Homework 7

(Maximum 100 points)
Exercise 1 due beginning of class Friday October 23, 2015
Exercise 2 due beginning of class Wednesday October 28, 2015
Show the steps of deriving your answers.

1. (25 points) [Leftist heap: merge] Textbook Exercise 6.19.

2. (75 point) [Binary heap: application programming] Build a print scheduler using a priority queue represented with a *maximum* binary heap as the data structure. Each print request has the following associated fields: ID, user's login name, request time, priority, file size in bytes, and file handle.

Write a program that supports the following operations:

- int add(string login, string time, int priority, int size, int handle): add a new request to a queue; return a sequential ID# if successful, or FULL if the total file size of all pending requests exceeds the maximum spool size of 50MB. The input argument handle is not used in this exercise, so set it to NULL. The ID cycles in the range of 1 - 1024.
- int print_next(): return EMPTY if the queue is empty, or return the ID of the pending highest-priority request and delete it from the queue.
- int find_next(): return EMPTY if the queue is empty, or return the ID of the pending highest-priority request.
- int cancel(string login): delete all the requests made by the login user; return the number of deleted requests if successful, or return NONE if not found.
- int cancel(int ID): delete the request with the ID; return 0 if successful, or return NONE if not found.
- int status(): display information about all pending requests' ID, login, creation time, priority, file size, and file handle; return EMPTY if there's no pending request. The order of the print requests returned does not matter.
- Error codes: -1 for FULL, -2 for EMPTY, -3 for NONE. (If you add more error codes, document them in a README file.)

Follow the common practice of "good programming" (i.e., comments, formatting). Feel free to use the codes from the textbook (http://users.cis.fiu.edu/~weiss/dsaajava3/code/) or other literature. (Note the codes in the textbook are for *minimum* binary heap.) Cite the source. Submit your codes via Blackboard. Your program will be tested against a sequence of simple commands. Your program should read the commands one by one from a file, and write the commands and the return values to an output file. See the provided sample_input.txt and sample_output.txt.

Last modified: October 15, 2015