**Project: Priority Queue for Emergency Room Admittance Based on Heap Array (Topics Discussed in Week 8)**

Name: Theo Shin

Class and Semester: CSPB2270, SUMMER 2019

As we’ve all learned, standard queues typically process elements in a first-in, first-out manner (“FIFO”) which can be a useful standard for many situations, but not every situation. Take for instance, a medical emergency room (ER) where patient admittance is based on a priority schedule. As Frankfort Regional Medical Center describes, “The emergency room (ER) staff uses a triage system to determine who to see first. ER staff sees the sickest people first. For example, if someone comes in with a heart attack, and someone comes in with a cough, obviously the person with the heart attack is probably going to take precedence. In the triage system, everyone is given a number, one through five. Someone who is essentially actively dying is going to be a one, and somebody coming in for a scratch, or because a toe hurts, might be a five. Depending on peoples' acuity level, they'll be assigned one of those triage numbers. ER staff tries to see everyone as fast as they can, but the sicker people are going to be seen first.”

Since standard queues aren’t particularly useful when priority is involved, a standard queue variation called a priority queue would serve as the optimal approach. This way, patients with critical issues can be dealt with promptly and pre-empt others who may even been waiting longer. In a priority queue, elements are prioritized relative to one another and when prompted to dequeue, the highest priority element (in this case, the patient with highest injury severity) is removed and checked. The priority queue is categorized as an abstract data type where elements with the maximum ASCII value takes precedence, as opposed to order of insertion.

My proposed priority queue will store a heap struct to maintain patient records including name and urgency score to dictate proper priority. Smaller integers can be considered higher priority and are dequeued ahead of larger values. Along with implementing a priority queue, my project will implement a user interface with menu options to demonstrate hospital entries where new patients can be added, patient information stored, and functions to locate patient information, all in the scope of treatment based order of arrival/urgency of treatment (priority).

Priority queues utilize operations to check whether the queue is empty, insertion of elements with associated priorities, remove elements from queue with highest priority and return it (in class, this was labeled as “pop”. Additionally, a peek() function nearly always executes in O(1) runtime which may prove crucial for this particular application. Concepts were covered in Zybooks chapter 13 and tested on questions 13 and 14 of midterm 2 to ensure we understood the methods for swapping and sequencing of queue elements.

My particular reasoning for implementing a heap priority queue was based on runtime complexity. Heaps perform faster across the board compared to the other viable choices of arrays or linked lists for this particular implementation. Hospital care relies on efficiency where time can be used to save more lives where this heap based priority queue will hopefully contribute towards.

The project will include a PatientQueue class with possibly 6 or more member functions and 3 private functions.

**Data Structs:**

* Patient struct
  + This struct would store patient information namely -- name and urgency of care. Could contain next patient instructions, priority labels, and name strings.
  + Consists of string name, int patient urgency score, and constructs patient info using pointer variables (parent, left, right)

**Publics Member Functions:**

* void insert (string name, int urgency);
  + For creating new patient info and adding to queue using specified priority (urgency 1-10).
  + Will consist of while loops, for loops, and pointers to compare indices (similar to prior homework assignments).
* shared\_ptr<patient> patientfind(string name);
  + Searches through heap for “name” pointer initialized by new pq node
* shared\_ptr<patient> pop();
  + Returns root of heap with priority info on patient with highest urgency score. Otherwise will return NULL.
  + May call delete function; Tests patient index against values using loop
* bool is\_empty();
  + Checks if queue is empty using if function
* void remove (string name);
  + Calls delete function to remove patient name by finding pointer and index
* void dequeue (patient \* next, int p\_index);
  + Removes patient and heap restructure using implement of pointers
* void print (string name);
  + Finds and prints name of patient in particular queue using particular if/else statements
* string peek (patient \*, &queue)
  + Accesses patient with highest urgency score and returns without modifying queue
  + Return empty string (“ “) if queue is empty using if/while statements and testing against NULL

**Private Member Functions:**

* void swap(int index\_a, int index\_b);
  + swap array contents
  + for loop to check index value and swap values based on result
* int size();
  + determines size based on index value
* void patient\_array(shared\_ptr<patient>)
  + initialize array that holds heap structure similar to homework initializing functions

**Testing Criteria:**

* Run through menu options for entering new patient, signaling patient with immediate urgency needs, along with other options for removal and editing. Since a heap structure will be used, runtime efficiency is optimal. Testing will include ensuring patient with highest urgency score is popped from queue and patient information is reported to user.
* Could possibly implement unit-test functions replicating similarly to homework assignments. Test could to see if proper patient information was popped based on queue order. Can also remove or delete patient info to see if heap structure maintains. Test to see if empty array.

References:

[11] <https://en.wikipedia.org/wiki/Priority_queue>

[2] Zybook Chapter 13

[3] <https://www.geeksforgeeks.org/priority-queue-set-1-introduction/> (Functional understanding of priority queue representation in C++ with functions for empty(), size(), pop(), swap(), etc)

[4] <https://brilliant.org/wiki/priority-queues/> Priority queues as it relates to hospitals