The Assignment

You will implement and test a PriorityQueue class, where the items of the priority queue are stored on a linked list. The Node struct for this linked list must be defined in your header file for the priority queue rather than in a separate header file. All of the PriorityQueue member functions should be written by you from scratch (not using the linked list toolkit).

Note: This is not a template class; you will use a typedef statement to define the underlying data type.

Purposes

Give you more practice in building and manipulating linked lists.

Introduce you to priority queues which we will use again later in the semester.

Before Starting

Read all of Chapter 8.

Files that you must write

1. pqueue.h
2. Download pqueue.h
3. : The header file for this first version of the PriorityQueue class. Actually, you don't have to write much of this file. Just copy our version from pqueue1.h and add your name and other information at the top. If some of your member functions are implemented as inline functions, then you may put those implementations in this file too.
4. pqueue1.cxx: The implementation file for the PriorityQueue class. You will write all of this file, which will have the implementations of all the PriorityQueue's member functions. Also, remember that the PriorityQueue's linked list consists of dynamic memory, so you will need to define a copy constructor, an assignment operator, and a destructor.

Other files that you may find helpful

1. pqtest.cxx
2. Download pqtest.cxx
3. : A simple interactive test program.
4. pqexam1.cxx

5. Download pqexam1.cxx
6. : A non-interactive test program that will be used to grade the correctness of your PriorityQueue class.

The PriorityQueue Class Implemented with a Linked List
Discussion of the Assignment

You will store the items on a linked list, where each node contains both an item and the item's priority, as shown here:

```
struct Node {
   PriorityQueue::Item data;
   unsigned int priority;
   Node *link;
};
```

This Node definition appears in the header file pqueue1.h, immediately after the PriorityQueue class definition. Neither class needs to be a template class (instead, the Item type is defined as an int by using a typedef within the PriorityQueue class definition).

This approach is different from the approaches discussed in Section 8.4 since all the items are stored on a single linked list. I suggest that you maintain the linked list in order from highest to lowest priority. If there are several items with equal priority, then the one that came in first will be in front of the others. If you choose to follow this suggestion, then make sure that you have a clearly stated invariant that expresses this way of storing the items.

Because the linked list is kept in order, the get_front operation is simple. It merely gets the head item of the linked list. On the other hand, the insert operation requires some amount of work, making sure that the new entry goes at the correct spot in the linked list. A new entry must go after any existing entries with a higher or equal priority.

Extra Credit

Covert the Node struct to a class template. Provide the minimum functions / operators needed. (Constructors and destructor, as well as operator overloads for assignment and

member wise equality). This will also require you change the pqexam1.cxx file. I added the following near the top:

```
typedef int TestDataType;
```

That way when I used the node template it would be cleaner. I addition to the typedef, there are 16 places in the example file you need to change. For example:

```
PriorityQueue<TestDataType> test;
```