Background

In this project we will make some additions to our existing class LL<T>. One of the most significant additions is the capability to use iterators to traverse the list. This requires implementation of a new class LL_iterator<T>. Once all additions are complete we will run some experiments and record performance data for the LL<T> class' various sorting methods. We will also implement container adapters Stack<T, S> and Queue<T, Q>. Unlike the sorting experiments you are not required to demonstrate any specific behavior of the stack and queue classes. However, you should test all of the member functions of those classes to ensure that they operate correctly.

1. You must use LL<T> from Project 3 as the container for both Stack<T, S> and Queue<T, Q>.

2. The design of LL_iterator<T> is roughly based on the functionality of an iterator for the C++ vector template class.

3. Thorough documentation for this project's classes is available on-line at: http://people.cs.georgetown.edu/~addison/projects/fall2021/p4docs/index.html

Getting Started

Set up a new project with your applicable code from Project 3 as the starting point. Next add new project files and copy and paste the given class declarations and other code. Once that is done I recommend that you immediately write function stubs for **all** methods of **all** new classes. Thereafter, use stepwise refinement and incremental development to implement the new code for this project.

Academic Integrity

This is an individual project and all work must be your own. Refer to the guidelines specified in the *Academic Honesty* section of this course syllabus or contact me if you have any questions.

Include the following comments (with appropriate substitutions) at the start of each file in your project:

```
/*
 * <FileName>.<file extension>
 *
 *  COSC 052 Fall 2021
 *  Project #4
 *
 *  Due on: NOV 11, 2021
 *  Author: <your name>  <your netID>
 *
 *
 *  In accordance with the class policies and Georgetown's
 *  Honor Code, I certify that, with the exception of the
 *  class resources and those items noted below, I have neither
 *  given nor received any assistance on this project.
 *
 *  References not otherwise commented within the program source code.
 *  Note that you should not mention any help from the TAs, the professor,
 *  or any code taken from the class textbooks.
 */
```

Submission Details

You will submit Project 4 to Canvas in the same way you did Project 2 and Project 3. Optional test submissions will also be accepted. The date and time for test submissions will be announced later. A new `Makefile` is also provided for Project 4. Although you may use any development environment for your project, it must compile and execute on the class server. Make sure it compiles and runs without error on `class-1` before submitting.

What to submit: One compressed file containing all source code and the `Makefile` associated with this project. The file name must be `submit.zip`.
You must separate your class specification details from your class implementation details. Therefore, you must prepare a specification file (`<filename>.h`) and an implementation file (`<filename.cpp>`) for each set of related classes. The files `LL.h` and `LL_adapters.h` different. Since they contain template classes, the class declarations and all member function implementation code shall also be stored in the `.h` files. There are no `LL.cpp` or `LL_adapter.cpp` files for this project. Ensure that your `.h` files contain sufficient comments for each data member and class method. Additionally, you must provide file `main.cpp` (that contains function `main`) along with its associate `main.h` file. This "driver" code is where class objects are instantiated and functionality of the software is demonstrated. Use the following file names (**with spelling and capitalization exactly as shown**):

```
LL.h          (reuse your LL class code, from Project 3 with appropriate additions/modifications)
LL_adapters.h
Resources.h, Resources.cpp          (reuse your Resources files from Project 3)
Exceptions.h, Exceptions.cpp        (reuse your Exceptions files from Project 3)
ProcessTimer.h, ProcessTimer.cpp    (given code)
main.h, main.cpp
Makefile    P4ExperimentResultsTable.pdf
```

Grading

This graded assignment is worth 100 points and will be counted as part of the *Programming Projects* category for the course. Your final score is based on automated tests, as well as a manual review conducted by one of our Teaching Assistants (TAs). A detailed rubric of points and a list of common deductions will be published separately.

Due date/time: 11 November 2021, no later than end-of-day (11:59pm). Late submissions will be penalized 2.5 points for each 15 minutes late. If you are over 10 hours late you may turn in the project to receive feedback but the grade will be zero. In general requests for extensions will not be considered.

Versions and Backups

Once you have submitted your project, it is important to keep an electronic copy on a university machine such as the class server (`class-1`) that preserves the modification date and time. You should also make periodic backups or "snapshots" of working versions of your project. If some disaster occurs, it is invaluable to have a working version to which you can revert and regroup.

Driver Program

The driver program shall accept an input data file as a command line argument. If there is no command line argument passed, function `main` shall throw an `invalid_argument` exception. Demonstrate the functionality of your code in function `main`. Include code to load the given data files (which contain date values). Then call member functions of the `LL<T>` class to generate the required experimental results. If a file fails to open your code shall throw a `logic_error` exception.

