15-112 Midterm #2a – Spring 2017
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.**
- **When problems specify to use recursion, you must use it meaningfully. In such cases, you may use wrapper functions or add optional parameters if it helps.**

DO NOT WRITE IN THIS AREA		
Part 1 (CT)	15 points	
Part 2 (RC)	5 points	
Part 3 (SA)	5 points	
Part 4 (Big-Oh)	10 points	
Part 5 (FR/sectionMap)	15 points	
Part 6 (FR/A-and-B)	20 points	
Part 7 (FR/fractals)	10 points	
Part 8 (FR/summish)	20 points	
Part 9/bonus	5 points bonus	
Total	100 points	

1. **[15 pts] Code Tracing**
   Indicate what these will print:

Statement(s):	**Prints:**

```python
def ct1(L):
    if (L == [ ]):
        return [42]
    else:
        t = L[0] + L[-1]
        if (t % 2 == 0):
            return ct1(L[:len(L)//2]) + [-t]
        else:
            return [t] + ct1(L[1:])
L = [2, 5, 4, 3]
print(ct1(L)) # prints 4 integers
```

Statement(s):	**Prints:**

```python
def ct2(n):
    calls = list()
    def g(n):
        calls.append('g%d' % n)
        if (n <= 0): return 10*n
        else: return n + h(n // 2)
    def h(n):
        calls.append('h%d' % n)
        if (n <= 0): return n
        else: return n - g(n - 2)
    result = g(n)
    return str(result) +  ''.join(calls)
print(ct2(6))  # prints a string of length 9
```

Statement(s):	**Prints:**

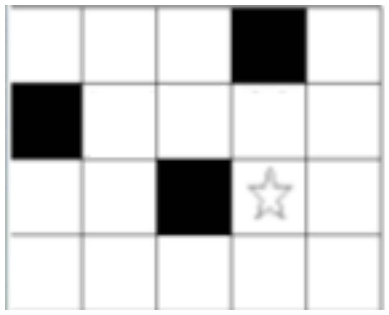
```
def ct3(d):
    t,u = set(),set()
    def f(s,t,u):
        for val in s:
            if (val in t):
                t.remove(val)
                u.add(val)
            else:
                t.add(val)
                if (val in u): u.remove(val)
    f(d.keys(), t, u)
    for key in d.keys():
        f(d[key], t, u)
    return [sorted(t), sorted(u)]
# prints a 2d list
print(ct3({ 2:{3}, 4:{2,0}, 3:{0,3} }))
```
	______________________

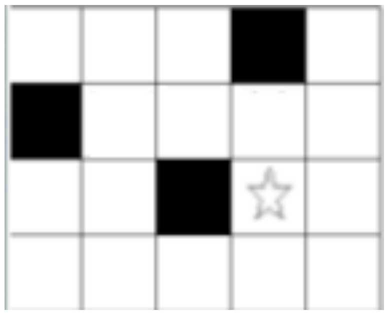
2. **[5 pts] Reasoning about code**
For this function, find values of the parameters so that the function will return True. Place your answers in the box on the right of the function.

```
def rc1(L):
    # L is a rectangular 2d list of ints
    def f(L, d=0):
        if (L == [ ]):
            return [ ]
        else:
            (rows, cols) = (len(L), len(L[0]))
            x, y = L[-1][0], L[0][-1]
            M = [row[1:] for row in L[1:]]
            return [x+d, y+d] + f(M, d+1)
    return (f(L) == [2, 4, 6, 8])
```
	L = ______________________

3. **[5 pts] Short Answer (just FloodFill)**
Answer the following very briefly.

A. Say we start with a 4x5 board and fill the black squares as shown below (in the code from our notes, the black squares would be green, and the white squares would be cyan). Then we right-click where the star is to start a floodFill from there. On the left, write the numeric labels for the depths that result in each cell after the floodFill completes. On the right, write the numeric labels for the ordinals.


depths


ordinals

Hint #1: the numbers on the left are depths, so some labels may occur more than once.
Hint #2: the numbers on the right are ordinals, so labels must occur only once each.
Hint #3: the first label is 0, not 1 (so a 0 should be where the star is).
Hint #4: Our floodFill code recursively tries to go up, then down, then left, then right.

