



Problem Set: Assignment: A07
Points: 10
Date Set: See Autograder
Course: CS218 - Data Structures

Semester: Fall 2019
Due Date: See Autograder
Instructor: Dr. Nauman

1 Heap Case Study

We have already created a max heap in class. This assignment intends to use it for a slightly practical usecase. We intend to create a priority queue type structure. We will have a list of people who need to be provided some sort of services (e.g. bill payment).

Each person will have a name, an age and a queue number which will tell us the order in which they arrived in the service center. For example, person A is of age 24 and arrived first in the service center, person B is 32 years old and came after person A. This information will be represented as follows:

```
people = [  
    Person('A', 24, 1),  
    Person('B', 32, 2)  
]
```

1.1 Person

First, write a person class with a constructor that takes in these three parameters and set them properly. Next, write a function `get_priority` in this class that returns the priority of the person. The priority of a person is:

$$\text{priority} = 100 - \text{queue_position}$$

However, there is a slight caveat: if a person is 40 years or older, their priority gets a bonus of 100 points. So, if a person has the age 29 and queue number 24, their priority will be 76 but if they are 41 years old (with the same queue number of 24), their priority will be 176. This formula ensures that people who arrive first will have a higher priority but people aged 40+ will have preference over everyone else.

Finally, in this class create the dunder `str` function that will return a string in the following format:

A 99

Where A is the name of the person and 99 is their priority. Make sure you test this function thoroughly before proceeding ahead.

1.2 Heap Sorting Persons

You now need to create a list of person objects and heap sort them based on their priority. Name this function `heap_sort`. So, if we give the following list to this function:

```
people = [  
    Person('A', 24, 1),  
    Person('B', 32, 2),  
    Person('C', 45, 3),  
    Person('D', 22, 4),  
    Person('E', 21, 5),  
    Person('F', 32, 6),  
    Person('G', 39, 7),  
    Person('H', 44, 8),  
]
```

```
    Person('I', 22, 9),
    Person('J', 29, 10),
    Person('K', 32, 11),
    Person('L', 31, 12)
]
```

the function should modify the list `people` so that person C is at the beginning (since they have a priority of 197), then we should have person H, then person A and so on. Make sure you understand why this is so and what affect it will have on your heap sort logic if you try to get the *highest* value out first instead of *lowest* value first as we did in class.

(Also make sure your `heap_sort` and associated functions are defined in the global scope and *not* within the `Person` class.)

2 Submission

Use `python run.py local` to ensure all tests are passing and then submit your assignment using `python run.py remote`

If you wish to request an extension, use the autograder UI to do so. Each student gets a maximum of 3 extension days per semester.