Homework 2: Firework Factory

Due Date

Tuesday, March 1 @ 2 hours after 11:59pm (2am Wednesday morning)

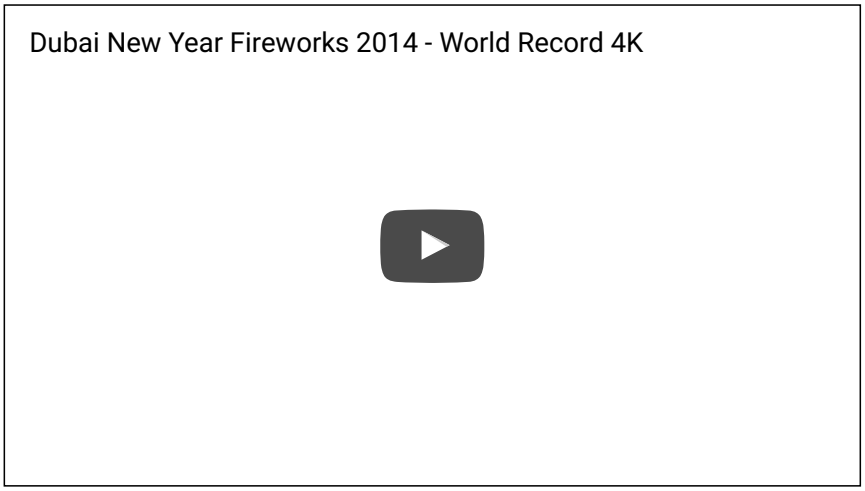
Objectives

- Practice with stack, queue, and enum
- Dynamic memory allocation on the heap (new and delete)
- Continued practice with C++ syntax

Resources

Stack: http://www.cplusplus.com/reference/stack/stack/ (http://www.cplusplus.com/reference/stack/stack/)
Queue: http://www.cplusplus.com/reference/queue/queue/ (http://www.cplusplus.com/reference/queue/queue/)
Enum: http://en.cppreference.com/w/cpp/language/enum (http://en.cppreference.com/w/cpp/language/enum)
Colored Fire: https://en.wikipedia.org/wiki/Colored_fire (https://en.wikipedia.org/wiki/Colored_fire)

Firework Factory



Danger: Never handle explosives unless you are a trained professional.

Warning: This is strictly a C++ programming assignment. It has nothing to do with actual fireworks or explosives of any kind.

For this assignment you will be implementing a FireworkFactory class in C++. This class manages the construction, purchase, and sale of different colored fireworks. Since the shelf life of a firework is limited, they must be sold in FIFO order to keep the factories inventory as fresh as possible. In addition, there will be a time when each factory is shut down. In this event, it is important to properly dispose of its entire inventory of fireworks (delete the objects).

LIFO and FIFO Inventory (1 point)

Implement this pair of functions to receive and ship fireworks while maintaining a proper order.

```
void fireworkShipment(stack<Firework*>&);
```

You will receive a stack of firework pointers via fireworkShipment and you must add the pointers to the factory's inventory starting with the top of the stack.

```
void sellFireworks(stack<Firework*>&, int quantity);
```

In sellFireworks, fireworks must be shipped in FIFO order to keep the inventory as new as possible. whether fireworks were delivered or created, they must maintain this FIFO order.

Create and Destroy Fireworks (1 point)

Implement functions to manage the creation and destruction of fireworks. It is important that every firework that is created with new is destroyed with delete exactly once.

```
void metalShipment(stack<Metal>&);
```

Receive a shipment of metal which will be used to make new fireworks in the factory. Whenever the factory has 5 of any type of metal, it must immediately make a firework of the corresponding color and add it to the inventory. To make a firework, you must use the new keyword to create it dynamically on the heap and manage a pointer to the firework.

Metals and colors are both defined using enums.

Metal	Color
CalciumChloride	Orange
CopperSulfate	Green
CopperChloride	Blue
PotassiumChloride	Purple

```
~FireworkFactory();
```

Before destroying the factory you must properly dispose of all the fireworks in your inventory (On the heap). Do this by calling delete on each firework in inventory exactly once. Leaving fireworks on the heap (memory leak (https://en.wikipedia.org/wiki/Memory_leak)) will fail the test for this point. Deleting a firework more than once will cause a runtime error.

Sell Fireworks by Color (1 point)

```
void sellFireworks(stack<Firework*>&, int quantity, Color color);
```

Some customers want to purchase fireworks all of a specific color and are not satisfied with just taking the oldest fireworks in stock. The FIFO ordering must still be maintained as much as possible, but fireworks of the specified color can be sold before other colors if there is no other way to fulfil the order.

Be sure not to sell the same firework more than once with either of the sellFireworks functions. Fireworks that are sold are launched by the customers. If two customers launch (delete) the same firework it will cause a runtime error. You must also be careful not to delete a firework that has been sold.

O(1) Efficiency (1 point)

Complete all functionality using only O(1) runtime operations in all data structures that maintain inventory for the factory. Runtime is in terms of the size of the inventory.

> This can be a difficult point to obtain and it is meant to be a challenge for students motivated enough to strive for an A in the course.

Code

Download and complete these C++ files with your solutions.

hw2.zip (homeworkCode/hw2.zip)

The provided run.cpp file contains extensive testing for this assignment that is very similar to what we'll use for grading. This will not be the case for later assignments and this is done here to show you what we expect you to do in order to test your own code.

> Caution: You will have to do you own testing later in this course. Do not rely on the feedback and tests we provide.

Submission

Submit: FireworkFactory.h, FireworkFactory.cpp
Submissions must be made individually on the submission website: https://www-student.cse.buffalo.edu/planetexpress/ (https://www-student.cse.buffalo.edu/planetexpress/)

Solution

hw2-solution.zip (homeworkSolutions/hw2-solution.zip)

Results