CPSC 2120/2121
Lab 01

Ring Data Structure
Due date/time are on the CPSC 2121 Canvas website

Learning Objectives

- To improve our ability to implement algorithms and data structures in C++
- To implement a data structure using a class template in C++
- To use a simple data structure (an array) as a building block used to construct a more complicated data structure (a Ring)
- To become proficient in fundamental data structures used throughout computer science

Problem / Exercise

For this lab, you will implement a Ring data structure (also called a circular array). Ring and other circular data structures are used as memory buffers in some fields of computer science. Your main task is to implement the functions at the bottom of the file Ring.h. However, before you start implementing Ring.h, you'll need to read the Ring class definition in Ring.h, lab01.cpp, and the examples section of this document. Once you get an idea of how this Ring data structure works, then you can implement Ring.h such that when it compiles and runs with our provided Makefile on our Unix machines, the output of the testing program, ./lab01.out, should be identical to the output in the examples section (if your output is different, then you must fix your bugs before proceeding to the next step). Once your class passes our test cases successfully, add your own test cases to lab01.cpp to make sure your Ring class works with various data types and sizes. Your implementation of the functions at the bottom of Ring.h must work with any testing program that we use to test it for grading purposes, which means that you CANNOT modify the function prototypes in Ring.h. Also, you should not change the names of the provided instance variables in Ring.h nor should you add more instance variables to the class.

Examples

Your C++ source code should be in a file called `Ring.h`, and it should be compiled (along with our provided `lab01.cpp` file) into an executable called `lab01.out` using our provided Makefile. The output of our testing program in lab01.cpp with your Ring class implementation must look exactly like the examples below when run on the command line on our Unix machines.

```
./lab01.out
Test 1 - An empty ring of doubles
r1's size is 0

Test 2 - A ring that can hold 5 integers
Adding 0,1,2,3,4 to the ring
0
1
2
3
4
Add 6 and 7 to the ring,
and they will replace the first two elements of the ring.
```

```
Also, the ring still contains the last three items
6
7
2
3
4

Test 3 - A ring that can hold a playlist of 3 songs (as strings)
Adding 3 songs to the ring
Roar
Happy
Eye of the Tiger
Playlist is full, and two new songs are added
Lucy in the sky with diamonds
Hello
Eye of the Tiger
r3[13] = Hello

Test 4 - A ring that can hold 6 integers
Adding 0,1,2,...,15 to the ring
12
13
14
15
10
11
```

1 C++ Program

1.1 Code Requirements

For this lab, you should place a copy of `Ring.h`, `lab01.cpp`, and our provided `Makefile` on one of our Unix machines (in the same directory), and complete the `Ring.h` class such that it will compile and work correctly when used with other testing programs. Here are some additional requirements.

- Fill in the comment at the top of your source file(s) with your name (first and last), the date of your submission, your lab section, and the assignment's name.
- *All functions should be commented. Use inline documentation, as needed, to explain ambiguous, tricky parts, or important portions of your code.*
- *All source code must follow good programming style standards such as properly indenting source code and all variables, functions, and classes should be well-named.*
- The resulting executable must be called `lab01.out`. The expectation is that after your class is compiled and linked with our testing program, the grader should be able to run your program via the command line on our Unix machines by using the same I/O as shown in the Examples section. Your class must use the same I/O as shown in the Examples section; otherwise, you may lose points.
- We have provided a Makefile for this lab, and the final version of your code that you submit for grading must compile with our provided Makefile on our Unix machines without any modifications to our Makefile.
- Your class must use dynamic memory allocation and deallocation properly, which means your class cannot contain a memory leak when Ring objects are created. You can use valgrind to check for memory leaks, `valgrind ./lab01.out` is a command you can use to run valgrind. Refer to the Unix manual for more information on valgrind.

2 Submission

Before the date/time stated on the CPSC 2121 Canvas webpage, you need to submit your code to our CPSC 2121 Canvas webpage under the correct lab assignment. Make sure to submit all of the following.

1. All source files required for this lab (Ring.h)

After you submit, always double check that the file(s) you submitted were the correct version. To double check, download the submitted file(s), put them on one of our Unix machines, and make sure they compile and run correctly.

3 Grading: 10 points

Make sure your program compiles and gets expected output on the lab machines before submitting your work. If your program does not compile on *our Unix machines* (or your assignment was not submitted on time), then you'll receive a grade of 0 on this assignment. Otherwise, your program will will be graded using the criteria below.

Your class works correctly with a program we will use to test your class	6 points
Proper variable names, documentation, and code organization	2 points
Your class uses dynamic memory allocation/deallocation properly (no memory leaks)	1 point
At least 3 distinct, passing test cases (not included ones provided) are submitted	1 point
Penalty for not following instructions (invalid I/O, etc.)	Penalty decided by grader

You must test, test, and retest your code to make sure it compiles and runs correctly and efficiently on our Unix machines with any given set of valid inputs. This means that you need to create many examples on your own (that are different than the aforementioned examples) to ensure you have a correctly working class. We will only test your program with valid sets of inputs.