Students:

This content is controlled by your instructor, and is not zyBooks content. Direct questions or concerns about this content to your instructor. If you have any technical issues with the zyLab submission system, use the **Trouble with lab** button at the bottom of the lab.

5.15 Lab 5 - Priority Queue

Due Friday August 20, 2021 at 11:59 PM

Collaboration Policy Reminder

You may not use code from any source (another student, a book, online, etc.) within your solution to this PROGRAM. In fact, you may not even look at another student's solution or partial solution to this PROGRAM to use as a guide or otherwise. You also may not allow another student to look at any part of your solution to this assignment. You should get help on this assignment by coming to the instructors' office hours or by posting questions on Slack. Do not post your code on Slack.

Problem Definition

You will implement a binary min-heap based priority queue class template. You will demonstrate the modularity of your design by adapting your implementation of a vector-based binary heap that has its root at the zero index of the underlying vector into a binary heap whose root is positioned at index 1. Next, you will augment your priority queue classes to allow for constructing a priority queue via a build heap operation. You should plan to reuse code whenever possible, and thoroughly test your implementations.

Priority Queue Class

The priority queue class for the zero index-based min-heap should have the following interface. You are allowed to add functions you deem necessary to the heap portion of the class. It is suggested that you first get your binary heap implemented and tested for correct functionality. You can do this by temporarily making the heap portion of your class public so you can test each heap function independent of the overall priority queue class. The priority queue class interface must have solely the public functions listed below. You may find the following expressions useful: parent index = (i-1)/2, left child index = 2i+1, right child index = 2i+2.

pq_zero.h

```
  vector<Item> heap; // the heap expands/shrinks as its data
  typedef int indx;  // index with heap
  map<Item,indx> index;  // records each Item's place in heap
  map<Item,float> priority; // records each Item's priority
  void percolate_up( indx i );
  void percolate_down( indx i );
public:
  // These use the min-heap functions above.
  int size( ) const;
  bool empty( ) const;
  const Item& top( ) const;
  void pop( );
  void push( const Item& w, float prio );
};
```

Priority Queue Push

Your *push* function should allow for simply inserting an Item into the priority queue with a given priority. If that Item is already in the priority queue, then it should have its priority decreased or increased if necessary. In addition, if an **Item** is pushed with the same priority as an Item already in the priority queue, your priority queue implementation should ensure that the Item that was originally in the queue has higher priority. You class should preserve this default first-in-first-out property.

Priority Queue Adaption

Next you will adapt your *pq_zero* class so that the heap has its root element based at index 1 in the underlying vector. Create a new class called *pq_one*. It is important to note that the goal of this portion of the exercise is to exploit code *reuse*. You should already have a modular design that allows for reuse of most of your code from your *pq_zero* class verbatim. If this is not the case, refine your *pq_zero* class now. Again you can include your entire priority queue class in either a single header file named *pq_one.h*, or have a separate class declaration and definition in files named *pq_one.h, pq_one.cc*. You may find the following expressions useful: parent index = i/2, left child index = 2i, right child index = 2i+1.

Build Heap

Next you will overload the priority queue constructor for each of your priority queue classes such that one can construct a priority queue from an initial set of items. This priority queue

LAB ACTIVITY | 5.15.1: Lab 5 - Priority Queue | 0 / 1

Current file: main.cpp

```
1  /**
2   * Course: CS010C Summer 2021
3   *
4   * First Name: Victoria
5   * Last Name: Nguyen
6   * Username: vnguy278
7   * email address: vnguy278@ucr.edu
8   *
9   *
10  * Assignment: Lab 5
11  * Filename : main.cpp
12  *
13  * I hereby certify that the contents of this file represent
14  * my own original individual work. Nowhere herein is there
15  * code from any outside resources such as another individual,
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.cpp (Your program) →

Program output displayed here