C Priority Queue

Due Sunday November 15, 2015

Objectives

For this assignment, you will:

- Gain experience programming in C
- Gain experience working with pointers and structs
- Gain experience using a header file

Description

In this assignment, you will create a priority queue. You should construct a correct, robust, and elegant data structure. You may want to write a small program to test it, but the product will be just the header file and the implementation file for `pqueue` named, naturally enough, `pqueue.h` and `pqueue.c`. This will also give you some practice in reading and using documentation.

We want to build a priority queue that stores pairs of integers. Each pair consists of a key and a priority. The priority queue will allow the user to insert pairs, to remove the pair with the lowest priority. A `struct` has already been defined for you to store the pair of integers in the priority queue. Also, you must check all preconditions and handle them as specified in the documentation provided in `pqueue.h`.

Classes

The `pqueue.h` header file

You will code your `pqueue` based on an implementation that uses a heap-ordered array like that in the class `VectorHeap` from Bailey's structure library (see the source code on the handouts page). In your implementation, **all of the functions must work correctly in logarithmic time**. In other words, doing a linear search for a key across a array is not permitted.

It is typical to create an identically named .c file (in this case, `pqueue.c`) that contains the *implementation* for each of the functions prototypes declared in the header file (in this case, `pqueue.h`). The functions declared in the header file are as follows:

```
pqueue_init(pqueue *self)

pqueue* pqueue_push(pqueue *self, int key, int priority)

pqueue* pqueue_pop(pqueue *self)

pqueue* pqueue_top(pqueue *self, int *key, int *priority)

unsigned int pqueue_size(pqueue *self)

bool pqueue_empty(pqueue *self)
```

Full descriptions of these methods can be found in `pqueue.h`, and you will need to implement them to the specifications. You should compare the design of this priority queue with the design of the stack example from class.

Getting Started

1. Before you begin make sure you understand how a heap based priority queue works by carefully reading the implementation in Bailey's library.
2. Read over the stack implementation example from class to get a feel for how we work "objects" in C.
3. When you are ready create a new C file called `pqueue.c` in the src directory. This is where you will implement all of the functions. You may (and should) add additional helper functions to `pqueue.c`.

Grading

criterion	points
Functionality of priority queue	10
Properly checking preconditions	4
Appropriate comments	2
Style and formatting	2
General correctness	2
submitted correctly	1

Submitting Your Work

Submit the files `pqueue.h`, `pqueue.c` and the `main.c` file you used for testing. Submit this assignment in the same way you submitted Laboratory 10. This will be the approach we will take for all of our C submissions.

Some hints

- Try and follow the outline from our existing binary heap.
- As always, try and code incrementally, testing and compiling as you go.
- Use `fprintf` to help you debug (but remove these when you submit your final version).
- Leave plenty of time for debugging!