HW: Temperature Queries (Part 1)

Points

Points	
85	Working Program
15	Code Review
100	**TOTAL**

Note: If Mimir reports less than 100 total points for this assignment, we will adjust your final grade to be out of 100.

Submission

1. **Source Code**: Submit the source code (**LinkedList.h, LinkedList.cpp, Node.h, TemperatureData.h, TemperatureData.cpp, TemperatureDatabase.h, TemperatureDatabase.cpp, and main.cpp** files) to Mimir

Specifications

- Part 1: You will implement a linked list to organize temperature sensor readings from a file.
- Part 2: Process a file with queries to report based on the data.

Design (optional)

1. Algorithm for inserting new nodes into the correct location.
2. Algorithm for determining if one set of data (i.e. location, date) is less than another set of data.
3. Additional Components: Test cases

Program

You will implement a program that receives as input two files:

1. **A temperature file (e.g., temps.dat)**

   This file contains historic average temperatures for several locations in Texas. The data has been obtained from the United States Historical Climate Network (USCHN). Each line of the file contains a location, a date, and the average temperature (in Celsius) for that location and date. A **string**, corresponding to a meteorological station id, represents a location. For example, “411048” (string not a number) corresponds to Brenham, Texas. A date is specified by two integers: a year and a month. A sample line in the temperature file would be:

   ```
   410120 1893 2 6.41
   ```

   The temperature value -99.99 is used by the UCHN agency to represent missing information, for example:

   ```
   410120 1893 1 -99.99
   ```

   Valid temperatures are assumed to be in the interval -50.0 to 50.0 (remember that they use Celsius, so this is a good range). The first valid year is 1800. The latest valid year should be our current year. You can declare a constant in your program to specify the current year. (If we were programming this to be used for real, we would instead obtain the current year from the system, so that no code changes are required for future years.)

   - temps.dat might look like this:

     ```
     411048 2015 1 9.58
     411048 2015 3 14.82
     411048 2016 4 20.51
     ```

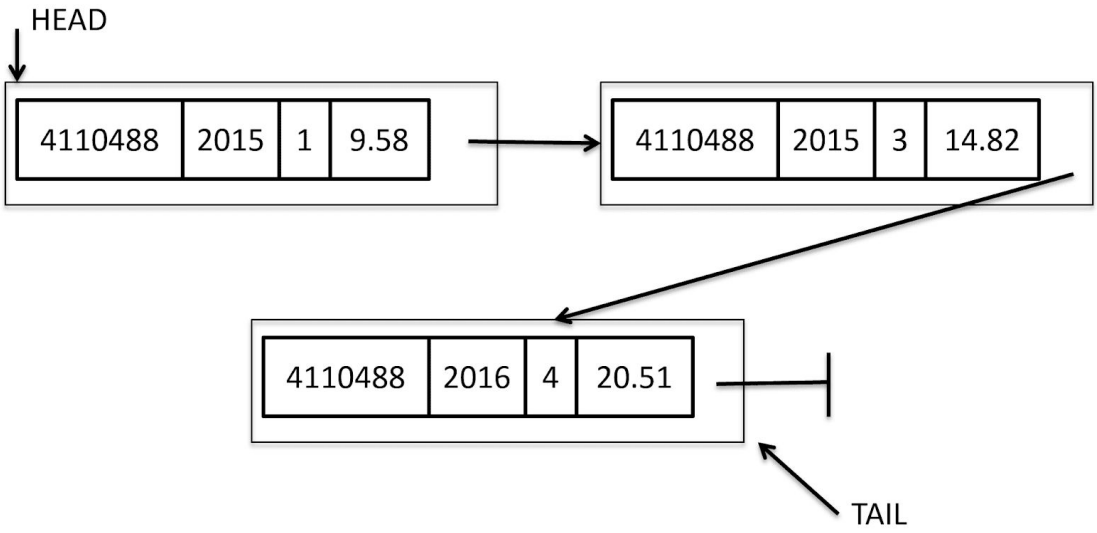
```
411048 2016 1 10.99
411000 1973 1 0.54
411048 2016 3 18.40
411048 2016 5 -99.99
```

