CMPS 109 Winter 2017: Assignment 2

Jack Baskin School of Engineering
University of California, Santa Cruz
Computer Science Department
Dr. Karim Sobh

Problem 1

Consider you have an embedded application to be deployed on a set of card readers for a classroom seats management and attendance system. An institute deploys this application on a set of 8 classrooms, with each classroom having 256 available seats. Seats are pre-assigned to students, where by the same seat will be occupied by the same student in all classrooms; i.e. no two students can have the same seat number ever. The seat number of the student will be hardwired on his card. Each student will stamp upon entering and upon leaving a specific class. A student cannot exist in two classes at any point in time. The whole idea is to be able to know how many students are there in any particular class, inquire about a specific student location in any class at any point of time, and to be able also to inquire on the availability of a specific student at any point in time. The maximum number of students that can go to any class is 256, and students are numbered from 0 to 255.

Implement the needed C++ classes that model this problem. The classes will be used in an embedded application (card reader) where storage constraints apply, and thus they should be optimum from the storage point of view, while being as efficient as possible during execution. Also, you cannot assume the existence of the STL in the target embedded deployment environment.

Your classes should provide all the needed constructors and destructor, in addition to implementing the following methods:

1. Allow class users to reserve and avail classroom seats.

2. Implement a method that checks if a specific seat is occupied in any particular class.

3. Implement a method that returns the number of available seats in a specific class at any point in time.

4. Inquire about the location of the student.

Apply all validations that you viable, and again, your classes should be self contained and should not use the C++ STL library, but should be based on built-in C++ types.

Note: you can consider that all the eight card readers are synchronized

What to submit

1. All your code.

2. You code should be split among header files (.h) and source files (.cpp), and all your implementation need to be in the source files. You need to have a very good reason for including implementation in header files, and an explanation of the motive needs to be included in your report.

3. Very detailed in-code documentation.

4. You need to provide a main program to show and illustrate all the functionalities that your program can perform.

5. A single make file that can compile the three programs by default or a specific program based on the command line arguments.

6. A report describing how to build the programs using g++, including all design decisions for the three problems, and explains anything unusual to the teaching assistant. The report should be in PDF format.

7. Do not submit any object, or executable files. Any file uploaded and can be reproduced will result in mark deductions.

Important Assumptions:

Some specifications that are built by default in the compiler's builds should not be assumed for granted. Please read the documentation of QNX as an example of a real-time operating system that withdraws some of the features granted by some builds of the compiler for better performance. http://www.qnx.com/developers/docs/660/index.jsp?topic=%2Fcom.qnx.doc.ide.userguide%2Ftopic%2Fmemory_Illegal_dealloc_.html.

This is just an example, and my point is to urge you to try to be as conservative as possible with practices you usually do and are guarded when the compiler default build options are being enabled. You are developing code that you do not know who is going to be using it, and where is it going to be deployed.

How to submit:

The git repo is now ready for you to checkout. Use this tutorial at https://git.soe.ucsc.edu/~git/index.html to setup your repo local version. Include everything under the folder hw2. Keep on updating your repository as many times as you need. When you are confident of your submission, submit the commit Id of the desired version to the ecommons.

Delays

You have up to 2 working days of delay, after which the assignment will not be accepted and your grade in that case will be ZERO. For every day (of the 2 allowed day), a penalty of 15% will be deducted from the grade. And of course, you will lose the 10% mentioned in point 1 above under the "Grade" section.