Test Results

The sorting experiments consist of running each sorting algorithm with three different input files. You will use the small, medium, and large files provided with Project 3. For each run record the time required for the sort to complete and the swap count. Below are my results for comparison. You are not expected to match these results, but your numbers should be roughly equivalent.
(Note: If you run the tests on the server times could be much faster, mine were about the same)

1 November 2021 *Project #4*

Sort Performance			
List Size	**Algorithm**	**Sort Time**	**Swap Count**
Small (1020)	Bubble Sort	4.0521	261,171
	Bubble Sort "I"	0.8978	261,171
	Selection Sort	2.5742	1,015
	Insertion Sort	1.5965	261,171
Medium (2132)	Bubble Sort	33.1213	1,096,627
	Bubble Sort "I"	3.7870	1,096,627
	Selection Sort	23.3492	2,126
	Insertion Sort	13.1802	1,096,627
Large (3276)	Bubble Sort	118.0510	2,596,106
	Bubble Sort "I"	8.8513	2,596,106
	Selection Sort	84.8638	3,265
	Insertion Sort	48.1894	2,596,106

```
/*
 *  P4ExperimentResultsTable.pdf
 *
 *  COSC 052 Fall 2021
 *  Project #4
 *
 *  Due on: NOV 11, 2021
 *  Author: W. A. Woods    waw23
 *
 *
 *  In accordance with the class policies and Georgetown's
 *  Honor Code, I certify that, with the exception of the
 *  class resources and those items noted below, I have neither
 *  given nor received any assistance on this project.
 *
 *  References not otherwise commented within the program source code.
 *  Note that you should not mention any help from the TAs, the professor,
 *  or any code taken from the class textbooks.
 */
```

COSC 052 Fall 2021 *Page 1 of 1*

Creating submit.zip

Please, PLEASE, PLEASE use the provided `Makefile` and create your `submit.zip` file on the class server. If you create the compressed file on your laptop it is highly likely something will go wrong even though it looks fine. It is easy to compress links to files, instead of actual files. It is easy to have the folder containing the project files included in the compressed file. If this, or any other problems happen; your program will not compile, automated grading programs will fail, **and you will get a zero**. Assuming all of your files are in the same folder on the server, the process to create the `submit.zip` file is shown below.

```
[waw23@cs-class-1 P4]$ make clean
rm -f *.o core a.out
[waw23@cs-class-1 P4]$ make submit
rm -f submit.zip
zip submit.zip P4ExperimentResultsTable.pdf main.cpp main.h LL.h LL_adapters.h Exceptions.cpp
Exceptions.h Resources.cpp Resources.h ProcessTimer.cpp ProcessTimer.h Makefile
  adding: P4ExperimentResultsTable.pdf (deflated 0%)
  adding: main.cpp (deflated 78%)
  adding: main.h (deflated 43%)
  adding: LL.h (deflated 82%)
  adding: LL_adapters.h (deflated 79%)
  adding: Exceptions.cpp (deflated 63%)
  adding: Exceptions.h (deflated 73%)
  adding: Resources.cpp (deflated 83%)
  adding: Resources.h (deflated 72%)
  adding: ProcessTimer.cpp (deflated 69%)
  adding: ProcessTimer.h (deflated 60%)
  adding: Makefile (deflated 62%)
[waw23@cs-class-1 P4]$ unzip -l submit.zip
Archive:  submit.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
    68871  01-10-2021 00:54   P4ExperimentResultsTable.pdf
    11118  01-10-2021 00:54   main.cpp
      843  01-10-2021 00:54   main.h
    52812  11-01-2021 04:49   LL.h
     9059  01-10-2021 00:54   LL_adapters.h
     1839  10-11-2021 01:56   Exceptions.cpp
     4287  10-11-2021 01:56   Exceptions.h
    42626  10-11-2021 01:56   Resources.cpp
     8928  10-11-2021 01:56   Resources.h
     2577  01-10-2021 00:54   ProcessTimer.cpp
     1987  01-10-2021 00:54   ProcessTimer.h
      975  01-10-2021 00:54   Makefile
---------                     -------
   205922                     12 files
[waw23@cs-class-1 P4]$
```

Remove files from last compile

Create the zip file to submit

Verify the zip file contents, make sure the date and time are correct and these are the files you want to submit, you may make unlimited submissions prior to the due date, the last submission will be graded

Course Materials Notice

The materials used in Georgetown University courses ("Course Materials") generally represent the intellectual property of course instructors which may not be disseminated or reproduced in any form for public distribution (e.g., sale, exchange, etc.) without the written permission of the course instructor. Course Materials include all written or electronic documents and materials, including syllabi, current and past examination questions/answers, and presentations such as lectures, videos, PowerPoints, etc., provided by a course instructor. Course Materials may only be used by students enrolled in the course for academic (course-related) purposes.

Published course readings (book chapters, articles, reports, etc.) available in Canvas are copyrighted material. These works are made available to students through licensed databases or fair use. They are protected by copyright law, and may not be further disseminated or reproduced in any form for distribution (e.g., uploading to websites, sale, exchange, etc.) without permission of the copyright owner.

More information about intellectual property and copyright can be found here:

https://www.library.georgetown.edu/copyright.

More information about computer acceptable use policy and intellectual property can be found here:

https://security.georgetown.edu/it-policies-procedures/computer-systems-aup