Priority Queue

Topic(s): Priority Queues, Code Reusability
Date assigned: Friday, October 16, 2015
Date due: Wednesday, October 28, 2015, 9:15 am
Points: 40 pts

For this assignment, you are to implement a Priority Queue ADT in a file called pQueue.c using the header file pQueue.h. You can find this header file, as well as a pQueue.c file, on zeus in **/home/CS300Public/2015/05Files**. All of the data structures and function prototypes are defined in the header file. Further, each function prototype has been described to the point that you should be able to implement each function. The error codes that can be produced are listed for each function. Higher precedence error codes are listed first. The pQueue.c file contains data and a function to help you display error messages to the screen.

The Priority Queue must be implemented using the Dynamic List from the previous assignment as the base data structure. We will be reusing this Priority Queue later on, so make sure you have completely tested and debugged each operation. For simplicity, the priority queue will store a **single integer value** of user data as well as a **priority**.

In addition to implementing the Priority Queue data structure, you must provide a Makefile and test driver (pQueueDriver.c that produces an executable named **pQueueDriver**) that thoroughly tests your Priority Queue. The driver should display to the screen a series of SUCCESS or FAILURE messages, with enough description that a user can quickly spot broken functionality.

You may add any helper functions you need to pQueue.c. These helper functions must be marked **static** so they are not available outside the module. You may not alter pQueue.h in any way.

1. Your code is to be written in C using Eclipse. Programs written in other environments will not be graded. Create an Eclipse project named **DynamicPriorityQ**. This project must contain the directories: src, include, and bin.
2. The Makefile must contain the necessary targets to build the pQueueDriver as well as a clean, valgrind, and tarball targets. Typing **make** on the command line must build pQueueDriver.
3. Submit a file called cs300_5_PUNetID.tar.gz into the CS300 Drop Box by 9:15am on the day in which the assignment is due. This file must include your DynamicPriorityQ **AND** your updated DynamicList projects.
4. Submit a color, double sided, stapled packet of code by that same deadline. The packet must be in the following order:
   Priority Queue Driver (.h then .c if you have both, otherwise just .c)
   pQueue.c (do not print pQueue.h)
   Any extra .h/.c pairs you have. (do not include any code from the List project)
   Makefile
5. Test one function at a time. This will lessen your level of frustration greatly.
6. You are to use the coding guidelines from V6.3 of the coding standards.
7. The only changes to DynamicList you can make are to fix bugs.

8. You must insert DynamicPriorityQ into your Subversion repository; DynamicList should already be in Subversion and any bug fixes must be committed to Subversion.
9. **<u>IMPORTANT:</u>** When implementing your priority queue ADT, you are to use the functions from the DynamicList API and not access any DynamicList data directly. As an example, the function pqueueSize could access numElements of the DynamicList directly BUT must not. Instead, the function lstSize is to be used.

Goals for this assignment:

1. Reuse your DynamicList API.
2. Code and test your program one function at a time.
3. Write efficient/clean code
4. Use the debugger to effectively develop a correct solution
5. Thoroughly test your code.
6. No Valgrind errors.

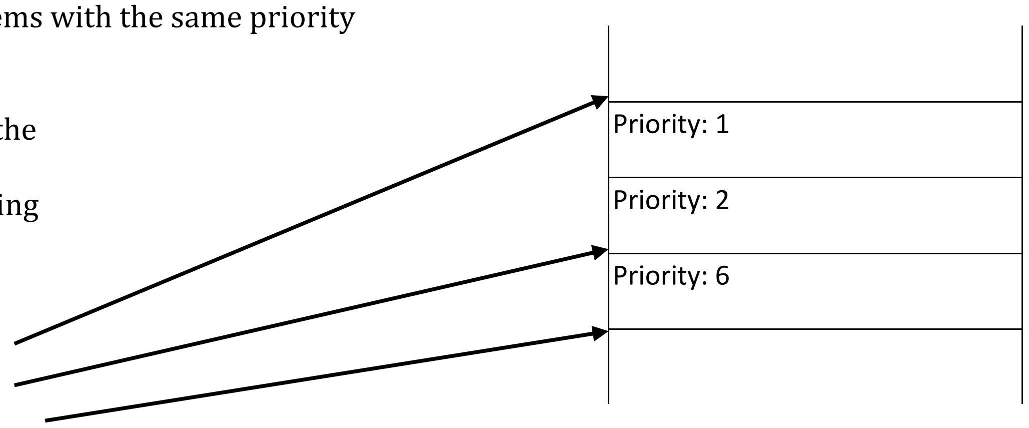
Priority in the Queue.

You must implement priority in your queue by inserting items into the queue using the priority value provided by the user. A priority of **zero** is the **highest priority**. A newly inserted item must be inserted:

1) ahead of all items with a lower priority
2) behind all items with the same priority

For example:
The Priority Queue on the right already contains some data. The following inserts will add data at the marked points.

insert Priority 0
insert Priority 2
insert Priority 10



♦ The function **pqueueChangePriority** accepts an integer (positive or negative) and adds that integer to the priority of *every* item in the queue.

♦ **There is only one deadline**. I expect you to start this project soon. Your priority queue data structure will likely be smaller than your pQueueDriver.

Using Eclipse, Makefiles, and Multiple Projects.

Since your DynamicPriorityQ relies on your DynamicList, which is in another Eclipse project, your **Makefile** may contain lines like the following.

```
bin/pQueue.o: src/pQueue.c include/pQueue.h ../DynamicList/include/list.h
	${CC} ${CFLAGS} -c src/pQueue.c -o bin/pQueue.o
```

In this example line, pQueue.o relies on pQueue.h as well as the header file from the list, which exists up a directory (to your workspace root) and then down in the DynamicList project's include directory. Your driver will also need to depend on the list.o file in the DynamicList project.

If you want to rebuild list.o via your Priority Queue Makefile you may need a line like this in your Priority Queue Makefile.[1]

```
../DynamicList/bin/list.o:  ../DynamicList/include/list.h  ../DynamicList/src/list.c
	cd ../DynamicList; make bin/list.o
```

This moves to the DynamicList directory and invokes make. The Makefile in DynamicList is read and list.o is rebuilt if necessary. Note that make executes *each line* of your file with a new shell so if you **cd** on one line and run a command on the next line, the command is run as if the **cd** had not been run.

You are most likely going to run into Eclipse problems with this project. Namely, Eclipse may not see an update to the DynamicList data structure while you are coding in the Priority Queue data structure and may produce errors *even if the Makefile succeeds in building your .o and executable files.*

If you right click a project, choose Properties, and select Project References you can mark which other projects this project relies on. (DynamicPriorityQ relies on StaticList, for example). This helps Eclipse determine where to look for data type definitions and header files. Eclipse is not perfect. Sometimes projects get out of sync and you need to: clean and build each project, right click a project, choose Index, and Rebuild.

Another way to set the references is: **Right click on Project > Properties > C/C++ General > Paths & Symbols > References**[2]

1 http://crawlicious.com/wp/2009/06/11/make-change-dir/

2 http://stackoverflow.com/questions/1270799/eclipse-cdt-make-a-project-rebuild-when-a-library-built-in-another-project-was-r