ORIE5270 - Homework 3
Due February 15th, 2019 at 1:25pm

You are not allowed to collaborate with your classmates. For this homework assignment you will implement a heap data structure in python and use it to model call-center service. Submit this assignment by pushing a folder HW3 to your ORIE5270 repository on cornell's github with your solution "hw3.py". **Make your repository private and add bdg79 as a collaborator**.

Problem 1 Make a class that implements a heap for people's names, which are always of the form "*Firstname Lastname*". We use the ordering on names given by alphabetically considering the last name (with ties being broken by alphabetically considering their first names). Your class (named NameHeap) should provide the following methods:

- **__init__()** Initialize an empty heap.
- **insertName(name)** Adds the given name to the heap. This method is allowed to take at most $O(\log(n))$ time, where n is the number of elements in the heap.
- **smallestName()** Returns the earliest (as defined above) name from the heap without modifying it. This method is allowed to take at most $O(1)$ time.
- **deleteSmallestName()** Removes the smallest name from the heap. This method is allowed to take at most $O(\log(n))$ time.
- **size()** Returns the number of elements in the heap. This method is allowed to take at most $O(1)$ time.
- **contains(name)** Returns true if the string name occurs anywhere in the current heap (and false otherwise). This method is allowed to take $O(n)$ time.

Problem 2 Make a class modeling a customer service call-center (with notoriously bad customer service due to artificial company policies). This call-center runs 24/7 and tracks customers solely based on the hour they called in and by their name. When determining the next customer to speak with, the call center will select the waiting customer from the oldest hour with the alphabetically first name (according to the ordering used in Problem 1). Your class (named CallCenter) should utilize multiple instances of NameHeap to provide the following methods:

- **__init__()** Initialize with no customers waiting at hour 0.
- **queueCustomer(name)** Add a customer with the given name and current hour to the underlying data structure. This method is allowed to take at most $O(\log(n))$ time, where n is the number of customers that arrived in the current hour.
- **dequeueCustomer()** Returns the name of the next customer to be processed and removes them from the underlying data structure. This method is allowed to take at most $O(\log(n))$ time, where n is the number of customers waiting from the oldest hour.
- **nextHour()** Ends the current hour, starting the next one. This method is allowed to take at most $O(1)$ time.
- **size()** Returns the number of customers currently waiting to be processed. This method is allowed to take at most $O(h)$ time, where h is the number of times nextHour() has been called.
- **contains(name)** Returns true if a customer of the given name is currently waiting, otherwise false. This method is allowed to take $O(n)$ time, where n is the total number of customers waiting.

As a simple test, the following simple test code should run and print out true nine times (but you should certainly test your code more extensively than this):

```
import hw3

heap = hw3.NameHeap()
heap.insertName("Bob Ross")
heap.insertName("Ben Grimmer")
heap.insertName("Lijun Ding")

print(heap.size()==3)
print(heap.smallestName() == "Lijun Ding")
print(heap.contains("Bob Ross"))

heap.deleteSmallestname()
print(heap.smallestName() == "Ben Grimmer")


callCenter = hw3.CallCenter()
callCenter.queueCustomer("Bob Ross")
callCenter.queueCustomer("Ben Grimmer")
callCenter.nextHour()
callCenter.queueCustomer("Lijun Ding")
callCenter.queueCustomer("Jim Renegar")

print(callCenter.size()==4)
print(callCenter.dequeueCustomer()=="Ben Grimmer")
print(callCenter.dequeueCustomer()=="Bob Ross")
print(callCenter.dequeueCustomer()=="Lijun Ding")
print(callCenter.contains("Jim Renegar"))
```