CS3100/5100: Data Structures and Algorithms

Summer 2019

Programming Assignment #1

1. Overview

The objective of this programming assignment is to practice your C++ programming skills. In this programming assignment, you will implement an abstract data type queue, and test its functionality by writing some sample test programs in main.cpp.

A queue is a container of objects that are inserted and removed according to the first-in-first-out principle. **The header file Qqueue.h will be provided to you.** Qqueue.h is the header file that specifies the interface of the abstract data type queue that you need to implement. You must implement all the functions in Qqueue.h. **The following basic operations and some other supportive operations must be supported by the ADT Queue.**

**Enqueue -- Add an element to the end (rear) of the queue**

**Dequeue -- Remove an element from the front of the queue**

**front -- return the element that is on the front of the queue (DO NOT remove it)**

**back -- return the element that is on the end of the queue (DO NOT remove it)**

**isEmpty -- Check whether the queue is empty or not**

**size: returns the number of elements stored in the queue**

A queue can be implemented in many ways. Here, we consider implementing the queue using a dynamic array of a given size, i.e., there is a limit on the number of objects the queue can hold. The size of the queue should be specified by the programmer or using a default size while creating the queue.

To test the queue of the string type, a file “all.last.txt” with a lot of last names is provided to you. In your main.cpp, you will write code to test all the member functions of the queue ADT. In your main.cpp, you should first create a queue of *type string* whose capacity is specified by you (should be in the order of 100). Then read the last names from all.last.txt and enqueue the last names to the queue. In other words, in the beginning of your main() function, you should open the input file, create a queue: prompt the user for the capacity of the queue (should be in the order of 100 or larger). Then read the specified number of the last names from the input file, all.last.txt, and enqueuer these last names into the stack. The number of last names enqueued to the queue should be less than the capacity of the queue. After you enqueue the last names onto the queue, you should test ALL the other functions in queue.h. You should print out a message describing the operation you tested.

Then you should create a queue using the copy constructor with the specified capacity. After you modify the second queue by enqueue and dequeuer operations, then you can test the equal( ) function.

In your main() function, you will provide a **user interface** **(MENU)** to test the member functions of the queue ADT and to support the following operations:

1. Enqueue: prompt user for a last name, and enqueue it to the back (end) of the queue, then print the number of the elements in the queue, and also the last 5 elements in the queue to the screen if the number of elements in the queue is larger than 5.

2. Dequeue: prompt user for a number, and if the number of the elements in the queue is larger than the number entered by the user, dequeue the specified number of the elements from the queue, and print the last names dequeued from the queue. Otherwise, dequeue all the elements from the queue.

3.Front: print the last name on the front of the queue.

4. Equal: Returns true if the two queues contain the same elements in the same order.

5. Print: Print all the elements in the queue onto the screen.

6.Save: save all the elements in the queue to a file on the disk.

7. Quit: clear all the elements in the queue and deallocate the memory, so there is no memory leak in your program, and quit.

2. Requirements

The main.cpp file includes code that will test the implementation of your member functions of your queue ADT.

1. In Qqueue.cpp, you must implement all the functions specified in Qqueue.h.

1. Your code should follow the Code Standards posted on Pilot. Your code will be graded according to its correctness, efficiency, organization, and readability.

3. Make sure that each file includes your name and email address in the header comments.

1. Code Submission and Testing
2. In order to use the c++ compiler environment installed under the school's unix server, unixapps1.wright.edu, you need to connect to this unix server remotely using a secure shell client, putty. You can remotely connect to this unix server, unixapps1.wright.edu, on campus from a Wright State computer or use your own laptop connecting to the WSU wifi network named “WSU-Secure”. Note that you cannot remotely connect to this computer using ssh using computers outside Wright State University without installing VPN or use the campus “WSU\_EZ\_CONNECT” wifi network. If you want to edit your C++ source programs under windows, download notepad++. Then edit your source programs using notepad++. After you finish editing the c++ source programs, using the secure file transfer client (WinSCP, you can download it online, and install it on your personal computer) to transfer your C++ source programs to the school’s unix server, unixapps1.wright.edu.

2. You must submit an ELECTRONIC COPY of your source program through Pilot before the due date. If for some reason Pilot is unavailable, submit your source code to the instructor Meilin Liu.

1. Submit the source code files including main.cpp, Qqueue.cpp, a compiling command, possibly a README file. You are recommended to submit a README file. It is also required that you write down your name and email address in all the C++ source programs and in the README file and also explain your programs clearly in the README file. Compress all these files into a zip file, and just submit the zip file. If you do not submit one zip file, 5 points will be deducted.
2. All the submitted project files should have: Course Number / Course Title, Your Name, Prof.’s Name, Date, and the Project Name. If you did not include these required contents in your submitted files, then 5 points will be deducted. You also need to submit a compiling command to compile your source codes. If not, another 5 points will be deducted.

5. The instructor will test your programs under the schools UNIX environment, e.g., unixapps1.wright.edu. It is YOUR responsibility to make your programs workable and runnable by others under school’s UNIX environment.

6. The programming assignment is individual. You must do the project by yourself. If you allow others to copy your programs or answers, you will get the same punishment as those who copy yours.