Special Member Functions and Overloaded Operators

You'll find a source file, BackList.cpp, and an expected output file on the course homepage. You can also download these with the following curl commands:

curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise26/BackList.cpp
curl -O https://www.csc2.ncsu.edu/courses/csc230/exercise/exercise26/expected.txt

This source file contains a class, BackList, that provides a resizable array, kind of like what we get with vector. However, compared to vector, BackList makes it look like its list is backward. New values pushed onto a BackList show up at index zero, and the value that's pushed first always shows up at the highest index. Internally, BackList stores its values in the same way vector does, with the most recently pushed value stored in the highest index. However, its overloaded [ ] operator reverses the indexing, so it looks like the list is backward.

This program will give you a chance to use the C++ new/delete operators to manage dynamically allocated memory for storing the array of elements in the list. It will also give us a chance to implement a copy constructor and an overloaded << operator to print instances of BackList. When your implementation is working, you should be able to run it like:

```
$ ./BackList
List1: 40 35 30 25 20 15 10

List1: 39 35 30 25 20 15 10
List2: 45 41 35 30 25 20 15 10

List1: 39 35 30 25 20 15 10
```

I've left three places for you to add code to complete the BackList implementation.

- In the push method, we need code to grow the resizable array when we run out of room. You'll need to do this with the new/delete operators, so you'll need to 1) allocate a new array with larger capacity 2) copy over the contents of the old array 3) free the old array 4) start using the new array instead of the old one.
- Fill in the body of the copy constructor so it initializes the fields of a new BackList to hold a copy of the given BackList. Don't just copy the seq pointer; that will give us two lists that are trying to use the same dynamically allocated array to store their contents. The new BackList made by the copy constructor will need its own dynamically allocated array. The code inside the code block in main will let you try out your copy constructor. It makes a new list that's a copy of an existing one (using the copy constructor).
- Fill in the body of the overloaded << operator so it prints out the contents of a BackList to the given stream. Remember to return the stream at the end of the function, so your overloaded << operator can be used as part of a longer output expression (e.g., like cout << someValue << myList << somethingElse;)

Be sure you don't change the code in main(). I'll depend on this being the same to help test your program, and I'll probably make some (automated) edits to this part of the program to see if your program can handle some other test cases.

When you're done, submit a copy of your completed BackList.cpp file using the exercise_26 assignment on Moodle.