# Northeastern University ETC 2103 – *Data Structure Applications in C++*

Final Project 30 points

The final project involves implementing a simple student grading system. The grading application will have the following functionality:

1. Store Student records in a generic container.
2. Update Student records
3. Search a Student’s record
4. Remove a Student from the container, and
5. Display all student’s names in alphabetic order

The interface the grading application provides is displayed in a Menu. The menu consists of commands available to the end-user. The commands are:

* 1. *Insert* a student’s data into the system. The container will allow duplicate names to be inserted.
  2. *Find* a student’s data, given the student’s name (key). If there are multiple students in the container with the given name, the application will display information about all of them.
  3. *Modify* a student’s record. Specifically, it will let the user modify the GPA. If there are multiple students with the same name, the application will display their records and ask the user to choose the one to modify.
  4. Remove a student record from the system. If there are multiple students with the same name, the application will display their records and ask the user to choose the one to remove.
  5. Display just the names of students in alphabetic order.
  6. *Exit* the system

Design/Implementation

The application will use the generic associative container multimap. The element type of an associative container is a pair.

The multimap stores pairs of <string, Student>. The library type named pair

is defined in the header <utility>. You define a pair consisting of a Student name and student info as follows:

pair<string, studentInfo> aStudent;

Here string is the first parameter and Student the second parameter.

A non-member, convenience template function make\_pair() is available to create the value pair without writing the types explicitly. For example, the following statement

make\_pair(“Nicholas”, Nicks\_Record);

will create the element <”Nicholas”, Nicks\_Record> which can be inserted into the multimap as shown:

multimap<string, studentInfo> student\_1 ; // Empty map

student\_1.insert(make\_pair(“Nicholas Cage”,Nicks\_record));

student\_1.insert(make\_pair(“Bob Murphy”, BobM\_record));

The student records are searched by the student name (the key). The data associated with the key is an object of type studentInfo. The application will define a studentInfo class (in a separate header file) with the following attributes:

studentName (First name and last name)

studentID, (an integer)

GPA (a double)

In addition to using the multimap container, the application will use STL algorithms

where appropriate.

The application will allocate the multimap on startup. It will populate the container with information for five or six students. After populating the container, the application will display the menu and wait for a user selection.

As part of the test of the application, you should insert new students, modify student grades, lookup student record given the student’s name, remove a student from the container and display the list of student names in the container.

To display the list of student names, you should create iterators to traverse the container and copy the student names from the container to a vector of strings using the generic copy algorithm. Please note that you are displaying just the student names and no other information. If there is more than one student with the same name, you should also display the number of such students in parenthesis.

Deliverables

Compile and test the system. Submit the following on Blackboard.

1. The source files (.h and .cpp)
2. Screen shot of the output.