. **Code Tracing**

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

   (a) (6 points) CT1

```
def ct1(L, depth=0):
    print("\t" * depth + str(L))
    if len(L) == 1:
        result = L[0] ** 2
    elif isinstance(L[0], str):
        result = ct1(L[1:], depth + 1) + ct1([L[-1]], depth + 1)
    else:
        result = ct1(L[1:], depth + 1)
    print("\t" * depth + "->" + str(result))
    return result

ct1(["yeet", 112, "yo", 6])
```



(b) (4 points) CT2

```
def ct2(d):
    for k in d:
        if k % 2 == 0:
            d[k + 1] = k + 1
    a = set()
    for k1 in d:
        for k2 in d:
            if k1 != k2 and d[k1] == d[k2]:
                a.add((k1, k2))
    return a

d = dict()
for i in range(6):
    d[i] = i + 1

print(ct2(d))
```

2. **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.

(a) (5 points) ROC1

```
def roc1(d):
    assert(len(d) <= 5)
    s = ""
    for k in sorted(d.keys()):
        if k % 2 == 0:
            s += 'x'
        elif d[k][::-1] == d[k]:
            s += d[k]
        else:
            s += d[k][:: -1]
    return s == 'teeyxyay'
```

(b) (5 points) ROC2

```
def roc2(x):
    def f(x, y):
        if x == 0:
            return y
        return f(x//10, x % 10 + y)
    def g(x, d):
        if x == 0:
            return d == 3
        return x % 10 > (x//10) % 10 and g(x//10, d+1)
    return g(x, 0) and f(x, 0) == 7
```

3. **Short Answer**

Answer each of the following *very briefly*.

(a) (1 point) Briefly explain why mutable data types cant be used as items in a set or as keys of a dictionary. Your answer must incorporate the concept of *hashing*.

(b) (1 point) If a Level 0 Sierpinski triangle is a black triangle, draw a Level 2 Sierpinski triangle.

(c) (1 point) The runtime of mergesort on a list of length N is O(Nlog$N$). Consider a variant of merge sort where we split the list into 3 parts, recursively sort each part and then merge the 3 sorted parts. Would this change the big-O runtime? If you answer yes, write down what the Big-O will be now. If you answer no, please explain your reasoning.

(d) (2 points) Recall the list method **.count** that takes as input an *item* you wish to count, and when called on a list L returns the number of times *item* appears in L. Here is an example to remind you how it works:

```
L = [1, 2, 3, 2, 4, 2]
print(L.count(2)) # prints 3
```

Write **one python expression** to implement our version of the count method which we will denote as myCount. Your solution must not use any loops, and must use some combination of map/filter/reduce. (you do not have to use all 3)

```
def myCount(L, item):
    # write your one Python expression here
```

(e) (1 point) Explain the difference between __str__ and __repr__

(f) (1 point) Consider our backtracking template. Briefly explain how we could modify it to return all the solutions for a given problem, rather than just the first solution that is found.

(g) (4 points) Write the big-O of the following function in terms of N.

```
def bigOh1(L):
    # assume L is an NxN (square) 2d list
    M=[]
    for i in range(len(L)):
        for j in range(i, len(L)):
            M.append(L[i][j])
    M.sort()
    return 112 if (112 not in M) else M.index(112)
```

(h) (4 points) Write the big-O of the following function in terms of N.

```
def bigOh2(s):
    # s is a string of length N
    myS = s * len(s)
    t = set()
    for i in range(len(s)):
        if s[i] not in t:
            t.add(s[i])
            k = 0
            while k < len(myS):
                k *= 2
                if str(k) in s:
                    print("wooo")
    return s
```

4. (15 points) **Free Response 1: visualizeRecursion decorator**

Write the decorator function **visualizeRecursion** that prints out the recursive depth and the returned result at each level of recursion for any generic recursive function (with any number of inputs and keyword inputs). To get a better idea of how it works, take a look at the example output for a recursive fibonacci function.

```
@visualizeRecursion
def fibonacci(n):
    if n == 0: return 0
    elif n == 1: return 1
    else: return fib(n - 1) + fib(n - 2)
```

Calling fib(3) will print (with the proper tabs and spacing):

```
        recursion depth: 2, result: 1
        recursion depth: 2, result: 0
    recursion depth: 1, result: 1
    recursion depth: 1, result: 1
recursion depth: 0, result: 2
```

Another example is:

```
@visualizeRecursion
def factorial(n):
    if n == 0: return 1
    return n * factorial(n - 1)
```

Calling factorial(4) will print:

```
                recursion depth: 4, result: 1
            recursion depth: 3, result: 1
        recursion depth: 2, result: 2
    recursion depth: 1, result: 6
recursion depth: 0, result: 24
```

Notice how the order in which the visualized calls are printed is such that the deepest recursive call is printed first with the biggest indent, and the initial function call is printed last.

