CSCI-2270
Assignment 8
Instructor: Boese

Priority Queue
Emergency Room Scheduling

Objectives

- Implement a Priority Queue using a Heap with an array as the underlying data structure
- Add element to Heap
- Remove element from Heap

Background

Hospital Emergency Room Waiting List
A hospital emergency room does not see patients on a first-come first-served (FIFO) basis. Rather, the most urgent patients are seen before patients whose injuries are less severe or non-life-threatening. You will implement a priority queue such that patients with the most severe injuries are treated before those who can wait. You will also track the total time that the ER doctors have spent treating patients. For each patient in the priority queue, the following information is stored:

- **Name**
- **Injury Severity** (an integer, higher values represent higher priorities for treatment)
- **Treatment Time** (time in minutes required to provide treatment)

Program Specifications

- Examples of the output formatting are shown in a later section of this document.
- Your priority queue will be implemented in a class based on the provided PriorityQueue.hpp.
- You will also implement a driver program with menu functionality.
- Your program will read in the maximum queue size from the first command line argument.
- When attempting to add a new patient, if the priority queue is full, those patient(s) will be sent to another hospital.
- Your program will give users the option to read patient information from a text file. This text file could represent a list of those injured in a major incident, such as a fire or other calamity, and are being transported by ambulance. The text file will have a header line, followed by patient information. Patient information will be stored one patient per line. There is no maximum number of patients that may be stored in the text file. Note that patient names will not have spaces. Information on each line will consist of name, followed by injury severity, followed by treatment time in minutes.

Example Text File: busCrashInjured.txt

```
<Name> <Injury Severity> <Treatment Time>
Bob 10 45
Charles 5 25
Adam 7 35
```

- **Menu:** In your driver function, display a menu with the following options: *Note that output formats and examples are shown later in this document.*

  1. Get Patient Information from File
     - When this option is selected, the user will be prompted for the name of a text file containing patient information. *Note: This file should be located in the current directory.*
     - Patients whose information is stored in the text file will be enqueued in the priority queue. The file will be processed line-by-line *(do not search the file for patients with the highest injury severity, simply process patients in order as they are read from the text file).*
     - If the priority queue is empty when this option is chosen, patients in the file are added to the queue.
     - If the priority queue is not empty when this option is chosen, patients in the file are added to the pre-existing priority queue.
     - If the queue reaches maximum capacity during this operation, remaining patients will be sent to another hospital.
     - If two patients have an identical injury severity, give priority to the patient with shorter treatment time. *We assume that two patients cannot have both the same injury severity and treatment time*.

  2. Add Patient to Priority Queue

     If the priority queue is not full:
     - Prompt the user for the name of the patient, patient's injury severity, and expected treatment time.
     - This patient is then inserted into the priority queue based on injury severity.
       - If two patients have an identical injury severity, give priority to the patient with shorter treatment time. *We assume that two patients cannot have both the same injury severity and treatment time.*

     If the priority queue is full:

     The patient will be sent to another hospital.

  3. Show Next Patient

     If the priority queue is not empty, displays information for the patient at the head of the priority queue, but does not dequeue the patient.
     If the priority queue is empty, notify the user of this fact.

4. Treat Next Patient

   Provides treatment for the patient at the head of the priority queue and removes patient from the priority queue.

   - If the priority queue is not empty, display patient's name and total time spent by the ER doctors treating patients. This total time includes the time spent treating the patient just removed from the priority queue.
   - If the queue is empty, notify the user of this fact.

5. Treat Entire Queue

   When the user selects this option, patients will be treated until the priority queue is empty.

   - As patients are removed from the priority queue and provided treatment, display their name, as well as the total amount of time that the ER doctors have spent treating patients.
   - If the queue is empty, notify the user of this fact.

6. Quit

   When the user selects this option, the program exits cleanly.

- **Output:** All output will be generated in the **main** function. See “Expected Output” section.

- **Class Implementation**:
  - Your class will store the priority queue in an array.
  - *See lecture notes for information on how to implement a binary heap in an array for implementation tips.*
  - Store the node with highest priority at index 1 of the array, not index 0.
  - Since the priority queue is an array of type PatientNode, not an array of pointers to PatientNodes, you do not need to create new nodes with the *new* keyword.

- Implement all methods in the PriorityQueue class header.

- **Expected Output**

  Used for formatting purposes only.

