CMPS 109 Winter 2017: Assignment 2

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

Assigned: Monday, January 23rd in Class

Due: Monday, February 6th at 16:00

Delayed submission are allowed with penalty until Wednesday 8th mid-night

Goals

This assignment is a programming assignment that is designed to get you ready for the project. The assignment is composed of three programming problems that you will need to submit separately, each with its own main program.

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 applies, 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

Problem 2

Consider an embedded application to be deployed on a scanning device that checks in/out products into a manufacture inventory. The manufacture manufactures 256 different products each product can have up to 8 colors. Each product has a unique ID from 0-255. Each time an instance of a product is checked in the corresponding product stock is incremented, and each time checked out the stock gets decremented. Product stock should be categorized by product color. A product with empty stock cannot be checked out, and the inventory can have a maximum capacity of 256 items of each product per color.

Implement the needed C++ classes to model this problem. This class will be used in an embedded application (scanning device) where it will run in an environment that you cannot assume the existence of the STL. The class should optimize both storage usage and execution efficiency.

Your class should provide all the needed constructors and destructor, in addition to implementing methods that allow class users to add and remove product items to and from the inventory. Moreover, you need to implement methods that performs the following functionalities:

- Checks if a specific product and/or color exists in the inventory.
- How many items exist for a specific product ID.
- How many items exist for a specific product ID of a specific color.
- How many items exist for a specific product color; cross products.
- Number of product types having stock in the inventory.

Again, your inventory classes should be self contained and should not use the C++ STL library but should be based on built-in C++ types.

Problem 3

The MAX heap is a data structure that can be used by a priority scheduling. A priority scheduler will always need the highest priority processes to be stored such that when retrieving a process from its queue, the highest priority process is at the front end of the heap data structure and its retrieval requires the least amount of processing. You should assume that your class may be used as a utility class that can be used and deployed on hybrid systems and deployment environments.

Using the below partial implementation, implement a C++ class for a MAX heap of integers that should represent the priorities of different processes. Your class should have the appropriate constructors (including a copy constructor) and destructor. In addition, your class should provide the following operators and functions:

- **operator+** to allow for adding of two heaps (e.g., heap3 = heap1 + heap2 results in heap3 contacting the element of heap1 and heap2).
- **operator+** to allow for adding a single integer to a heap (e.g., heap2 = heap1 + 5 results in heap2 containing the elements of heap1 and the value 5)
- **operator[]** to allow for accessing the individual elements of the heap as they were stored in a sorted array (e.g., heap[5] accesses the 5th largest element in the heap).
- **operator=** to allow heap assignments (e.g. heap1 = heap2 = heap3)
- **operator+=** to allow heap assignments (e.g. heap1 += heap2 and heap3 += 5)
- **operator<<** to allow for printing heaps on cout (e.g. cout << heap1 prints heap1 on cout).

You have to use the below pseudo code and make it workable and extend it to achieve all the above requirements.

```
class Heap
{
        private:
                int * array;
                int MaxSize, Nel;
                void adjust (int a[], int i, int n);
        public:
                Heap(int MSize);
                bool insert (int item);
                bool delMax(int & item);
                ~Heap();
};
```

```
void Heap::adjust (int a[], int i, int n)
{
        int j = 2*i, item = a[i];
        while ( j <= n )
        {
                if (j<n && (a[j] < a[j+1])) j++;
                        // Compare left and right child
                        // and let j be the larger child
                if ( item >= a[j] ) break;
                a[j/2] = a[j]; j*=2;
        }
        a[j/2] = item;

}
Heap::Heap(int MSize):MaxSize(MSize)
{
        array = (int *) calloc(MSize+1,sizeof(int));
        Nel = 0;
}
bool Heap::insert (int item)
{
        int i = ++Nel;
        if ( i == MaxSize)
        {
                cout << "heap size exceeded" << endl;
                return false;
        }
        while ( (i > 1 ) && ( array[i/2] < item )) {
                array[i] = array[i/2];
                i/=2;
        }
        array[i] = item;
}
```

```
bool Heap::delMax(int & item)
{
        if ( !Nel) { cout << "heap is empty" << endl ; return false; }
        item = array[1];
        array [1] = array[Nel--];
        adjust (array,1,Nel);
        return true;
}
```

What to submit

1. All your code.
2. You code should be split among header files (.h) and source files (.cpp), and all you 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. For each problem of the above mentioned 3 problems 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, and each assignment code should be in separate director named as problem<number of problem from 1-3>.
7. Do not submit any object, or executable files. Any file uploaded and can be reproduced will result in mark deductions.

Important Assumptions:

As stated in all problems, the deployment environment is either limited (problem 1 and 2), or undefined (problem 3). 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 you 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.

Grade

This assignment is worth 10% of the overall course grade. The assignment will be graded on a 100% grade scale, and then will be scaled down to the 10% its worth. The grading of the assignment will be broken down to the following:

1. 10% for submitting the assignment. A submission that contains code and reports i.e. if you just submitted an empty assignment you will get nothing. This 10% does not account for the correctness of your assignment.
2. 30% for problem 1.
3. 30% for problem 2.
4. 30% for problem 3.

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.