Background

You are a new programmer with Piedmont Accident Analysts, Inc. (PAAI). The Department of Transportation (DOT) Pipeline and Hazardous Materials Safety Administration (PHMSA) awarded PAAI a valuable data analysis contract. PHMSA guidelines state that "Each operator of a hazardous liquid pipeline system shall file Form PHMSA F 7000-1 for an accident that meets the criteria in 49 CFR §195.50 as soon as practicable but not more than 30 days after discovery of the accident. Requirements for submitting reports are in §195.54 and §195.58." This process involves a considerable amount of manual data collection, significant duplication of effort, and redundant data entry. PAAI is tasked to develop object-oriented software that will modernize and streamline PHMSA's reporting processes. UML diagrams for the classes that must be implemented can be found at the end of this document. Many additional details and validation rules are also available at:
http://people.cs.georgetown.edu/~addison/projects/fall2019/p1docs/index.html

To begin, we will need to load a subset of the Form PHMSA F 7000-1 data values from an existing file. We will begin the development project using this subset of values to better manage the scope of the effort. This subset of data elements is provided in an Appendix.

All data are stored in provided data files. Data files of two different formats must be accommodated by the software. One format is a binary file that is not viewable, the other is a text file that is viewable. File contents are identical except that in the text file any strings that could contain blank spaces are enclosed in double quotation marks. This ensures that the start and end of the string can be definitively located. When processing a binary data file, its contents shall first be read into an instance of the `HazMatData struct`. Then `HazMatData struct` can be passed to the convert constructor of the `HazMat7k` class. When processing a text data file, its contents shall be read directly into an instance of the `HazMat7k` class.

The complete path and file name of the input data file must be passed to your program as a command line argument. Additionally, a code indicating the data file type must be passed as a second command line argument. Valid file type codes are:

b – binary
q – text file, double quote enclosed strings

The software shall accept either upper case or lower case file type codes.

Requirements

To begin, you should implement a `Date` class. There are three date values and one time value within each record of input data. The `Date` class is especially important and includes several overloaded operators that must be correctly implemented. The complete `Time` class is provided for you to use in your program without attribution. Also provided is extensive test code for the `Time` class. You are encouraged to study this code as an example of how to implement your own unit testing. Once your `Date` class is complete, you should next implement the `HazMat7k` class.

The `HazMatData struct` has a data member to store each data element from the binary input file. Correspondingly, the `HazMat7k` class shall include a data member to store the value of each data member of the `HazMatData struct`. There will be some type conversions required as the `HazMatData struct` uses `c-strings` and the `HazMat7k` class uses strings.

Finally, we will need an `IncidentLog` class. The `IncidentLog` class shall include a vector data member to store `HazMat7k` objects.

Once all classes have been fully implemented and tested; you must demonstrate the functionality of your software. This shall be done by instantiating an object of the `IncidentLog` class within function main. Call the `read` method of the `IncidentLog` object passing the path and name of the input data file and the file type code as function arguments. This method shall iterate through the input data file. Regardless of file type, the result of reading each record of data shall be a `HazMat7k` object storing values from that record of data (assuming all validation tests pass). The resulting `HazMat7k` object shall be appended to the vector data member of the `IncidentLog` object. Once all file contents have been processed you must invoke the `displayReport` member function of the `IncidentLog` object. The result shall be a report output to the terminal screen having the following format (long narratives have been truncated below, all text should be output in your program):

```
Form PHMSA F 7000-1 Accident Report - Hazardous Liquid Pipeline Systems (2577) records:

Report Number and Date:       20120098  2012/03/30
Local Date and Time:          2012/03/30  02:04
Number of Injuries:       3
Number of Fatalities:     2
Narrative Length:      1208
Narrative:  ON MARCH 3, AT 2:04 AM, THE SCADA ALARMS AT THE CONTROL CENTER IDENTIFIED A DROP IN
PRESSURE AT MP 455.71.  LINES 14 AND 64 WERE IMMEDIATELY SHUT DOWN IN ORDER TO INVESTIGATE.  AT
APPROXIMATELY 2:20 AM, THE CONTROL CENTER RECEIVED A CALL FROM THE WILL COUNTY SHERIFF'S OFFICE
NOTIFYING ENBRIDGE OF A VEHICLE FIRE IN THE AREA OF LARAWAY ROAD AND SCHOOL HOUSE ROAD IN NEW …


Report Number and Date:       20120072  2012/03/14
Local Date and Time:          2012/03/14  18:29
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:      1163
Narrative:  AT 18:15 HRS ON MARCH, 2, 2012 DURING A REVIEW OF THE HOURLY TANK REPORTS, THE
OPERATIONS CONTROLLER IDENTIFIED A SHORTAGE ON THE WTS TANK VOLUME REPORT FOR WEST FULLERTON
STATION.  THE OPERATIONS CONTROLLER FOLLOWING ESTABLISHED PROCEDURES NOTIFIED FIELD PERSONNEL  AT
18:29 HRS.  THE FIELD PERSONNEL WERE SENT TO THE FACILITY TO INVESTIGATE.        AT APPROXIMATELY…


Report Number and Date:       20120096  2012/03/29
Local Date and Time:          2012/03/29  10:00
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       201
Narrative:  PINHOLE IN LOW POINT OF PIPE DUE TO INTERNAL CORROSION ON THE LIBERTY TO HULL 8 LINE
SEGMENT. LINE SEGMENT REPAIRED IN CONJUNCTION WITH A PREVIOUS RELEASE BY INSTALLING APPROXIMATELY
200 FEET OF PIPE.


Report Number and Date:       20120094  2012/03/28
Local Date and Time:          2012/03/28  08:45
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       239
Narrative:  7 GALLON HEAVY GAS OIL SPILL DUE TO LEAK FROM THE 1-6 LINE DUE TO EXTERNAL CORROSION.
THE LEAK WAS CONTAINED AND THE AREA WAS REMEDIATED. THE PIN HOLE IN THE LINE WAS CLAMPED WITH A
WELDED INCLOSURE. THIS WAS THE FINAL REPAIR TO THE LINE.
```

Additionally, you should thoroughly test all other methods of all classes in the project. All of those methods will be evaluated when your project is graded.

