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. Additional details and validation rules are also available at:

http://people.cs.georgetown.edu/~addison/projects/fall2021/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 text data files. File contents include dates, numbers, strings with no blank spaces, and strings with blank spaces. 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 text data file, its contents shall be read directly into an instance of the `HazMat7k` class using the class' overloaded stream extraction operator.

The complete path and file name of the input data file shall be passed to your program as a command line argument.

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 and fully tested, you should next implement the `HazMat7k` class.

The `HazMat7k` class shall include a data member to store the value of each data element from the input data file.

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 as an argument. This method shall iterate through the input data file. 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. Rows of data that fail validation tests do not cause the program to end, but that row shall be skipped and not appended to the vector. 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 (181) records:

Report Number and Date: 20160238  2016/07/29
Local Date and Time:            2016/07/02  06:50
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       352
Narrative:  ON JULY 2, 2016, A WINK STATION PIPELINE OPERATOR DISCOVERED A RELEASE OF CRUDE OIL AT
THE TANK 8 SUCTION LINE. THE CRUDE OIL WAS CONTAINED WITHIN THE TANK BERM. THE STANDING OIL WAS
REMOVED, AND PLACED BACK INTO ANOTHER TANK. AFFECTED SOIL IS BEING REMEDIATED BY INTERNAL LAND FARM
ON-SITE. A B-SLEEVE WAS INSTALLED ON THE SECTION OF PIPE THAT LEAKED.

Report Number and Date: 20160233  2016/07/11
Local Date and Time:            2016/06/27  11:00
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       744
Narrative:  DURING A ROUTINE FACILITY INSPECTION AT THE LEBANON TERMINAL, LOCAL OPERATIONS
PERSONNEL DISCOVERED A SMALL DRIP OF GASOLINE COMING FROM A WEEP HOLE ON A NOZZLE PAD PLATE ON TANK
3301. THE RELEASE WAS DISCOVERED AT 11:00 AM CST ON JUNE 27, 2016. AT 2:00 PM CST, IT WAS
DETERMINED THAT THE LIKELY COST OF REPAIRS WOULD EXCEED $50,000. THE NRC WAS…

Report Number and Date: 20160227  2016/07/19
Local Date and Time:            2016/06/27  09:45
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       599
Narrative:  ON JUNE 27, AN ENTERPRISE OPERATOR IDENTIFIED PRODUCT STAINING ON THE FOUNDATION OF
TANK 107 AT THE BEAUMONT MARINE EAST TERMINAL. FOLLOWING FURTHER EXAMINATION, IT WAS DETERMINED AT
10:50 AM CST ON JUNE 28 THAT THE SOURCE OF THE PRODUCT WAS LIKELY A LEAK IN THE TANK AND REPAIR
COSTS WOULD EXCEED $50,000. AN NRC CALL WAS….

Report Number and Date: 20160236  2016/07/28
Local Date and Time:            2016/06/25  09:27
Number of Injuries:       0
Number of Fatalities:     0
Narrative Length:       682
Narrative:  NUSTAR PIPELINE CONTROL CENTER WAS NOTIFIED OF VISIBLE VAPORS ABOVE THE GROUND NEAR
VALVE SITE 5-10 IN MARION COUNTY, MISSOURI. NUSTAR EMPLOYEE ARRIVED ON SITE AND CONFIRMED LEAK.
WORK CREW ARRIVED ON SITE, EXCAVATED AND FOUND BURIED STOPPLE TEE LEAKING FROM SIDE SEAM WELD.
PLANS WERE THEN MADE TO WELD ON ADDITIONAL STOPPLE TEES…
```

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**
- **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

- 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

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

Milestone 3

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

Milestone 4

- Implement all member and non-member functions of the `HazMat7k` class
- Complete unit testing of `HazMat7k` class

Milestone 5

- 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

- 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: 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 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
PHMSA7000.h, PHMSA7000.cpp
IncidentLog.h, IncidentLog.cpp
```

```
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 P1]$ make clean
rm -f *.o core a.out
[waw23@cs-class-1 P1]$ 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 66%)
  adding: main.h (deflated 48%)
  adding: IncidentLog.cpp (deflated 77%)
  adding: IncidentLog.h (deflated 64%)
  adding: PHMSA7000.cpp (deflated 75%)
  adding: PHMSA7000.h (deflated 76%)
  adding: Resources.cpp (deflated 83%)
  adding: Resources.h (deflated 72%)
  adding: Makefile (deflated 63%)
[waw23@cs-class-1 P1]$ unzip -l submit.zip
Archive:  submit.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
     5404  08-17-2021 04:28   main.cpp
      915  08-17-2021 04:28   main.h
    18923  08-17-2021 04:28   IncidentLog.cpp
     3122  08-17-2021 04:28   IncidentLog.h
    18606  08-17-2021 04:28   PHMSA7000.cpp
    10603  08-17-2021 04:28   PHMSA7000.h
    42586  08-17-2021 04:28   Resources.cpp
     8771  08-17-2021 04:28   Resources.h
      846  08-17-2021 04:28   Makefile
---------                     -------
   109776                     9 files
[waw23@cs-class-1 P1]$
```

