This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

Semester Project Part 4: Managing an ER using a Priority Queue

Data Structures and Analysis of Algorithms, akk5

Objectives

- To strengthen student's knowledge of C++ programming
- To give student experience reading and parsing strings of commands
- To give student experience in writing a non-linear data structure
- To give the student experience implementing a priority queue using a Heap

Background

Queuing is a problem which is commonly encountered in stores, movie theaters, IT services, and hospitals; many people or problems waiting in line to be addressed in an orderly and controlled fashion. Basic queues address the problem based solely on the relative positions of the object in the queue; the first object in the queue is the first object out of the queue. Basic queues are well suited to model simple lines like the ones found in stores, the DMV, and movie theaters. When other parameters such as degree of urgency impact the order in which the objects in the queue must be addressed, priority queuing is the answer. Priority queues pull objects from the queue based on measure of precedence known as priority. IT services and emergency rooms in hospital rooms often employ forms of priority queuing.

Heaps are commonly used to implement a priority queues. The reason is because all elements in the heap are organized relative to a given heap order property. This property requires that items retrieved from the heap are either the smallest value in the heap (called a min heap), or the largest value in the heap (called a max heap). In other words, when retrieving data from a min heap, the element with the smallest value over all elements in the heap has priority and is removed first; the same holds for a max heap. The heap data structure is a complete binary tree, implemented as a static or dynamic array, where each subtree in the tree also follow the heap order property (i.e. the root is either the minimal or maximal value in that subtree).

This exercise will focus on the potential use of priority queuing in emergency rooms. Hospital emergency rooms often divide the severity of their cases into five general levels from most urgent to least urgent: Code, Critical, Urgent, Non-urgent disabled, Ambulatory.

The Code level "refers to someone who has suffered cardiac arrest outside of the hospital or someone whose vital signs crash within the emergency department [1]." It also refers to " people with gunshot or stab wounds with possible vital organ involvement and/or altered or absent vital signs [1]." Patients at the Code level bypass the management system and are taken directly to the next available trauma bed.

The Critical level refers to "a person with stable vital signs who is exhibiting symptoms or who gives a history that clearly delineates a life-threatening condition. This might be a patient with chest pain, shortness of breath, and profuse sweating (diaphoresis). This also would include people who have a history of vomiting blood, multiple traumas with head injury, or a gunshot or stab wound, as well as asthmatics, diabetics with low blood sugar or extremely high blood sugar, and the like. [1]"

The Urgent level refers to "patients with serious conditions requiring medical intervention within two hours … These are people with abdominal pain, high fever and/or productive cough, deep lacerations with bleeding under control, closed fractures with deformity, and so on [1]."

The Non-urgent disabled level refers to "individuals, unable to walk or remain in a chair, … for whom the triage nurse determines that up to a four-hour wait is clinically acceptable [1]." These include patients with herniated discs as well as nursing home patients with dislodged catheters and feeding tubes.

The Ambulatory level refers to patients "who do not need emergency care but are there anyway with colds, toothaches, headaches, bumps, bruises, abrasions, small lacerations, skin rashes, and so on [1]."

It is not uncommon for multiple patients to have complaints at the same level of severity. When this happens, priority is given to the patient that arrived in the emergency room first.

Instructions

Using heaps and priority queuing, design and implement an emergency room simulator. The purpose of the simulator is to model the daily flow of patients through an emergency room. The system must be able to store the patient's name and complaint and then be able to prioritize their access based on the severity of their complaint and the order they entered the ER.

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

Your program should implement a command line (text-based interface) capable of handling the following commands:

exit – exits the program

load *<file>* - parses the contents of the file as if they were entered from the command line

display – displays the patient record for the patient at the top of the priority queue.

next – removes the current patient and moves the next patient in the priority queue to the top.

admit *<last name> <first name>* - beings the admission process for patient <first name> <last name>. This should create default patient record with no description or symptoms listed, a triage level of ambulatory, and the current ticket number. Upon execution, admit should display the following:
Creating a patient record for <last name>, <first name>
Priority: ambulatory
Ticket: <ticket no>
Please add a description of the complaint and any reported symptoms

set complaint *<complaint>* - Adds a description of the complaint to the patient record.

add symptom *<symptom>* - Adds a symptom to the open patient record.

set priority *<priority>* - Sets the priority level to ambulatory, non-urgent, urgent, critical, or code. Please use **ambulatory**, **non-urgent**, **urgent**, **critical**, and **code** as possible values of ***<priority>***.

timda - Closes the current patient record and adds it to the priority queue

Guidance

The parsing for this program is slightly different than previous programs; the command set is modal. The load, display, next, and admit commands only work if an admit command isn't be processed; set, add, and timda only work while processing the admit command. An easy strategy to use for handling this problem is to implement two different parsing functions, one which is called while process an admit request and one to handle the rest of the process.

Pay close attention to your Patient class, you can greatly simplify your heaps code if you overload the < and << operators. As always, these overloads aren't strictly necessary, but they do make the code you write cleaner and easier to read.

Each patient can have a variable number of symptoms, so make sure to use an appropriate structure to represent that data. It is also necessary for your program to assign ticket numbers to each patient as they are admitted; this number is essential for handling ties in priority levels. Make certain your program handles the ticket numbers automatically.

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

This program asks you to display the current patient at the head of the queue. You can accomplish this by implementing the equivalent of a peek operation for your heap. This will allow you to display the top element in the priority queue without needing to remove it.

Grading Breakdown

Point Breakdown	
Structure	12 pts
The program has a header comment with the required information.	3 pts
The overall readability of the program.	3 pts
Program uses separate files for main and class definitions	3 pts
Program includes meaningful comments	3 pts
Syntax	28 pts
Implements Class Patient correctly	9 pts
Implements Class Heap correctly	19 pts
Behavior	60 pts
Program handles all command inputs properly	
• Exit the program	5 pts
• Create a default patient record	7 pts
• Set the description for a patient record	7 pts
• Add symptoms to a patient record	7 pts
• Set the triage level for the patient	7 pts
• Admit the patient	7 pts
• Display the record of the current patient	7 pts
• Remove the current patient from the queue	7 pts
• Load a file	6 pts
Total Possible Points	**100pts**
Penalties	
Program does NOT compile	-100
Late up to 24 hrs	-30
Late more than 24hrs	-100

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

Header Comment

At the top of each program, type in the following comment:

/*

Student Name: <student name>

Student NetID: <student NetID>

Compiler Used: <Visual Studio, GCC, etc.>

Program Description:

<Write a short description of the program.>

*/

Example:

/*

Student Name: John Smith

Student NetID: jjjs123

Compiler Used: Eclipse using MinGW

Program Description:

This program prints lots and lots of strings!!

*/

Assignment Information

Due Date: 3/22/2020

Files Expected:

1. Main.cpp – File containing function main
2. heap.h - File containing the Heap and Patient class definitions.
3. heap.cpp - File containing the code for the Heap and Patient classes' methods.

References

[1] T. A. Sharon, "Standards of Care in the Emergency Room: The Emergency Waiting Game," LinkedIn. [Online]. Available: https://www.linkedin.com/pulse/standards-care-emergency-room-waiting-game-sharon-r-n-m-p-h-. [Accessed: 15-Oct-2018].

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.