Programming Skills

The programming skills required to complete this assignment include:

- **Exception Handling**
- **Binary file input**
- **Text file input**
- **Information hiding**
- **Object oriented design**
- **Class composition**
- **Function overloading**
- **Unit Testing**
- **Operator overloading**

How to approach this program

For this project several milestones are recommended, however you are NOT required to turn anything in. Always make sure that your code compiles and runs, **on the class server**, before starting the next milestone and consider making a backup.

Milestone 1 – NLT September 3rd

- Create an empty project: add a skeleton for each source code file, these files need not contain much; the required comments, header guards, and preprocessor directives would be fine at this point
- Add the `main.h` and `main.cpp` files, write a skeleton of function `main()` (minimum code should be preprocessor directives, maybe a "bread crumb" or two, and `return 0;`)
- Copy and paste the given code for `Resources.h` and `Resrouces.cpp` into the appropriate files
- Write function stubs for all member functions of the `Date` class

Milestone 2 – NLT September 5th

- Copy and paste the given code for `PHMSA7000.h` and `PHMSA7000.cpp` into the appropriate files
- Add the `IncidentLog.h` and `IncidentLog.cpp` files
- Write function stubs for all member functions of all classes

Milestone 3 – NLT September 10th

- Implement all member and friend functions for the `Date` class
- Complete unit testing of `Date` class

Milestone 4 – NLT September 12th

- Implement all member and non-member functions of the `HazMat7k` class
- Complete unit testing of `HazMat7k` class (include test of convert constructor to instantiate a `HazMat7k` object and initialize all data members from the data members of a `HazMatData struct`)

Milestone 5 – NLT September 17th

- Implement the `read` methods of the `IncidentLog` class and test reading all input file data
- Implement all remaining member and non-member functions of the `IncidentLog` class
- Complete unit testing of `IncidentLog` class

Milestone 5 – NLT September 19th

- Complete integration testing of entire program
- Copy the input data file and all project source code files to your class server account
- Compile and run your project on the server
- Correct any issues

Submission Details

What to submit: Submit to Canvas one compressed file containing all source code and any other files associated with this project. The file name should be `submit.zip`.
You must separate your class specification details from your class implementation details. Therefore, you must prepare a header file (`<filename>.h`) and an implementation file (`<filename.cpp>`) for each set of related classes. 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 **exactly** the following file names (**with spelling and capitalization exactly as shown**):

```
Makefile
Resources.h, Resources.cpp
PHMSA7000.h, PHMSA7000.cpp
IncidentLog.h, IncidentLog.cpp
main.h, main.cpp
```

Due date/time: 19 September 2019, 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.

Creating `submit.zip`*:* **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.

```
cs-class% make clean
rm -f *.o core a.out
cs-class% ls
IncidentLog.cpp  Makefile     PHMSA7000.h    Resources.h  main.h
IncidentLog.h   PHMSA7000.cpp Resources.cpp  main.cpp
cs-class% make submit
rm -f submit.zip
zip submit.zip main.cpp main.h IncidentLog.cpp IncidentLog.h PHMSA7000.cpp PHMSA7000.h
Resources.cpp Resources.h Makefile
  adding: main.cpp (deflated 92%)
  adding: main.h (deflated 47%)
  adding: IncidentLog.cpp (deflated 78%)
  adding: IncidentLog.h (deflated 60%)
  adding: PHMSA7000.cpp (deflated 77%)
  adding: PHMSA7000.h (deflated 77%)
  adding: Resources.cpp (deflated 83%)
  adding: Resources.h (deflated 73%)
  adding: Makefile (deflated 65%)
cs-class% ls
IncidentLog.cpp  Makefile     PHMSA7000.h    Resources.h  main.h
IncidentLog.h   PHMSA7000.cpp Resources.cpp  main.cpp     submit.zip
cs-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 at the start of your program:

```
/*
 * <FileName>.<file extension>
*
 *  COSC 052 Fall 2019
 *  Project #1
 *
 *  Due on: SEP 19, 2019
 *  Author: <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.
 */
```

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 common deductions, as well as, a detailed rubric of points.

	Rubric Points	100.00
1	Compiles on Server	*5.00*
2	Code Quality	*10.00*
3	Date class	*25.00*
4	HazMat7k class	*25.00*
5	IncidentLog class	25.00
6	Driver	*10.00*

Common Deductions	
Program does not compile ON THE CLASS SERVER (deduction varies depending on how bad, value listed is max)	-60.00
Program compiles but has warnings ON THE CLASS SERVER (deduction varies depending on how bad, value listed is max)	-60.00
Program crashes during execution ON THE CLASS SERVER (deduction varies depending on how bad, value listed is max)	-60.00
Inline function with more than one statement (-10 for each occurrence up to the max deduction listed at the right)	-100.00
Filenames do not follow conventions specified (deduction varies depending, value listed is max)	-60.00
Uses abnormal exit (-10 for each occurrence up to the max deduction listed at the right)	-60.00
Uses any global variables	-60.00
Required comments and honor statement not included at start of file exactly as specified	-60.00
Late penalty for each 15 minutes late	-2.50

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