Display menu

```
cout << "======Main Menu======" << endl;
cout << "1. Get Patient Information from File" << endl;
cout << "2. Add Patient to Priority Queue" << endl;
cout << "3. Show Next Patient" << endl;
cout << "4. Treat Next Patient" << endl;
cout << "5. Treat Entire Queue" << endl;
cout << "6. Quit" << endl;
```

Get Patient Information from File

```
cout << "Enter filename:" << endl;
//if priority queue is full, or becomes full during the process of
//enqueueing all patients in the file, print once regardless of how
//many patients are turned away
cout << "Priority Queue full. Send remaining patients to another hospital." << endl;
```

Add Patient to Priority Queue

```
//if priority queue is not full
cout << "Enter Patient Name:" << endl;
cout << "Enter Injury Severity:" << endl;
cout << "Enter Treatment Time:" << endl;
//if priority queue is full
cout << "Priority Queue full. Send Patient to another hospital." << endl;
```

Show Next Patient

```
//if the priority queue is not empty
cout << "Patient Name: " << patientName << endl;
cout << "Injury Severity: " << patientInjurySeverity << endl;
cout << "Treatment Time: " << patientTreatmentTime << endl;
//if priority queue is empty
cout << "Queue empty." << endl;
```

Treat Next Patient

```
//if priority queue is not empty
cout<<"Patient Name: "<< patientName
      << " - Total Time Treating    Patients: "
      << totalTreatmentTime <<endl;
//if priority queue is empty
cout << "Queue empty." << endl;
```

Process Entire Queue

```
//if priority queue is not empty:
cout<<"Patient Name: "<< patientName
      << " - Total Time Treating    Patients: "
      << totalTreatmentTime << endl;
//if priority queue is empty
cout << "Queue empty." << endl;
```

Quit

```
cout << "Goodbye!" << endl;
```

- **Sample Program Execution**:
  Some of the following program executions read info from *busCrashInjured.txt*

busCrashInjured.txt

```
<Name> <Injury Severity> <Treatment Time>
Bob 10 45
Charles 5 25
Adam 7 35
```

  - $./a.out 5 - executes program with priority queue size 5, add two patients, and treat patients one at a time

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
2
Enter Patient Name:
Smith
Enter Injury Severity:
2
Enter Treatment Time:
15
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
2
Enter Patient Name:
Jones
Enter Injury Severity:
8
Enter Treatment Time:
45
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
Patient Name: Jones - Total Time Treating Patients: 45
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
```

```
Patient Name: Smith - Total Time Treating Patients: 60
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
6
Goodbye!
```

- $./a.out 10 - executes program with priority queue size 10, read patient info from busCrashInjured.txt, and treat entire queue

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
1
Enter filename:
busCrashInjured.txt
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
5
Patient Name: Bob - Total Time Treating Patients: 45
Patient Name: Adam - Total Time Treating Patients: 80
Patient Name: Charles - Total Time Treating Patients: 105
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
6
Goodbye!
```

- ./a.out 3 - execute program with priority queue size 3, read patient info from busCrashInjured.txt, attempt to add one more patient, then treat entire queue. This will attempt to enqueue more patients than the priority queue size allows.

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
1
Enter filename:
busCrashInjured.txt
```

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
2
Priority Queue full. Send Patient to another hospital.
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
5
Patient Name: Bob - Total Time Treating Patients: 45
Patient Name: Adam - Total Time Treating Patients: 80
Patient Name: Charles - Total Time Treating Patients: 105
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
6
Goodbye!
```

- ./a.out 3 - execute program with priority queue size 3, add two patients, read patient info from busCrashInjured.txt, then treat entire queue. This will attempt to add two patients while the queue is full during the *Get Patient Information from File* execution, so the last two patients from the file are not added to the queue. Two patients have the same injury severity, so priority is given to the patient with shortest treatment time.

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
2
Enter Patient Name:
Wellington
Enter Injury Severity:
10
Enter Treatment Time:
120
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
2
Enter Patient Name:
```

```
Taylor
Enter Injury Severity:
3
Enter Treatment Time:
33
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
1
Enter filename:
BusCrashInjured.txt
Priority Queue full. Send remaining patients to another hospital.
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
5
Patient Name: Bob - Total Time Treating Patients: 45
Patient Name: Wellington - Total Time Treating Patients: 165
Patient Name: Taylor - Total Time Treating Patients: 198
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
6
Goodbye!
```

- ./a.out 5 - executes program with priority queue size 5, read patient info from busCrashInjured.txt, and *Show Next Patient* / *Treat Next Patient* until queue is empty.

