COSC 251 – Programming Languages
Project 1
Spring 2019

Objective: Implement a data structure in C++ that is easily dealt with in JAVA.

Your Task: You will create an Object-Oriented version of the priority queue data structure that we will review, briefly, in class (you all should have seen it in your 201 class). Internally, you may code this as whatever data representation you choose, but will have to provide the following functionality:

enqueue, dequeue, and front functionalities.
Ensure that the queue is sorted by priority order (1 is the highest priority).
Clear the queue.
Print out the entire queue in priority order.
Boolean check for empty queue.
Size return.

Restriction: You may not use Vector or List from standard C++.

This functionality will be tested through execution of various methods that are specified below. This project will test your understanding of pointers and pointer chasing. You will build a container type (called Node) that contains a Student object and an integer for the priority level of the object as well as any pointers that you'll need to complete the functionality of the priority queue. Student should have three private members (int id, double gpa, String name) and get/set functions as well as a constructor.

Your PriorityQueue class should hold your actual priority queue made of nodes and contain the functionality noted in the list above. This will be the primary class that our driver will interface with. Also note, since your priority queue is a queue, we will not be allowing for arbitrary index retrievals.

Code Requirements:

Your project files must follow the requirements below. If they do not, it will either not work with our driver, or it will not be structured correctly for a priority queue. You must implement the following methods with the exact method signature noted:

In PriorityQueue.h/PriorityQueue.cpp

PriorityQueue() //constructor call
int size() //returns the size of the queue
void printQueue() //prints every member of the priority queue, in proper order

void enqueue(Student element, int priority) //adds element to the priority queue
Student front() //returns the Student at the front of the queue
void dequeue() //remove the front Student
void clear() //empties the queue
Student topandpop() //removes and returns the front element of the queue
bool isEmpty() //returns true if the queue is empty

In Node.h/Node.cpp

Node(Student s, int priority) //Constructor call
Student getStudent()
int getPriority()
void setPriority(int priority)

In Student.h/Student.cpp

Student(int i, double g, string n) //Constructor call, sets id, gpa, and name
int getID()
double getGPA()
string getName()

You may have more methods and classes than specified, but not less.

Testing:

You should test your code against our driver (posted in the next week) on our servers using g++ to compile your code. If your code does not complete correctly (i.e. the correct, matching outputs) with our driver, then you have code issues that you must fix. If your code works correctly with our driver then you can be reasonably assured that you will do well. Note that errors will need to be caught and dealt with without crashing the driver program. See the driver program for specifics as all errors we will be testing for are tested in the driver.

Deliverables: the class files (.h and .cpp) for Student, Node, and PriorityQueue, plus any other class files you may need. You should have at least 6 files.

Expectations: The code should be clean, concise, well-commented and correct. If you use an outside source, be sure to document that source. Significant use of outside sources will result in a deduction. Grading rubric will be provided a week ahead of the due date. You may work in pairs on this project and pairs across sections are acceptable. If you are going to work in a pair, you must email your professor with your team by 5pm on Tuesday, January 30th. Failure to email by that deadline will mean that you will be working alone for this project.

DUE: February 15th, 11:59pm via Blackboard, question cutoff 5pm, 2/15.

Potential Deductions:
-100 Does not compile with the driver (compilation must be tested on Prometheus)
-50 Runs but crashes
-40 memory leaks (i.e. neglecting to free something you dynamically allocated via new)
-20 lack of comments, including header for every file
-15 queue is not ordered correctly based on priority (priority level 1 is the highest priority and items enqueued at priority level 1 should be dequeued first)
-15 queue has a finite size not based on memory space (i.e. your PriorityQueue can only hold a specific number of elements maximum)
-10 per incorrect test (see driver, each test is a block of code divided by comments)
-10 per missing (but not tested) method. This includes signatures specified in the project document, but not called.

Header comments should include:
Name of file
Your name and names of any partners
Assignment (e.g. COSC 251, Project 1)
Date
Description of the file