CS300/CS500
Advanced Object-Oriented Programming with C++

Lab #4:

Part I: Create a deque data structure. Recall a deque is like a queue or a stack, but elements can be accessed or pushed from the front or the back. The deque should be able to hold up to 1000 ints, and this data should be dynamically allocated on the heap (using the **new** and **delete** keywords). Define your deque as a class, and include the member functions: **push_front(), push_back(), pop_front(), pop_back(), peak_front(), peak_back(), remove_all()**.

- push_front() and push_back() should add an element to the deque from the front or the back respectively.
- pop_front() and pop_back() should return the value at the front or back of the deque, and delete the given value.
- peak_front() and peak_back() should return the value at the front or back of the deque.
- remove_all() should free all memory allocated on the deque, and the deque should be empty after running this. *It is best practice to include a deconstructor that calls this function.*

Make sure you account for the case that the deque is empty in all member functions. Lastly, this data structure should be memory safe, meaning that the remove_all() function should free all memory you have allocated. **5 points**

If you are confused, check the stack example at the end of the slides.
You should be able to complete this assignment by modifying that example

Part II: Create a class called User. This class should have member variables userName, email, and a collection (array, vector, or anything that works) of pointers to other users called friendList. Assume each user has no more than 100 friends. Include member functions addFriend(), removeFriend(), numFriends() and getFriendAt(). As well as setters and getters for the userName and email variables. **5 points**

Include the member functions:

- addFriend() should take a pointer to a User and add it to friendList
- removeFriend() should take a username as a string, and remove the corresponding user from friendList
- numFriends() should return an int corresponding to how many users are in friendList
- getFriendAt() should take an index and return a pointer corresponding to the user at the index in friendList *Note: this should never return a nullptr as long as the index passed into this function is between 0 and numFriends()-1.*

Submission: Create a deque.h and deque.cpp file, include the prototype for your deque in deque.h, and the definition in deque.cpp. Similarly create the files user.h and user.cpp, include the prototype for User in user.h, and the definition in user.cpp. Push all 4 of these files to your class repo. Make sure there is no main function in any of these files. Send a link to the repo for your submission.