You may assume that the decorator will only be applied to recursive functions.

Hint: consider having a variable to keep track of the depth in the decorator

Additional Space for Answer to Question 4

5. (15 points) **Free Response 2: palindromePartition**

Write the function `palindromePartition(s)` that given an input string `s`, returns a 1D list `L`, such that every element of `L` is a palindromic string and furthermore, all the strings in `L` concatenated together give the original string s or returns `None` if such a solution does not exist.

Note : We are only interested in meaningful outputs. Note that for any string s, a trivial solution is to return the list `L` of the individual characters in `s`. (strings of length 1 are palindromic by default) Therefore your solution must not return a list of individual characters in `s`.

For example:

```
palindromePartition("geeks") == ["g", "ee", "k", "s"] # "ee" has length > 1
palindromePartition("abcde") == None # only has the trivial solution
palindromePartition("racecar") == ["racecar"] # whole word is a palindrome!
```

You must use recursive backtracking to solve this problem.

Additional Space for Answer to Question 5

6. (20 points) **Free Response 3: OOPy Animation**

First, write the classes Circle and MovingCircle such that the following test cases pass. You must use inheritance properly.

```
###Make a Circle with a given position, radius and color
c = Circle(x=50, y=100, r=5, c="blue")
assert(c.x == 50)
assert(c.y == 100)
assert(c.radius == 5)
assert(c.color == "blue")
### Circles can be compared, and are considered to be the same so long
### as they have the same size and color
d = Circle(r=5, x=10, y=20, c="blue") # notice args can be given in any order
assert(d == c)
e = Circle(c="blue", x=10, y=20, r=3)
assert(d != e)
assert(str(c) == "blue circle of radius 5, at position (50, 100)")
### Moving circles (abbreviated as MC) are circles that move upon every timerFired.
### First arg is velocity in x direction and second arg is velocity in y direction.
mc = MC(5,10, x=200, y=300, r=5, c="blue")
assert(mc.velocityX == 5)
assert(mc.velocityY == 10)
assert(mc.move())
assert(mc.x == 205)
assert(mc.y == 310)
### We should be able to add both circles and moving circles to a set
### Remember that 2 circles (or moving circles) are the same if they have the
### same radius and color
s = set()
s.add(c)
assert(c in s)
assert(d in s)
assert(mc in s)
assert(e not in s)
```

Next, using the classes you have written, assuming the run function has been written for you, finish up your animation code with the following specifications:

- Add in a draw method that draws the circles and moving circles as required.
- A new circle with random x,y location, random radius between 5-40 and a random color chosen from either red, blue, yellow or green) must be spawned every second.
- A new moving circle with random x,y location, random radius between 5-40, random x and y velocities (between 5 and 10) and a random color chosen from either red, blue, yellow or green) must be spawned every 5 seconds.
- You must use a set as part of *data* to keep track of all the circles and moving circles.

- Every timer fired, the moving circles should move according to their respective x and y velocities.
- When spawning circles or moving circles ensure that you only spawn unique circles. If a circle or moving circle with the same radius and color already exists, you should not create this circle and try to spawn a new circle/moving circle.

Additional Space for Answer to Question 6

7. (15 points) **Free Response 4: findTriplets**

   Write the function **findTriplets(L)** that takes as input a list L of integers of length N and returns a set of all triplets in the list whose sum is equal to 0. For example, if the given list is [-1, 0, -3, 2, -1], you should return {(1, 0, -1), (-3, 2, 1)} (or any permutation of those numbers). If there is no valid triplet, you should return the empty set.

   You may assume that L is a list containing only integers.

   To receive any credit for this problem, your solution must run in O(N^2) time where N is the length of the input list.

Additional Space for Answer to Question 7