2. **A query file (e.g., queries.dat)**

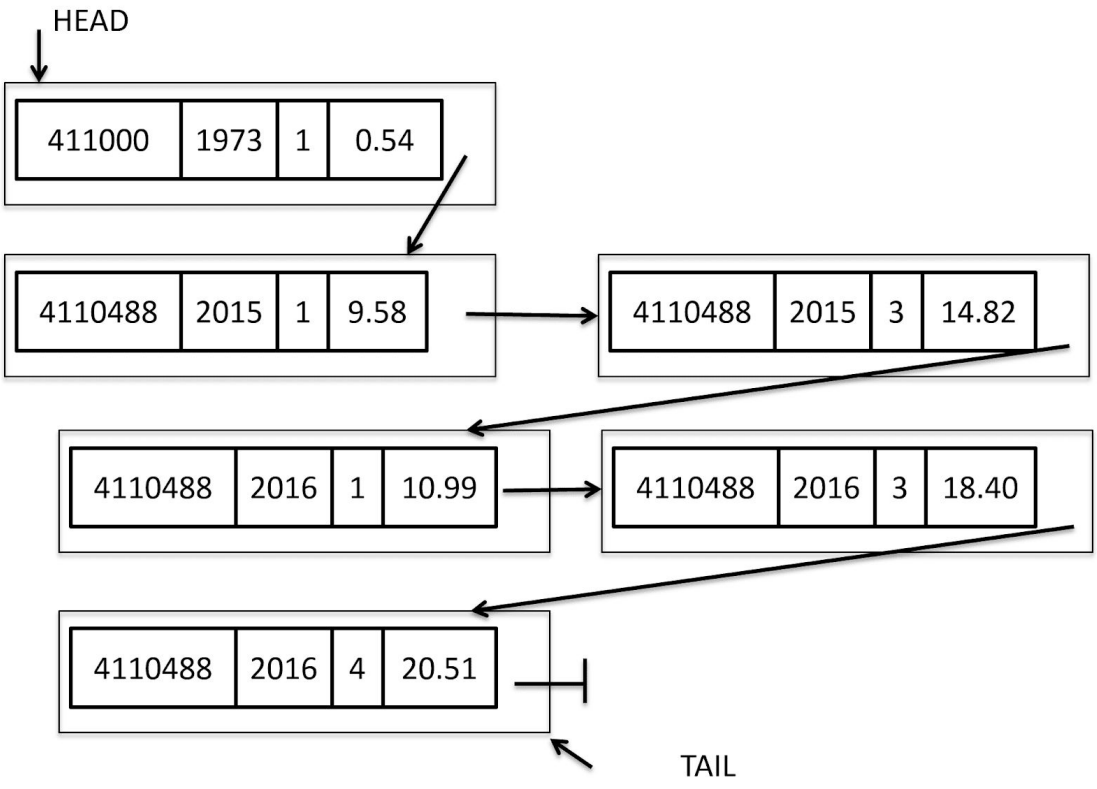
   Details will be given with part 2.

Requirements

- You must use this skeleton code to create your program.
- Your implementation has to use a linked list to store the data you read from the temperature file.
    - You *must* implement a function named “getHead()” in the class LinkedList. This function is supposed to return the pointer to the start Node of the list.
    - You *must* implement an overloaded `operator<` for `struct Node` and you must use this operator to to compare `Node`s while inserting into the `LinkedList`. The nodes in the linked list are:
        - First ordered by location
        - Then by date *(i.e. by year and then by month)*
    - For example, for the sample temps.dat above, after reading 3 lines of the file your linked list looks like:



After reading all 7 lines from the file, it looks like:



Notice that we ignored the entry with value -99.99.

- Your program should receive the names of the input files as command line arguments. The first argument will be the temperature file, the second the queries file. We'll use the queries file next week.
- If the input files contain a line with an invalid format, you should output on the console an error message and finish executing the program.
    - Do not throw an exception.
    - Output the message in the console beginning "Error: " followed by the description shown below followed by the invalid value. For example:
        - Error: Invalid temperature -1221.11
        - Error: Invalid year 2020
        - Error: Invalid month 0
        - Error: Unable to open input.dat
        - Error: Other invalid input
    - Examples of lines with an invalid format for temps.dat:

```
4111000 1889 0 -4.3
4111000 1889 18 -4.3
4111000 2016 01 -1222.11
4111000 1889 .8 8.3
4111000 2015 11
```

- You are required to use classes and comply with the "Rule of 3" (see zyBook and slides). More specifically, you need to implement the constructor, destructor, copy assignment, and copy constructor for the LinkedList class and any other class that uses the freestore, but I don't think any other class should use the heap.
    - The constructor for your Node and TemperatureData classes should take all data items (station id, year, month, average temperature value) as

parameters. Use initialization in the “membername(value)” format (see zyBook and slides)
    - We’ll test these directly.
- Implement the print() function for the linked list. Use this function and/or the overloaded output operator to test your results. We’ll test the linked list functions directly for part 1. The expectation is that the list will be printed out in the correct order based on the comparison rules listed earlier.
    - The output format is the same as the format in the input file. Id, year, month, and temperature separated by a space.
    - When you output, it will look like the input file, with each item on one line. However, the order should be sorted according to the requirements above.

Grading

1. We will unit test the LinkedList class, including insert, print, constructors, and rule of 3, and overloaded output operator (which is already done for you if you get print to work.
2. We will unit test Node and TemperatureData classes.
3. We will unit test TemperatureDatabase loadData() function. We will look for error messages.

Sample Input/Output Files

- Building the list input files
    - Zip file
        - temp-3lines.dat, temp-7lines.dat, etc.

Hints & Comments

- Write a myTest() function for the TemperatureDatabase. You can call that function from main after loading the Database to test things like the rule of three, print(), etc.
- Use the print() function to print out your linked list so that you can make sure that you are maintaining it in order as required. The first test with the small sample input files that we provide (e.g., first the one with 3 lines, then the one with 7 lines of data.), output your linked list and inspect it to see if it is representing the input data correctly.
- It is often a good idea to develop your solution incrementally, completing and testing components of your overall program as you develop them.