COP 3502 Fall 2021 Sections 1, 2 Recitation Program #4

I Can Guess the Data Structure!

Kattis link: https://open.kattis.com/problems/guessthedatastructure

For each recitation program, in order to get full credit, you must submit your solution to open.kattis.com and get your solution accepted on all test cases. In addition, each one will have some separate requirements to fulfill based on your code. When submitting your work to Webcourses, please carefully read the corresponding directions document before submitting all of your files.

NOTE: Over the course of the semester, you must submit TWO out of the four recitation programs. It is expected that while you are in recitation, you start working on each of them. But, afterwards, you can choose which two to finish up.

What This Program Is Testing

This program is testing simulation.

You are to implement a stack, queue and priority queue in C and use those implementations to answer the queries given. You may consult the implementations of these data structures on the course website. However, credit may be deducted if, even if your code passes the data, if you have issues with dynamically allocated memory.

Finally, explain why with the bounds given, it might be possible to get a solution to pass with an inefficient priority queue (O(n) delete max) and suggest different bounds that will ensure that an efficient priority queue must be implemented. **Please submit this in a separate text file.**

Note: for full credit you must have both an O(lg n) insert and O(lg n) delete maximum for your priority queue and O(1) push/pop for the stack and O(1) enqueue/dequeue for the queue.

What to Submit

Please submit the following:

1) Your source file, **datastruct.c.**

2) A screenshot of your solution's accepted status on Kattis. **This screen shot needs to include your name at the top right and all of the check marks as well as the starting of your code below.**

3) Text file **analysis.txt**, which explains why it might be possible for a solution with an inefficient priority queue to pass and suggest a different set of bounds that will require an efficient implementation of a priority queue. **(For your calculations, assume that 10^8 simple operations is the upper limit of what can run in the time limit.)**