Lab #10

CS-2050 - Section D

Week of April 12, 2021

1 Requirements

In this lab, you will write a set of functions for maintaining **a descending priority queue**. The structure of the queue is *not clearly defined*, and you must design the queue structure to meet the performance constraints outlined in the functions below. There are at least two queue designs that could be used to achieve these performance constraints, which you have learned about in lecture.

Your implementation of the required functions should match the performance requirement *exactly*. The structure of your queue should reflect the expected performance of each function, and no functions should perform **faster or slower** than expected.

```
> lab.h
typedef struct Queue Queue;
    ...
typedef struct {
        float squareFeet;
        int baths;
        short houseNumber;
} House;

> lab.c
struct Queue {
    // What goes here is up to you
};
```

Info: In this lab, you will be graded for your implementation of the Queue structure. You may choose to include as many or as few Queue struct members as you like, and you may define additional struct types to complete this assignment. The names of struct types and members should be **clear and accurate** as to their purpose, and you should **leave comments** describing your implementation.

1.1 Support Functions

```
// Complexity: O(1)
Queue* initPQ();
// Complexity: O(n)
void freePQ(Queue *pq);
```

Info: These functions are required for grading purposes, but are not part of the testing for this lab. You are expected to implement these functions to support your implementation, but they are being counted as a single "function group" and will not be a significant part of your grade for this lab.

1.2 insertHouseDescendingPQ

```
// Complexity: O(n)
int insertHouseDescendingPQ(Queue *pq, House *house);
```

Info: Inserts the given **struct pointer** into the priority queue based upon the *houseNumber* member.

1.3 peekMinHouse

```
// Complexity: O(1)
House* peekMinHouse(Queue *pq);
```

Info: This function returns the object which is next to dequeue **without removing it**, in **O(1) time**

1.4 deQueueMinHouse

```
// Complexity: O(n)
House* deQueueMinHouse(Queue *pq);
```

Info: This function dequeues and returns the object which is next to dequeue in **O(n) time**

2 Notice

Grading: Total 28 points

1. Write required *support* functions
   * 4 points
2. Write required *insert* function
   * 6 points
3. Write required *peek* function
   * 6 points
4. Write required *dequeue* function
   * 6 points
5. Queue structure is properly formatted and supports performance requirements
   * 6 points

Notice:

1. All of your lab submissions must compile under GCC using the $-Wall$ and $-Werror$ flags to be considered for a grade.
2. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab policy document.