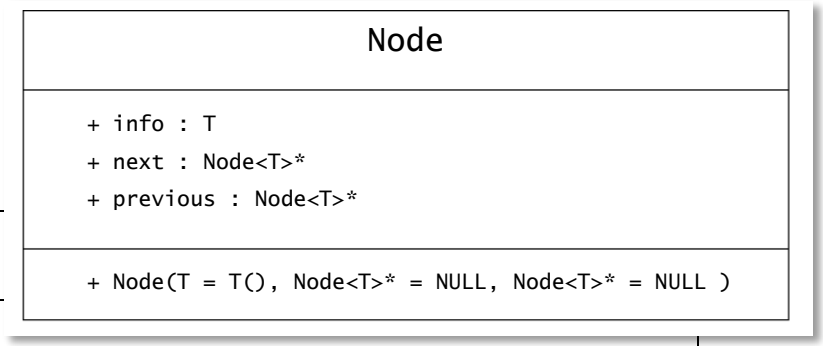
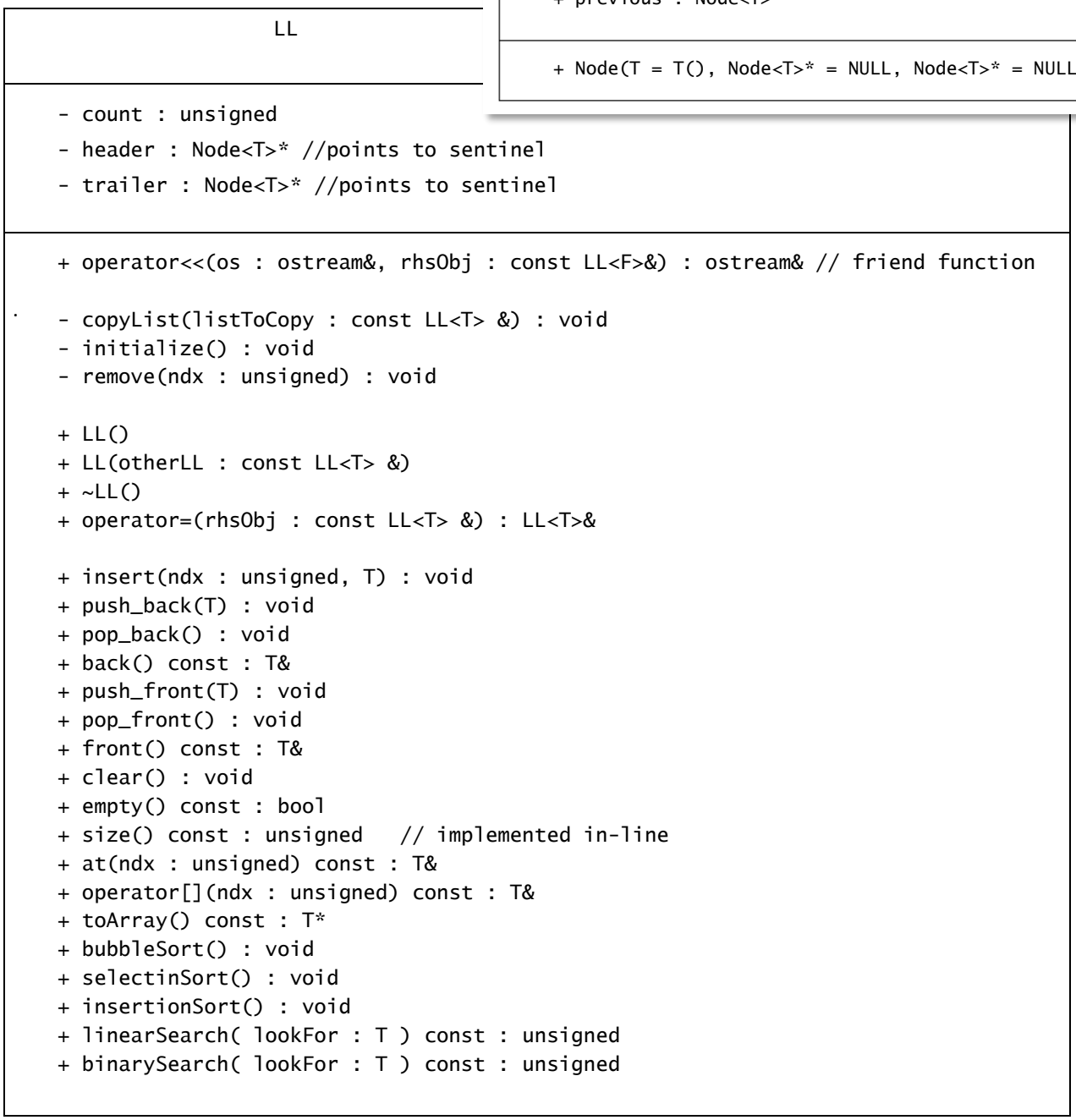
## Background

