COMP 2611 – Data Structures 2018/2019

Semester 1

Assignment #3 Date Due: 7th November, 2018 @ 11:55p.m

Description

This assignment requires you to write an application for handling patients who come in for treatment at the Emergency Department of a certain hospital. The application uses a simple database to store information on patients. When a patient comes in for treatment, if he/she is in the patient database, a priority is assigned to the patient and the patient's information is inserted in a max priority queue. The priority assigned depends on the nature of emergency. If the patient is not in the patient database, his/her information is first stored in the patient database and then a priority is assigned to the patient and his/her information is inserted in the max priority queue.

Patients are treated in order of priority. The next patient to be treated is the one with the highest priority in the max priority queue. When a patient is treated, his/her information is removed from the priority queue and inserted in another queue which keeps track of the patients who have already been treated.

The patient database must be implemented using a binary search tree. The max priority queue must be implemented using a max heap. The queue of patients already treated can be implemented using a regular first-in first-out queue.

The application must provide a menu to facilitate the entering of new patient information in the patient database and to handle patients as they come in for treatment. The menu must also allow the user to view the information stored in the different data structures.

You are supplied with a Dev C++ project containing various `.h` and `.cpp` files. You do not have to modify the project. The code for this assignment must be written in the `Heap.cpp`, `MaxPriorityQueue.cpp`, and `Main.cpp` files. There is no need to modify the code in the other files supplied.

Structure Definitions

The file, `DataTypes.h`, defines the data types to be used in Assignment #3. These are listed in Table 1:

Data Type	**Description**
Patient	A struct containing patient information which is stored in the patient database (binary search tree).
Incident	A struct containing information about a patient to be treated. The max priority queue and the queue of patients already treated contain data of type *Incident*.
BTNode	A node in the binary search tree (which stores data of type *Patient*).
QueueNode	A node in the queue of patients already treated.

Table 1: Data Types

Heap Functions (to be written in MaxHeap.cpp)

Return type	Function and Description
`int`	`parent (int i):` Returns the index of the parent node of node *i* in the heap.
`int`	`leftChild (int i):` Returns the index of the left child of node *i* in the heap.
`int`	`rightChild (int i):` Returns the index of the right child of node *i* in the heap.
`void`	`maxHeapify (Incident A[], int heapSize, int i):` Assuming that the left and right subtrees of node *i* are max heaps, maintains the max heap property starting at node *i*.
`void`	`buildMaxHeap (Incident A[], int lengthA):` Given an array *A* of type *Incident* where *lengthA* is the amount of elements in *A*, converts *A* into a max heap based on the priority of each patient.
`void`	`displayHeap (Incident A[], int heapSize):` Given an array *A* of type *Incident* which implements a max heap, and *heapSize*, which is the amount of elements in the heap, displays the elements in the heap using a level order traversal.

Priority Queue Functions (to be written in PriorityQueue.cpp)

Return type	Function and Description
`void`	`heapInsert (Incident A[], int heapSize, Incident newIncident):` Inserts *newIncident* in the priority queue stored in array *A* based on the priority of the Incident.
`Incident`	`heapMaximum (Incident A[][], int heapSize):` Returns the highest priority element in the max priority queue stored in array *A*. If there is none, the patient ID of the returned Incident struct should be -1.
`Incident`	`heapExtractMax (Incident A[], int heapSize):` Removes and returns the highest priority element in the max priority queue stored in array *A*. If there is none, the patient ID of the returned Incident struct should be -1.

`void`	`heapIncreasePriority:`
	`(Incident A[], int heapSize, int i, int newPriority);`
	Increases the value of element *i's* priority to the new value, *newPriority*, where *newPriority* ≥ element *i's* current priority.

`bool`	`isEmptyHeap(Incident A[], int heapSize):` Returns *true* if the priority queue stored in array *A* is empty.
`int`	`indexOfHeap(Incident A[], int heapSize, int patientID):` Returns the index of *A* where there is an Incident for the patient with the given ID. If there is no such Incident in the priority queue, returns – 1.

Application (To be written in Main.cpp)

You are required to write an application which, on startup, reads Patient information from a file, `Patient.txt`, and stores the information in a binary search tree. Each line of data in the `Patient.txt` file contains the following information:

```
      Patient Id         Patient Name (no spaces)        Patient Telephone Contact
```

Next, the application reads Incident information from the file, `Incident.txt`, and stores each Incident in a priority queue based on the priority of the Incident. Each line of data in the `Incident.txt` file contains the following information:

```
      Patient Id         Condition (no spaces)           Priority
```

The application then provides a menu from which various operations are performed. The operations are performed by calling one or more of the functions written in the previous sections. The following is the menu that must be displayed:

```
   Patient Management System
   ---------------------------------------------------------
   1. Add New Patient to Patient Database
   2. Add Patient to Priority Queue
   3. Increase Patient Priority
   4. Display Information on Next Patient to be Treated
   5. Treat Highest Priority Patient
   6. Display List of Patients Currently Awaiting Treatment
   7. Display List of Patients Already Treated
   8. Display List of Patients in Patient Database
   9. Quit

   Please enter an option:
```

When an option is selected, the appropriate action must be taken, after which the menu is re-displayed. If the user types '0' after selecting Options 1-3, control returns to the menu. The following is a description of each option.

Option 1:

When this option is selected, the program should allow the user to enter the patient ID, name, and telephone contact for a new patient at the keyboard. The data must be read and stored in the binary search tree containing patient information.

Option 2:

This option is selected when a patient comes to the Emergency Department for treatment. The program should allow the user to enter the patientID, condition, and priority of the patient. A check is first made to determine if the patient in stored in the patient database. If so, the patient information is stored in the priority queue based on his/her assigned priority. If not, the user is requested to enter the new patient using Option 1.

Option 3:

This option allows the priority of given patient on the priority queue to be increased. A check must first be made to determine if the patient is currently on the priority queue. The user would be prompted to enter the `Patient Id`.

Option 4:

When this option is selected, the program should display information on the highest priority patient on the queue (if there is one).

Option 5:

When this option is selected, the program should remove the highest priority patient from the queue (if there is one). At the same time, the patient is added to a regular queue containing a list of patients already treated.

Option 6:

When this option is selected, the program should display information on all the patients currently waiting to be treated and their priorities, in order of priority.

Option 7:

When this option is selected, the program should display information on all the patients who have already been treated, in the order in which they were treated.

Option 8:

When this option is selected, the program should display information on all the patients stored in the patient database, in order of patient ID.

Submission Details:

1. *Complete* and <u>*sign*</u> the plagiarism declaration form. Please note marks for the assignment will not be released unless this form is submitted. The entire plagiarism form must be completed.
2. Submit the zipped Assignment3 project folder you worked on containing the project file as well as the .h and .cpp files and the completed plagiarism form.