```
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
1
Enter filename:
busCrashInjured.txt
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
3
Patient Name: Bob
Injury Severity: 10
Treatment Time: 45
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
Patient Name: Bob - Total Time Treating Patients: 45
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
3
Patient Name: Adam
Injury Severity: 7
Treatment Time: 35
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
Patient Name: Adam - Total Time Treating Patients: 80
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
3
Patient Name: Charles
Injury Severity: 5
Treatment Time: 25
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
Patient Name: Charles - Total Time Treating Patients: 105
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
3
Queue empty.
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
4
```

```
Queue empty.
======Main Menu======
1. Get Patient Information from File
2. Add Patient to Priority Queue
3. Show Next Patient
4. Treat Next Patient
5. Treat Entire Queue
6. Quit
6
Goodbye!
```

PriorityQueue.hpp Header File:

```cpp
#ifndef PRIORITYQUEUE_HPP
#define PRIORITYQUEUE_HPP

#include <string>

// a struct to store patient information
struct PatientNode
{
  std::string name;
  int injurySeverity;
  int treatmentTime;
};

class PriorityQueue
{
public:
/*
 class constructor
 Purpose: perform all operations necessary to instantiate a class
                  object
 Parameter _queueSize: Maximum size of the priority queue
 return: none
*/
PriorityQueue(int _queueSize);

/*
 class destructor
 Purpose: free all resources that the object has acquired
 Parameters: none
 return: none
*/
~PriorityQueue();

/*
 Method Name: enqueue
 Purpose: enqueue new patient into priority queue; call other
                   method(s) as required to maintain heap
 Parameter: _name - name of patient to be entered in priority queue
 Parameter: _injurySeverity - severity of injury
 Parameter: _treatmentTime - time required for doctor to treat patient
                  injury
 returns void
*/
```

```
void enqueue (std::string _name, int _injurySeverity, int _treatmentTime);

/*
 Method Name: dequeue
 Purpose: remove the patient at the front of the priority queue from
                  the queue, call other method(s) as required to maintain heap
 Parameters: none
 return: void
*/
void dequeue();

/*
 Method Name: peekName
 Purpose: get name of patient at front of priority queue while
                  maintaining encapsulation
 Parameters: none
 return: name of patient at the front of the priority queue
*/
std::string peekName();

/*
 Method Name: peekInjurySeverity
 Purpose: get injury severity of patient at front of priority queue
                  while maintaining encapsulation
 Parameters: none
 return: injury severity of patient at the front of the priority queue
*/
int peekInjurySeverity();

/*
 Method Name: peekTreatmentTime
 Purpose: get treatment time of patient at front of priority queue
                   while maintaining encapsulation
 Parameters: none
 return: treatment time of patient at the front of the priority queue
*/
int peekTreatmentTime();

/*
 Method Name: isFull
 Purpose: indicate if priority queue is full
 Parameters: none
 return: true if queue is full, false otherwise
*/
bool isFull();

/*
 Method Name: isEmpty
 Purpose: indicate if priority queue is empty
 Parameters: none
 return: true if queue is empty, false otherwise
*/
bool isEmpty();
```

```
private:
/*
 Method Name: repairUpward
 Purpose: maintain heap properties by swapping node with parent if necessary
 Parameter nodeIndex - index of node that may violate heap properties
 return: void
*/
void repairUpward(int nodeIndex);

/*
 Method Name: repairDownward
 Purpose: maintain heap properties by swapping node with child if necessary
 Parameter: nodeIndex - index of node that may violate heap properties
 return: void
*/
void repairDownward(int nodeIndex);

PatientNode* priorityQueue;      //pointer to the array used to implement priority queue
int currentQueueSize;            //number of patients currently in priority queue
int maxQueueSize;     //maximum capacity of priority queue

};
#endif
```

Submitting Your Code:
Log into Moodle and go to the Homework 8 link. It is set up in quiz format. Follow the instructions on each question to submit all or parts of your code. You can check your solution to each question by clicking on the "Check" button multiple times without penalty. If you wrote helper functions, you may use them in the same answer boxes as your method implementations. Simply copy the helper function definition before the definition of the method.

Note that you can only submit your homework once.

Note: there is no late period on assignments! If you miss the deadline or do not do well, you can sign up for an optional grading interview to get up to half the points missed back. There is also an optional extra credit assignment at the end of the semester you can use to replace one of your homework scores.