4. **[10 pts] Big-Oh:** State the worst-case Big-Oh runtime for each of the following functions:

Function:	**Big-Oh:**

```
def bigOh1(L):
    # assume L is a 1d list
    N = len(L)
    s = set()
    for i in range(N**2):
        s.add(L[i % N])
    return sorted(s)
```
	______________

```
def bigOh2(s):
    # assume s is a set
    N = len(s)
    d, L = dict(), list()
    d[N//3] = N**4
    for i in range(N):
        for j in range(i, i+3):
            if (j not in s):
                L.append(j)
                d[i] = j
    for val in reversed(L):
        d[val] = L.index(val)
    return d
```
	______________

```
def bigOh3(L):
    # assume L is an NxN (square) 2d list
    # assume selectionSort, mergeSort, and
    # binarySearch are written as usual
    N = len(L)
    M = [ ]
    for A in L:
        mergeSort(A)
        for val in A: M.append(val)
    selectionSort(M)
    return binarySearch(M, 42)
```
	______________

```
def bigOh4(L):
    # assume L is an NxN (square) 2d list
    N = len(L)
    def f(i=N-1, d=0):
        if (i == 0): return 0
        else: return (L[i][i] + d + f(i//2, d+1))
    return f()
```
	______________

5. **[15 pts] Free Response: makeSectionMap(roster)**

This problem uses a 'roster', which is a list of tuples with 3 string values -- id, section, and role (where role is either 'student' or 'ta'). For example:

```
roster = [ ('fred', 'B', 'student'),
           ('wilma', 'B', 'ta'),
           ('betty', 'Q', 'student'),
           ('barney', 'Q', 'ta'),
           ('bambam', 'B', 'student'),
           ('pebbles', 'Q', 'ta')
         ]
```

With this in mind, write the function makeSectionMap(roster) that takes a roster and returns a sectionMap, which is a dictionary where the keys are the sections in the roster, and the values are a tuple of two sets -- a set of ta's and a set of students who are in that section. For example, the roster above returns the following sectionMap:

```
{ 'B': ( {'wilma'}, {'fred', 'bambam'} ),
  'Q': ( {'barney', 'pebbles'}, {'betty'} )
}
```

Also, if anyone appears more than once in the roster, your function must raise an appropriate Exception instead of returning normally. For example, if we added the tuple ('fred', 'Q', 'ta') to the roster above and called makeSectionMap again, this would result in the Exception 'Duplicate in roster: fred' being raised.

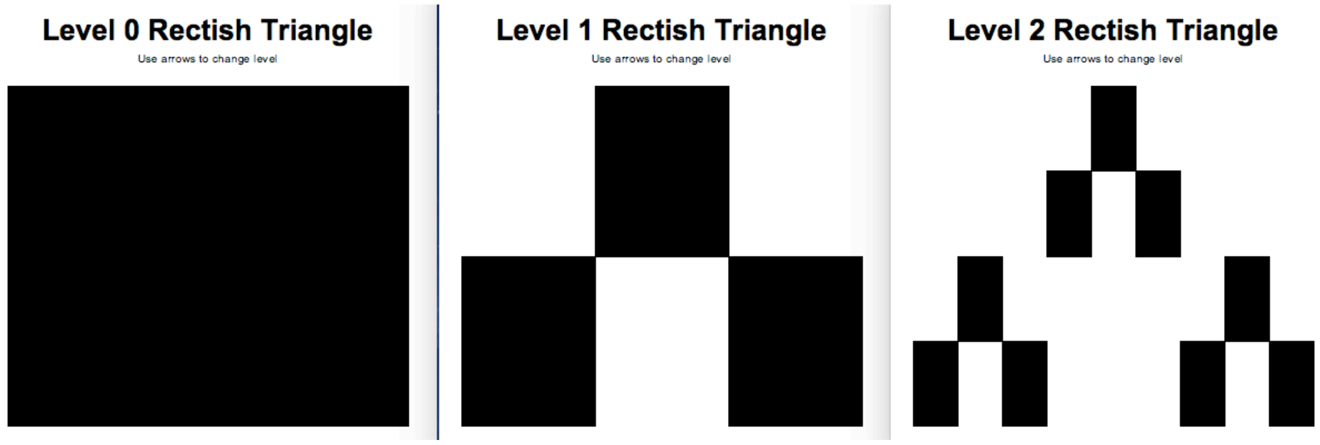
[this page is blank (for makeSectionMap)]

