CSEN 79L: Lab 4 Roster using Pointers

Switch the data storage to pointer, from the previous lab as C-style array.
In the previous lab, the class "Roster" container uses a fixed-sized array to store records of "Person" objects. In this lab, we use a pointer instead.

Assignment

Derive a new class from Roster as "RosterPtr" as in the provided code. Think "RosterPtr" has everything of "Roster". If "RosterPtr" has members of the same name, then those in "Roster" become hidden to the objects of the RosterPtr class.
(What we really use from Roster are the Command enum and "T" type which is an alias of Person. We will redefine all other members.)
Specifically:

1. Write a constructor that initialize all the data members as in the provided code. The members "idx" and "last" are meant for insertion and iteration. You may remove them as you see fit. The other two members — roster and capacity — are probably best left alone.
2. Implement the "insert" function to allocate memory blocks when the existing capacity has been exhausted.
   (Every time insert needs to increase the capacity, it should allocate new memory, copy over the old record, and de-allocate the old. Of course, it should also update various tracking members.)
3. Write the destructor for the Roster class to deallocate whatever previously allocated.
4. Implement "begin", "end", "next" as well.

The program, therefore, behaves as if there is infinite capacity and should be able to handle arbitrary number of data records. Examine "erase" and iterator member functions to see if they need to change or not.

5. Write a test plan that describe how would *someone else* will test your program such that when he/she finishes, there are evidence that the program indeed behaved as designed. Implement your "main", possibly with external data (such as what was used in lab3), to carry out the test plan.
   Note main could be very similar to lab3's version.