CS2123 Data Structures Summer 2017

Assignment 3: Priority Queues

Due 7/5/17 by 11:59pm

1. Implementing a Priority Queue as a Sorted Doubly Linked List (15 points)

This is a program for simulating enrollment of students in a set of courses. Included in this is a waitlist which allows students to queue up for possible free space in the course. This free space may come from additional seats being added to the course or from students choosing to drop the course. The waitlist queue is a priority queue where some students are given a higher priority for being added to the course (e.g., seniors may be given a higher priority than freshman or majors given preference over non-majors).

The priority queue is represented using a sorted doubly linked list. Higher numbers are associated with higher priorities. The higher the priority a student has the closer to the front of the queue they should be. In the case of students with the same priority, the one who entered the queue first should be given preference.

There are two data files used for this assignment. The code for reading from these files has already been provided (see the main of the provided code). Students are specified by a 3 digit id and courses are specified by a 4 digit id. The "a3_EnrollData.txt" file specifies a sequence of three types of operations to be performed on the courses/students. You are responsible for implementing functions which execute these three operations:

(1) enroll - Attempt to enroll the student in the course
(2) addSeats - Add the specified number of seats to the course and adds students from the waitlist to fill these seats
(3) drop - Drop the student from the course and fill the empty seat from the waitlist

Some code has been provided for you to help in creating functions to execute the above three operations.

In addition to the above you also need to implement your priority queue using a doubly linked list. Specifically,

(1) enqueue - Add a new waitlist element to the priority queue. This element needs to be inserted in sorted order.
(2) dequeue - Remove and return the front element from the LL.