CS 2213 Advanced Programming Recitation - Exercise

Improving queueADT implementations with a ReverseQueue function

First copy the files under 10-Linear-Structures from the class web page (follow the link "programs from the textbook" and then go to 10-Linear-Structures). You can get all files, but you will mainly need the following files for this recitation...

Makefile   qarray.c   qlist.c   qtest.c   queue.h   scanadt.c   scanadt.h

We already studied queue.h, qarray.c and qlist.c in class. So **examine** the **qtest.c** client/driver program which simply asks user to enter some commands to enqueue and dequeue strings in a queue and display the queue elements. You can simply compile the same qtest.c driver/client program with qarray.c and qlist.c implementations of queue.h interface to create array-qtest and list-qtest executable files, respectively. scanadt.h and scanatd.c are used in both cases as a library so you don't need to fully examine or understand them to complete this recitation. To compile everything as is, just say make. Then test both array-qtest and list-qtest programs.

Now, you are asked to do the followings:

1. Add **`void ReverseQueue(queueADT queue);`** prototype into **queue.h** interface:

2. Implement this function under both array and list representations. That is, add its implementation into both **qarray.c** and **qlist.c** files

When Implementing this function, make sure that you will use the original memory cells and do not allocate any additional storage (of course you can use local variables)

3. Modify **qtest.c** client/driver program so that a user can reverse the existing queue by calling the above function that is just added to the interface.

4. Finally say make and compile qtest.c with new qarray.c and qlist.c. Then test both array-qtest and list-qtest programs....

As always, make sure you release (free) the dynamically allocated memories when you quit from your program. So, before submitting your program, run it with `valgrind` to see if there is any memory leakage…

Also if you need to debug your program, compile your programs with –g option and then run it with `gdb` and/or `ddd`.

Even though you just simply change three files, you need to zip and submit everything so that TAs can easily compile and check your implementations....

/* Don't forget to include comments about the problem, yourself and each major step in your program! */

_

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.

__