

CS 302 Data Structures
Spring 2008 - Dr. George Bebis
Programming Assignment 5

Due date: 5/6/2008

Specify and implement an updatable priority queue U_PQType which is **derived** from $PQType$. A standard priority queue, such as the one discussed in class, does not let you remove items after they have been added, except by popping them off when they (eventually) reach the front of the queue. Furthermore, if the priority of an item is increased or decreased after it is added to the priority queue, there is no way to tell the queue to adjust itself. U_PQType is different from $PQType$ in that it contains two new member functions: $Remove(item)$ and $Update(item, newItem)$. $Remove(item)$ should remove item from the queue; nothing should happen if item is not in the queue. $Update(item, newItem)$ should update item to newItem and move it if necessary; nothing should happen if item is not in the queue.

Upon execution of your program, it should first read the priority queue items from a file (i.e., a sample file is available from the course's webpage) and should create a heap using the items as a key. Once the heap has been built, it should display the following menu:

- (1) Dequeue
- (2) Enqueue
- (3) Remove
- (4) Update
- (5) Print

Option (1) removes the highest priority item from the queue. Option (2) inserts a new item onto the queue. Options (3) and (4) perform the operations described above. Finally, option (5) prints the items in the priority queue on the screen. You should explain clearly in your report the implementation of $Remove(item)$ and $Update(item)$ as well as their running time requirements.

Extra Credit (20 pts): Consider the problem of organizing a collection of computer user-ids and passwords. Each time a user logs in to the system by entering his or her user-id and a secret password, the system must check the validity of this user-id and password to verify that this is a legitimate user. Because this validation must be done many times each day, it is necessary to structure this information in such a way that it can be searched rapidly. Moreover, this must be a dynamic structure because new users are regularly added to the system and some users deleted from the system.

Upon execution of your program, it should first read the user-ids and passwords from a file (i.e., a sample file is available from the course's webpage) and create a binary search tree using the user-id as a key (assume unique user-ids for convenience). Once the tree has been built, it should display the following menu:

- (1) Add new user
- (2) Delete user
- (3) Verify user
- (4) Print users
- (5) Store data in file using inorder traversal
- (6) Store data in file using preorder traversal
- (7) Store data in file using postorder traversal

Options (1) and (2) simply add/delete new/existing users. When option (3) is selected, the user is supposed to enter a user-id and a password. Then, you should search the tree and a print message like "Valid User" or "Invalid User". When option (4) is selected, the users and their passwords should be printed out in alphabetical order. Finally, when options (5)-(7) are selected, the elements of the binary search tree should be stored to a file and execution should be terminated.

Your report should follow the general guidelines posted on the course's webpage ("How will the assignments be graded?")