Assigned: Monday February 9, 2015, **Due: Wednesday, February 18, 2015, 11:59pm.**

Tufts SIS (lite) Designing a Class Roster with Waitlists

1. Overview

As much as everyone likes to criticize Tufts SIS, it does have a difficult job to do. In this assignment, we will be modeling the SIS task of placing students into a course that has certain requirements. Namely, we will be allowing students who are majoring in a subject to go directly in to the class roster, and those that are not majors will go onto a waitlist. Additionally, if the class is at a defined capacity, major students will be placed onto a waitlist of their own.

We will be using two Abstract Data Types (ADTs) for this assignment: a Set and a Queue. We have discussed both in class, and for this assignment, we have built a Set for you, and you will be responsible for building the Queue based on a header file for a Queue. You will be limited to standard Queue functions: **`enqueue(), dequeue(), and is_empty()`**. You are not allowed to write any other public functions for the Queue.

We will model a Student with a **`struct`** that includes two fields: name (a string) and major (a boolean value for whether the student is a major or not).

We will be modeling the Course itself with a class called "**`IsisCourse`**". For the assignment, the Set will model the actual class roster, and the Queue will model the two waitlists. Students will attempt to enroll in the class, and based on the rules above (and listed in **`IsisCourse.h`**, as well). Students can be dropped from the class, and when a student is dropped, Students come off the waitlist in the priority described above, up to the class capacity.

The class capacity can be raised (but not lowered), and if that happens, Students come off the waitlists according to their waitlist (majors first), and by their position on the waitlist (who registered first).

2. Implementation Specifics

We have provided you the **`Set`** and **`Student`** implementations in full. We have also provided you with the header files **`Queue.h`** and **`IsisClass.h`**, and we have started the **`Queue.cpp`** and **`IsisClass.cpp`** files. There is also an example **`main.cpp`** file that demonstrates how the program will be used. We have provided **exact** output for **`main.cpp`**, (called **`default_main_output.txt`**) and in your testing you should ensure that your output is identical to that output. You can do that as follows:

```
./hw3 > my_output.txt
diff my_output.txt default_main_output.txt
```

You should, of course, also write your own tests for the Queue functions and IsisClass functions that you write.

You should read the header files carefully, as we have annotated them extensively to describe how we want the program to behave. Obviously, if you are unclear on the expected behavior, post a question on Piazza so we can clarify the goals.

4. Low Level Details

Getting the files

There are two ways to get the files for this assignment. The first is by copying the original files from the class folder. The second is to use "git" to pull the files from the GitHub cloud server.

Method 1: copy files from the class folder

First ssh to the homework server and, and make a directory called orderedListAssignment

```
ssh -X your_cs_username@homework.cs.tufts.edu
mkdir hw3
```

change into that directory

```
cd hw3
```

At the command prompt, enter (don't forget the period):

```
cp /comp/15/public_html/assignments/hw3/files/* .
```

Method 2: pull from GitHub:

At the command prompt, enter

"`git clone https://github.com/Tufts-COMP15/2015s_HW3.git hw3`"

change into the directory that git created:

"`cd hw3`"

Using Eclipse (assuming you have followed the steps above to get the files):

1. See online video https://www.youtube.com/watch?v=BbnLZ2oqrD0
2. If in the lab, simply open Eclipse, and then start following from step 6 below.
3. If at home, ensure that you have installed (Mac) XQuartz, or (Windows) Cygwin (see here: https://www.youtube.com/watch?v=BbnLZ2oqrD0) or (Win) putty and XMing (http://sourceforge.net/projects/xming/).
4. ssh to the homework server using the -X argument:
   ssh -X your_cs_username@homework.cs.tufts.edu
5. start eclipse by typing "eclipse" (no quotes)

6. Use the following steps to set up your project (you'll get used to the steps quickly, steps A & B only need to be done once at the beginning of the course):
   A. Default location for your workspace is fine
   B. Click on "Workbench", then click "Window->Open Perspective->Other", and choose C++
   (note: if C++ is not listed, close Eclipse and re-open by typing Eclipse)
   C. Window->Preferences
      General->Editors->Text Editors
        Displayed tab width (8) (recommended)
        Show print margin: 80 col

Show line numbers
C/C++ -> Code Style
Select K&R [built-in] and then “Edit”, then rename to “K&R 8-tab”
Change Tab size to 8 (recommended)
Click “Apply” then “Ok”

D. File->New C++ Project
Fill in project name (no spaces!) IMPORTANT: the name CANNOT be exactly the same as your folder name. I suggest to simply put name_project (with the _project) for all project names (e.g., hw3_project)
Use default location should be checked.
Select Empty Project->Linux GCC
Click Next
Click "Advanced Settings"
1. On the left, under C/C++ General->Paths and Symbols (left hand side) (you may have to click on the triangle next to C/C++ General)
Select "[Debug]" at the top for Configuration
Source Location Tab
Click "Link Folder"
Check "Link to folder in the file system"
Browse to find your folder.
(should be something like h/your_username/hw3)
Click OK
Click Apply

2. C++ Build->Settings (may have to click on the triangle)
For Configuration (at the top), select [All configurations]
GCC C++ Compiler: Command should be "clang++" (no quotes)
GCC C Compiler: Command: clang
GCC C++ Linker: Command: clang++
Click "Apply"
Click OK

Click Finish

E. Click on the white triangle next to your project name.
The folder you linked to should be listed. Click on its triangle.
The files in that folder should be listed.

F. Test the build by clicking on the hammer in the icon bar.
You should see some compiling messages in the Console window at the bottom. (Things like, "Building file...; clang++, etc.)

G. The program will compile, but will have many warnings. You need to write the functions — double-click on List_linked_list.h to see the header file and List_linked_list.cpp to see the code.

Compiling and running:

1. At the command prompt, enter “**`make`**” and press enter or return to compile.
2. At the command prompt, enter “**`./hw3`**” and press return to run the new program.

Providing:
At the command prompt, enter “**`make provide`**” and press enter or return.