Name: _____ Recitation: _____ Andrew Id: _____

15-112 Summer 2018 Quiz 3

Up to 50 minutes. No calculators, notes, books, or computers. Show your work!

1. (10 points) **Big-O:** What is the Big-O runtime of each of the following in terms of n?

Built-in Big-O Runtimes			
Strings		Lists	
`len(s)`	$O(1)$	`len(L)`	$O(1)$
`s.count(c)`	$O(n)$	`copy.copy(L)`	$O(n)$
`s.lower()`	$O(n)$	`L.append(k)`	$O(1)$
`str(i)`	$O(1)$	`L.pop()`	$O(1)$

```
import string
def bigOh1(s): # n is the length of the string s
    result = ""
    for c in s.lower():
        for c in string.ascii_lowercase:
            if s.count(c) == string.ascii_lowercase.count(c):
                result += c
    return result
```

```
import copy
def bigOh2(L): # n is the length of the 1D list L
    n = len(L)
    lst = copy.copy(L)
    for i in range(n):
        for j in range(n-1,0, -1):
            if L[i] == L[j]:
                lst.pop()
        for k in range(n-1,0, -1):
            if L[i] == L[j]:
                lst.append(k)
    return lst
```

```
def bigOh3(L): # n is the length of the 1D list L
    n = len(L)
    count = 1
    while n > 0:
        count *= 4
        n //= 4
    return count
```

```
def bigOh4(L): # n is the length of the 1D list L
    n = len(L)
    count = 0
    for i in range(n):
        if i == 0:
            break
        else:
            count += 1
    return count
```

2. (10 points) **Code Tracing**

   Indicate what the following program prints. Place your answers (and nothing else) in the box below the code.

```
import string
def ct1(s):
    if len(s) == 0:
        return ""
    index = string.ascii_lowercase.find(s[0])
    letter = string.ascii_uppercase[-index]
    return letter + ct1(s[1:]) + ct1(s[:-1])
print(ct1("abf"))
```

3. (30 points) **Free Response: highestPopulation** Given a dictionary `d` which maps cities to populations, write the function `highestPopulation(d)` that returns the city with the highest population in d. If multiple cities tie for the highest population then return a set of all of the cities with the highest population.

   For example `highestPopulation({"Seattle": 300000, "Pittsburgh": 100000})` would return the string "Seattle" since Seattle has the highest population.

   Calling `highestPopluation({"Seattle": 300000, "Pittsburgh": 300000, "Casper": 300 })` would return the set {"Seattle", "Pittsburgh"} since Seattle and Pittsburgh tie for the highest population.

Additional Space for Answer to highestPopulation

4. (50 points) **Free Response: fourColoring** Write the function `fourColoring(grid)` that takes a grid and returns a valid 4 coloring for the grid, or None if no such coloring exists.

   A grid has a valid four coloring if you can assign each cell in the grid to one of red, green, yellow, or blue such that no two neighboring cells have the same color (where a cell's neighbors are the 8 cells surrounding it).

   We represent the grid as a 2D list of strings where an empty cell is marked by ' ', and the colors are marked by 'G', 'R', 'Y', and 'B'.

   For example, given

```
grid = [
[' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' '],
[' ', ' ', ' ', ' ']]
```

   `fourColoring(grid)` would return:

```
[['R', 'G', 'R', 'G'],
 ['B', 'Y', 'B', 'Y'],
 ['R', 'G', 'R', 'G'],
 ['B', 'Y', 'B', 'Y']]
```

   Note: You must use recursive backtracking to solve this problem. You may use iteration as long as you still use recursive backtracking as your main approach. You also can assume that you have a function `getDirections()` that returns the list: [(0,1),(1,0), (-1,0), (0,-1), (-1, -1), (1,1), (1,-1), (-1,1)].

Additional Space for Answer to fourColoring