15-112 Fall 2022 Lecture 3
Midterm 1
80 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- **You may not use any books, notes, or electronic devices during this exam, except if you elect to use Coconut on your laptop for the FR's, as explained in lecture and on piazza.**
- **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 unless we state otherwise. Assume any imports are already included as required.**
- **Do not use these topics: sets/dictionaries and recursion.**
- **You may use almostEqual() and rounded() without writing them. You must write everything else.**

Do not write below here

Question	Points	Score
1. CT	30	
2. FR: nthSummy	20	
3. FR: Growing Moving Dot	20	
4. FR: getFlightTable	30	
5. Bonus	6 (bonus)	
TOTAL	100	

1. CT [30 pts, 6 pts each]
Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def ct1(n):
    r = 0
    for x in range(n, n**2, 2):
        if x%3 == 0:
            continue
        elif len(str(x)) > 1:
            break
        else:
            r = 10*r + x
    return r
print(ct1(4))
```

```
def ct2(s, t):
    d = 0
    r = ''
    s = s.replace('a','d').replace('cd', 'ab')
    t = t[::-1]
    t *= 2
    print(s, t) # Do not miss this!
    for i in range(min(len(s), len(t))):
        if s[i] == t[i]:
            r += chr(d + ord(s[i]))
            d += 1
    return r
print(ct2('abcad', 'abd'))
```

```
import copy
def ct3(L):
    M = L
    M = M + [1]
    L += [2]
    N = copy.copy(L)
    N.extend([4, L.pop(0)])
    return (M, N)
L = [5]
print(ct3(L))
print(L) # don't miss this!
```

```
def ct4(L):
    L.append(L[0])
    L[0].append(L[1][0])
    for i in range(1, len(L), 2):
        L[i-1], L[i] = L[i], L[i-1]
    return [i for i in range(len(L)) if len(L[i]) == 2]
L = [[2], [3], [4]]
print(ct4(L))
print(L) # don't miss this!
```

```
import copy
def ct5(L):
    M = copy.copy(L)
    N = copy.deepcopy(M)
    M.append([5])
    M[1].append(6)
    N.append(sum(L.pop(0)))
    return (M[1:], N)
L = [[2, 3], [4]]
print(ct5(L))
print(L) # don't miss this!
```

2. Free Response: nthSummy [20 pts]
Note: for this exercise, you may not use strings or lists.

We will say that an integer is summy (a coined term) if it is at least 10, and the sum of its digits equals the product of its digits. For example, 132 is summy because 1+3+2 == 6 and 1*3*2 == 6. The first few summy numbers are: 22, 123, 132, 213, 231...

With this in mind, write the function nthSummy(n) that takes a non-negative integer n and returns the nth summy number, where nthSummy(0) returns 22. Again: for this exercise, you may not use strings or lists.

Here are some test cases for you:

```
assert(nthSummy(0) == 22)
assert(nthSummy(1) == 123)
assert(nthSummy(2) == 132)
assert(nthSummy(3) == 213)
assert(nthSummy(4) == 231)
assert(nthSummy(5) == 312)
assert(nthSummy(6) == 321)
assert(nthSummy(7) == 1124)
```

This page is blank (for your nthSummy solution, if needed).

3. Free Response: Growing Moving Dot [20 pts]

Write an app such that:

1. When the app starts, a cyan dot of radius 50 is centered in the canvas.
2. When the user presses the left arrow, the dot moves 10 pixels to the left.
3. When the user presses the right arrow, the dot moves 10 pixels to the right.
4. Four times per second, the dot's radius grows by 20.
5. When the user presses 'p', the app is paused or unpaused.
6. When paused, the dot's radius does not grow, but the arrow keys still work.
7. When paused, if the user presses 's', the app takes one step, so the radius grows by 20.
8. If any part of the dot ever goes beyond the left or right edge of the canvas, whether from an arrow press or from the radius growing, then the dot returns to the center and its radius is set to 50.

Notes:

1. You may assume the canvas is 400x400.
2. To solve this, you need to write onAppStart, onStep, onKeyPress, and redrawAll.
3. You do not need to include any imports or the main function.

This page is blank (for your Growing Moving Dot solution, if needed).

4. Free Response: getFlightTable [30 pts]

Background: this problem starts with a multiline string of flight times between cities, like so:

```
flightInfo = '''
From Pittsburgh to LA takes 5.5 hours
From LA to Pittsburgh takes 5.5 hours
From LA to Seattle takes 3 hours
From Seattle to Pittsburgh takes 4.5 hours
'''
```

Every line in the flightInfo will always be of the form: `From <city1> to <city2> takes <time> hours`

We can convert this string into a 2d table of flight times like so (note that we added extra spaces to make the table easier to read):

```
    [['from/to',    'LA', 'Pittsburgh', 'Seattle'],
     ['LA',         '-',      5.5,       3.0     ],
     ['Pittsburgh', 5.5,      '-',       '-'     ],
     ['Seattle',    '-',      4.5,       '-'     ]
    ]
```

The cities on the left are the "from" cities, and the cities at the top are the "to" cities. Important points:

1. The top-left value in the table is always 'from/to'
2. The table's first column (the 'from' cities) and first row (the 'to' cities) include all the cities that occur anywhere in the string, in alphabetical order, regardless of the order they appear in the string.
3. The times in the table are always floats, even if they were ints in the string.
4. There are dashes in the table where the time was not included in the string.
5. You can assume that any from/to city pair occurs no more than once in the string.
6. You can assume that the string contains at least one non-blank line.
7. You can assume that city names do not contain any spaces.
8. You should ignore blank lines in the string.

With this in mind, write the function getFlightTable(flightInfo) that takes a multiline string of flight times and returns a 2d list of flight times, as just described. Hint: you may want to use both s.splitlines() and s.split() here.

Here is another test case for you (again with spaces added to the table to make it easier to read):

```
    flightInfo2 = '''
From NY to Boston takes 1.25 hours
From Scranton to NY takes 0.5 hours
'''
    assert(getFlightTable(flightInfo2) ==
           [['from/to',   'Boston', 'NY', 'Scranton'],
            ['Boston',      '-',    '-',     '-'    ],
            ['NY',         1.25,    '-',     '-'    ],
            ['Scranton',    '-',    0.5,     '-'    ]
           ])
```

This page is blank for your getFlightTable solution..

5. Bonus [6 pts, 2 pts each]

Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def bonusCt1(L):
    def f(g, d, L):
        while g(L): L = [v+d for v in L]
        while 0 in L: L.remove(0)
        return L
    g,d = max,-1
    while len(L) > 2:
        L = f(g, d, L)
        g,d = (min,+1) if g==max else (max,-1)
    return L
print(bonusCt1([1,2,3,4,2,3,4]))
```

```
def bonusCt2(n):
    def f(n): return n+f(n-1) if n else 1
    return f(n)
print(bonusCt2(300))
```

```
def bonusCt3(n):
    def f(n, g):
        a,b = 0,100
        while (b - a > 10**-6):
            c = (a + b)/2
            if g(c) > n: b = c
            else: a = c
        return round(c)
    return [f(n, lambda n:n**k) for k in range(1,4)]
print(bonusCt3(64))
```