UAH Electrical and Computer Engineering Department
Project04 (3 points) Version 1.0

Submit Your Solution to the Canvas Dropbox by
5:00pm on Friday, March 8, 2019

Important Notes

If the preview script tells you that there is a problem with your solution, then you must find and correct the problem.

Start early to give yourself an opportunity to ask questions!!!

Waiting to the last minute to begin the assignment is not recommended and will not get you an extension on the submission deadline.

Debugging issues are far easier to resolve in person when we can see exactly what you and your code are doing.

All CPE 212 projects are *automatically graded* in order to provide you timely feedback. So, it is critical that you follow all directions for the preparation and submission of your projects for grading. Failure to follow the directions may result in zero credit (0 points).

Hint

Match the output of your program to that of the sample solution!!!
Directions for running the sample solution appear below.
Directions for running the preview script appear on the following pages.

Project04 Description

Complete the provided partial C++ program that will implement a ***Priority Queue ADT (abstract data type)*** in which the internal representation of the priority queue is a double-linked series of dynamically allocated nodes.

The ***Priority Queue*** will be used to store text messages read from an input file. Each message has a priority (**H** for **H**igh, **M** for **M**edium, **L** for **L**ow). When input from the file by the **main()** function, the message and priority information are stored within an object of the type **Message**. The text message is stored as a **string** attribute, and the message priority is represented by an attribute with type **Priorities**, an enumerated type provided.

Priority Queue **Rules of Operation:**
(1) If the ***Priority Queue*** contains messages which have different priority levels, then the highest priority messages must be removed from the FRONT of the ***Priority Queue*** before any lower priority messages are removed whenever the **Dequeue()** method is invoked.

> **HINT:** The priority queue is sorted by Priority Level (highest to lowest) as it is being constructed by **Enqueue()** in much the same way as we discussed with the implementation of a sorted linked list.

(2) Given two messages of the **same** priority level, the message that has been in the ***Priority Queue*** for the **longest time** has the highest priority. (FIFO operation within a priority level).

> **HINT:** A new message of a given priority level must always be added AFTER any other messages with that same priority level that were already in the ***Priority Queue.***

Between the lecture notes and textbook, we have already reviewed much of the source code that may be adapted to implement the ***Priority Queue*** operations, but you will need to modify this code to complete this assignment. Two of the key modifications will be the addition of the specified exception detection/handling and the "previous" links. The other modification requires you to adapt the container code to store a User-Defined Data Type (of type **Message).**

Additional details may be found on subsequent pages of the handout.

Running the Sample Solution on blackhawk

The best description of what your code must do is the Sample Solution for the project.

Run the sample solution by typing the following at **blackhawk** terminal window command prompt where `inputfilename` is the name of one of the provided input files (for example, ***p04input1.txt***).

```
/home/work/cpe212/project04/p04    inputfilename
```

Your current working directory must contain the input files for this to work.

Unzipping Sample Input Files on blackhawk

Use the Firefox browser to access Canvas and download **main.cpp** and the sample input files into your **Project04** directory. At terminal window prompt, use the unzip utility to uncompress the files. For example, to unzip the files into your current directory:

```
unzip  Project04_Materials.zip
```

Since this project is worth five points, you have been given three input files to test your program.

Running the Preview Script on blackhawk

Preconditions for running the Preview script:

(1) ***Your current working directory must contain both your project04 executable and all sample input files BEFORE you execute the preview script!!!***

(2) Linux is case sensitive – be sure the name of your executable is ***project04***

Run the preview script by typing the following in a **blackhawk** terminal window command prompt

/home/work/cpe212data/project04/preview04.bash

This script will run both the Sample Solution AND your project04 executable program on the complete set of input files, and it compares the outputs of the two programs line by line

to identify errors in your program's outputs. Make sure that the output of your program exactly matches the output of the Sample Solution.

Be sure to fix any memory leaks identified by the preview script to receive full credit.

Project04 Files Provided – DO NOT MODIFY OR SUBMIT THESE FILES **

priorityq.h – declares the ***Priority Queue ADT*** class and describes its member functions

message.h – declares the **Message** class, the type of data stored in each ***Priority Queue*** node.

main.cpp – includes the **main()** function test driver for the ***Priority Queue*** class.

makefile – includes all commands required to compile and link your project on ***blackhawk***

Do not modify or submit any of the provided files named above** !!!

Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.
All output to the monitor *(stdout) will be performed by the code provided.*

Project04 FILES YOU MUST WRITE AND SUBMIT **

message.cpp – add your code here to implement the member function of the Message class.

priorityq.cpp – add your code here to implement PriorityQ functionality

The code in message.cpp and priorityq.cpp must not include any INPUT or OUTPUT statements.

PRINT functions have been included in message.h and priorityq.h for debugging purposes

Use of the Standard Template Library containers or container adapters is not allowed.

===> Submit only the SOURCE CODE in message.cpp and priorityq.cpp for grading. <===**

****Failure to satisfy these requirements will result in zero credit (0 points) on this assignment.**

How to get started on this assignment?

See message.h and priorityq.h for descriptions of Message ADT and PriorityQ ADT.

Once again, I recommend creating an Empty Framework of function definitions in message.cpp and priorityq.cpp that compiles first – before adding any code to any function.

Complete the entire Message ADT first and then address the critical operations within PriorityQ functions [constructor, Enqueue(), Dequeue()] since these critical items must be functional in order to test other operations.

Make sure these critical operations work before adding code for any non-critical operations.

Program Compilation Instructions

This project consists of two C++ files provided by the instructor, along with a file named **makefile** to help you compile your program. So, for this assignment, all you must do to compile the program is to use the following command at the Linux command line

make

which will create an executable named **project04** from the provided files.

If your program compiled successfully, you may then type

./project04 NameOfInputFile

to execute your program assuming that the input file is located in the same directory with the executable.

Submission Instructions

Submit only the files message.cpp and priorityq.cpp as attachments to your Canvas dropbox submission.

Submissions that are incomplete will receive zero credit (0 points).

Submissions that do not compile will receive zero credit (0 points).

Submissions by email will receive zero credit (0 points).

Submissions that are late will receive zero credit (0 points).