In previous projects we have been making use of the vector template class, which is a C++ container class. For this project we will develop our own container class implemented as a doubly-linked list with sentinel nodes. Unified Modeling Language (UML) diagrams below document the class designs.



Computer scientists and software engineers commonly use UML to design software and to communicate those designs. We will discuss the diagram in class, but its interpretation should be relatively straightforward. A rectangle represents a class and is divided into three sections for the class' name, its data members (or attributes or properties), and its methods (or member functions). Characters precede attributes and methods. The hyphen (-) denotes private, the pound sign (#) denotes protected, and the plus sign (+) denotes public. Many additional details and validation rules for this project's classes are also available at: http://people.cs.georgetown.edu/~addison/projects/fall2021/p3docs/index.html

## Getting Started

As the diagram indicates, you should implement the Node and LL classes. These are template classes. For convenience, several files are attached to the Project 3 Assignment on Canvas to help get you started. You should create a directory on your server account named p3 and copy all of these files to that directory. Once that is done, the p3 directory, will contain a `Makefile` and `.h` files for the project. If you are developing via an IDE, then do the same thing on your local computer and then add the `.h` files to your IDE project. I strongly recommend that you immediately write function stubs for **all** methods of **all** classes. Thereafter, use stepwise refinement and incremental development. Additionally, I recommend that you consider implementing the Node class and LL class as non-template classes first. Once you have the basic container class functionality working, you can convert Node and LL to template classes. Finally, leave the searching and sorting member functions until last after we have discussed those topics in class.

## 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 #3
 *
 *  Due on: OCT 28, 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

*What to submit:* One compressed file containing all source code and any other files associated with this project. The file name should be `submit.zip`. Except for `LL.h`, you must separate your class specification details from your class implementation details. The `LL.h` file is an exception since it contains template classes. Therefore, you must prepare header files (`<filename>.h`) and associated implementation files (`<filename.cpp>`). Ensure that your `.h` files contain sufficient comments for each data member and class method. Additionally, you must provide another `.cpp` file that contains function `main()`, along with its associated `.h` file. This "driver" program is where class objects are instantiated and functionality of the software is demonstrated. Use these names with **spelling and capitalization exactly as shown**:

```
Resources.h, Resources.cpp (reuse from P2)
Exceptions.h, Exceptions.cpp
LL.h
```

```
main.h
main.cpp
Makefile
```

*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 P3]$ make clean
rm -f *.o core a.out
[waw23@cs-class-1 P3]$ make submit
rm -f submit.zip
zip submit.zip main.cpp main.h LL.h Excepti[illegible]cpp Exceptions.h Resources.cpp Resources.h
Makefile
  adding: main.cpp (deflated 76%)
  adding: main.h (deflated 44%)
  adding: LL.h (deflated 81%)
  adding: Exceptions.cpp (deflated 63%)
  adding: Exceptions.h (deflated 73%)
  adding: Resources.cpp (deflated 83%)
  adding: Resources.h (deflated 72%)
  adding: Makefile (deflated 60%)
[waw23@cs-class-1 P3]$ unzip -l submit.zip
Archive:  submit.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
     8723  10-11-2021 01:56   main.cpp
      746  10-11-2021 01:56   main.h
    42073  10-11-2021 01:56   LL.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
      738  10-11-2021 02:11   Makefile
---------                     -------
   109960                     8 files
[waw23@cs-class-1 P3]$
```

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

*Due date/time:* 28 October 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.

You will submit Project 3 to Canvas in the same way you did Project 2. Several 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 3.

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 `cs-class` before submitting.

## 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.

## 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`. Instantiate an object of class `LL<char>`. Use `push_back`, `push_front`, and `insert` methods to populate the linked list. Test the search and sort methods to ensure that they function correctly. Of course, you should also test all other class methods before submitting your project.

## 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*