6. **[20 pts] Free Response: A and B classes**
Write the classes A and B so that the following test code runs without errors. Do not hardcode against the values used in the example test cases, though you may assume the types of those values match the examples. For full credit, you must use proper OOP, placing methods at the right level and using inheritance appropriately.

```
a1 = A(1,5,3,4,2,3) # A's constructor takes variable # of arguments
assert(a1.evens == [2, 4])   # evens are in sorted order
assert(a1.odds == [1,5,3,3]) # but odds are in the original order
assert(str(A(4,3,2,5) == 'A(evens=[2, 4],odds=[3, 5])'))

# Two A's are equal if their evens are equal, regardless of their odds.
assert(A(4,3,2,5) == A(2,3,4))
assert(A(4,3,2,5) != A(3,4,5))
assert(A(4,3,2,5) != "don't crash here!")

# Note: clearOdds and clearedOdds are not the same (only one is destructive)
a2 = A(4,3,2,5)
a2.clearOdds()
assert(str(a2) == 'A(evens=[2, 4],odds=[])')
a3 = A(4,3,2,5)
a4 = a3.clearedOdds()
assert(isinstance(a4, A))
assert(str(a3) == 'A(evens=[2, 4],odds=[3, 5])')
assert(str(a4) == 'A(evens=[2, 4],odds=[])')

s = set()
assert(A(1,2) not in s)
s.add(A(1,2))
assert(A(1,2) in s)
assert(A(1,2,3) in s) # because A's only use evens, not odds, for equality

b1 = B(3, 7) # creates an A using values [3,4,5,6,7]
assert(isinstance(b1, A))
assert(str(b1) == 'A(evens=[4, 6],odds=[3, 5, 7])')

# Note: only B's and no other A's can call shifted:
b2 = b1.shifted(2) # so instead of (3,7) it's now (3+2,7+2)
assert(str(b2) == 'A(evens=[6, 8],odds=[5, 7, 9])')
assert(type(b2) == B)
crashed = False
try: a = A(1, 2).shifted() # this should crash
except: crashed = True
assert(crashed == True)
print("Passed!")
```

[this page is blank (for A and B classes) [this page is blank (for A and B classes)

7. **[10 pts] Free Response: drawRectishTriangle**
Background: here, we will modify the Sierpinsky Triangle to instead create a Rectish Triangle (a coined term). This picture shows Rectish Triangles at levels 0, 1, and 2:


Level 0 Rectish Triangle
Use arrows to change level
Level 1 Rectish Triangle
Use arrows to change level
Level 2 Rectish Triangle
Use arrows to change level

You may assume most of the animation code for this program is already written for you. The only function you must write is drawRectishTriangle, which takes (canvas, x0, y0, x1, y1, level) and uses the canvas to draw a rectish triangle of the given level so that it fits in the rectangle with an upper-left corner of (x0, y0) and a lower-right corner of (x1, y1). Naturally, this function must use recursion in a meaningful way.

8. **[20 pts] Free Response: makeSummish(L)**

We will say that a list L is summish (a coined term) if for each sequence of 3 consecutive values in L, their sum is within 1 (inclusive) of a multiple of 10. For example:

```
L = [2, 4, 14, 23, 22]
```

We see:

```
 2 +  4 + 14 == 20, a multiple of 10
 4 + 14 + 23 == 41, within 1 of a multiple of 10
14 + 23 + 22 == 59, within 1 of a multiple of 10
```

So L is summish. However:

```
M = [2, 4, 6]
```

We see:

```
2 + 4 + 6 == 12, not within 1 of a multiple of 10
```

So M is not summish. Also, all lists shorter than length 3 are summish.

With this in mind, and using backtracking properly, write the function makeSummish(L) that takes a list L, and returns the new list M such that L and M contain identical values, so M is some re-ordering of L, but M is summish. Return None if no such list exists.

For example, makeSummish([2, 4, 6, 3]) might return [4, 2, 3, 6], among other possible answers (any legal answer is fine).

Note: you must use backtracking to receive credit. In particular, you may not create every possible permutation of L, as that would take far too long.

[this page is blank (for makeSummish))

9. **Bonus/Optional:**

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

```
def bonusCt1(f=lambda x:x%7):
    def g(f, n=8): return f if (n<=0) else lambda x:g(f,n-1)(x)+f(n)
    return g(f)(2)
print(bonusCt1(lambda x:x//2))
```

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

```
def bonusCt2():
    def f(n, e=10**-10): m = 1+1/n; return m if (abs(n-m)<e) else f(m)
    return round((f(1)*2-1)**2)
print(bonusCt2())
```