CPSC 2430 Data Structures

Homework Assignments #2

Assigned: 10/05/2016

Due by 9:20AM, 10/12/2016, Wednesday

1. Problem

Queues are an ADT that is specifically designed to operate in a FIFO context (first-in first-out), where elements are inserted into one end of the queue and extracted from the other. Please refer to https://en.wikipedia.org/wiki/Queue_(abstract_data_type) for more details.

In this assignment you need to design and implement a queue named MyQueue using a linked list. MyQueue stores integers.

MyQueue must support the following operations:

- `empty`
- `size`
- `front`
- `back`
- `push`
- `pop`

`MyQueue()`	Constructor: to construct an empty queue
`~MyQueue()`	Destructor
`bool empty() const`	Returns whether the queue is empty: i.e. whether its size is *zero*.
`int size() const`	Returns the number of elements in the queue.
`int front() const`	Return the next element in the queue.
`Int back() const`	Return the last element in the queue.
`void push(int val)`	Inserts a new element at the end of the queue, after its current last element. The content of this new element is initialized to *val*.
`void pop()`	Removes the next element in the queue, effectively reducing its size by one. The element removed is the "oldest" element in the queue whose value can be retrieved by calling member front().

A queue can be accessed on both ends. So MyQueue needs to be able to access the head and tail nodes in its underlying linked list efficiently. In order to do so, you may have to maintain two pointers: head pointer and tail pointer, pointing to the first and last nodes, respectively.

2. Provided Files

You are provided with two files to facilitate your code testing.

client.cpp: This is a driver program to test if your MyQueue works properly. You should not touch anything on the code. When you execute the executable client, you need provide an argument that has the data file for your testing, e.g., ./client data.in

data.in: This is a sample data file for your testing. Each line represents an operation performed on the MyQueue object. There are two operation: "push" operation followed by an integer to be pushed, and "pop" operation. It is a text file. You can use your own text file for further testing.

3. Submission

You need to submit the following files:

- myqueue.h: header file for MyQueue class. You need to include Redundant Declaration in the header file.
- myqueue.cpp: implementation file for MyQueue class.
- client.cpp: a client program to test the MyQueue class. The code is provided.
- Makefile

Before submission, you should ensure your program has been compiled and tested (extensively). Your assignment receives zero if your code cannot be compiled and executed.

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, you should follow two steps below (assuming your files are on cs1.seattleu.edu):

1). Wrap all your files into a package, named ***hw2.tar***

```
tar -cvf hw2.tar myqueue.h myqueue.cpp client.cpp Makefile
```

2). Submit your newly generated package ***hw1.tar*** as the first programming assignment ***p2***

```
/home/fac/zhuy/class/CPSC2430/submit p2 hw2.tar
```

4. Grading Criteria

Label	Notes
1a. Submission (1 pt)	All required files are submitted.
1b. Compilability (1 pts)	Your Makefile can compile the code and generate the executable file.
1c. Format & Style (1 pt)	Clean, well-commented code. No messy output/debugging messages. For C++ comments, please refer to https://google.github.io/styleguide/cppguide.html#Comments for more details.
1d. Functionality (7 pts)	The MyQueue class should behave as specified. All operations are implemented and tested.
1e. Overriding policy	If the code cannot be compiled or executed (segmentation faults, for instance), it results in zero point.