CSEN 79L: OOP and Adv. Data Structure

Lab 3: A container class for a course roster.

This is an exercise of implementing a simple container as "bag" in the textbook. For this exercise, we will use an array to hold data. In future version of this same exercise, we will use more advanced data structures.

Design and implement a C++ class representing a typical course roster. A typical use case"

```
      Int main(void]) {
            …
            Roster csen79;
            csen79.insert(/* student information here */);
            cseo79.erase(/* ID */);
            …
      }
```

1. Declare two classes in a header file roster.h. First "Student" that capture relevant student information such as ID, first name, and last name. Secondly, a "Roster" that has an array (initially size 30) that stores the student roster for a class.
   In a separate file, roster.cxx, finish the class definition. Put your test code in rostermain.cxx.
   Write a Makefile to compile and build the program.
   The Roster class should have 3 public members functions as in the example above.
   You should have a constructor for each class.
   Build your program to be named "rostertest."
2. Overload the output operator "<<" for the Student class.
3. Implement 3 member functions for Roster class: `begin()`, `end()`, and `next()`
   The function begin() will return a pointer to the first record of the roster data; end() the final one, and next() the next one relative from the previous retrieval (begin() counts as the first retrieval, but not end()).
   Traverse the records in roster like this:
   ```
   for (auto a = roster.begin(); a <= roster.end();
     a = roster.next())
     cout << a;
   ```
4. "rostertest" should read from stdin line by line and process the lines immediately after reading them. It should output plain text to stdout. Therefore, you can run your program as:
   ```
   rosterTest < input_file > outout_file
   ```
   a) Each line begins with an "operator" character that is "A", "X", or "L".
   b) "A" line indicates the ensuing data to be inserted into the roster. The data followed will be an ID, first, and last names.
   c) "X" line indicates the ID'ed record to be removed from the roster.
   d) "L" line has no data. The program is to print all members in the existing roster to stdout.

   A sample input file is provided as `lab3sampleinput.txt`
5. Write a "test plan" enumerating the "test cases" and the steps to conduct the tests. A test case describes the program behavior under certain conditions. Make sure to cover both

positive and negative conditions. For example, a simple test plan can be: "it should work when the data is valid, it should print an error message when not."

6. Submit source code, test plan (in PDF or plain text), and program output (as a text file).