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: 16 September, 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.

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 2021
 *  Project #1
 *
 *  Due on: SEP 16, 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.
 */
```

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. Grade rubric published separately.

Appendix - Form PHMSA F 7000-1 Data Fields

(Note this is just the subset of data fields that we will be using for Project 1)

array index	Form PHMSA F 7000-1 field name	class data member name	class Type	struct data member name (disregard this column)	data type (ignore)	maximum number of characters	Default Values
0	REPORT_RECEIVED_DATE	reportReceivedDate	Date	report_received_date	Date	n/a	all minimums
1	REPORT_NUMBER	reportNumber	string	report_number	char	10	valid number
2	SUPPLEMENTAL_NUMBER	supplementalNumber	string	supplemental_number	char	7	empty string
3	REPORT_TYPE	reportType	string w/spaces	report_type	char	20	valid type
4	OPERATOR_ID	operatorID	string	operator_id	char	7	empty string
5	NAME	name	string w/spaces	name	char	55	empty string
6	OPERATOR_STREET_ADDRESS	operatorStreetAddress	string w/spaces	operator_street_address	char	57	empty string
7	OPERATOR_CITY_NAME	operatorCityName	string w/spaces	operator_city_name	char	18	empty string
8	OPERATOR_STATE_ABBREVIATION	operatorStateAbbreviation	string	operator_state_abbreviation	char	4	empty string
9	OPERATOR_POSTAL_CODE	operatorPostalCode	string w/spaces	operator_postal_code	char	12	empty string
10	LOCAL_DATE	localDate	Date	local_date	Date	n/a	all minimums
11	LOCAL_TIME	localTime	Time	local_time	Time	n/a	all minimums
12	COMMODITY_RELEASED_TYPE	commodityReleasedType	string w/spaces	commodity_released_type	char	84	empty string
13	UNINTENTIONAL_RELEASE_BBLS	unintentionalReleaseBbls	double	unintentional_release_bbls	double	n/a	0.0
14	INTENTIONAL_RELEASE_BBLS	intentionalReleaseBbls	double	intentional_release_bbls	double	n/a	0.0
15	RECOVERED_BBLS	recoveredBbls	double	recovered_bbls	double	n/a	0.0
16	FATAL	fatal	int	fatal	int	n/a	0
17	INJURE	injure	int	injure	int	n/a	0
18	IGNITE_IND	igniteInd	string	ignite_ind	char	5	empty string
19	EXPLODE_IND	explodeInd	string	explode_ind	char	5	empty string
20	PREPARED_DATE	preparedDate	Date	prepared_date	Date	n/a	all minimums
21	AUTHORIZER_NAME	authorizerName	string w/spaces	authorizer_name	char	45	empty string
22	AUTHORIZER_EMAIL	authorizerEmail	string w/spaces	authorizer_email	char	42	empty string
23	NARRATIVE	narrative	string w/spaces	narrative	char	4000	empty string

NOTE: column 10 and column 11 are one field in the Form PHMSA F 7000-1, they are separated here and in the data files for convenience

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