# Temperature Queries (Part 1)

## Points

| **Points** | | **Due** |
|---|---|---|
| 10 | Design | April14, 2017 |
| 90 | Working Classes | April 19, 2017 |
| **100** | **TOTAL** | |

## Submission

1. **Design**: Submit the PDF to eCampus.
2. **Source Code**: Submit the source code (**linkedlist.h, linkedlist.cpp, node.h, temperaturedb.h, temperaturedb.cpp, and main.cpp** files) to Vocareum

## Specifications

In parts 1 and 2 you will use a linked list to organize temperature sensor readings from a file. Another file will provide queries to report based on the data.

## Design

1. Algorithm for inserting a node in the correct location
2. Test cases

## Program

For part 1, you will build a program that receives as input two files:

- **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 Historic 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 containing the following:

  - A meteorological station id that is an integer, represents a location. For example, “411048” 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.)

- **A query file (e.g., queries.dat)**
  We'll address this file in part 2. However, for testing just enter any value in the command line for this file.

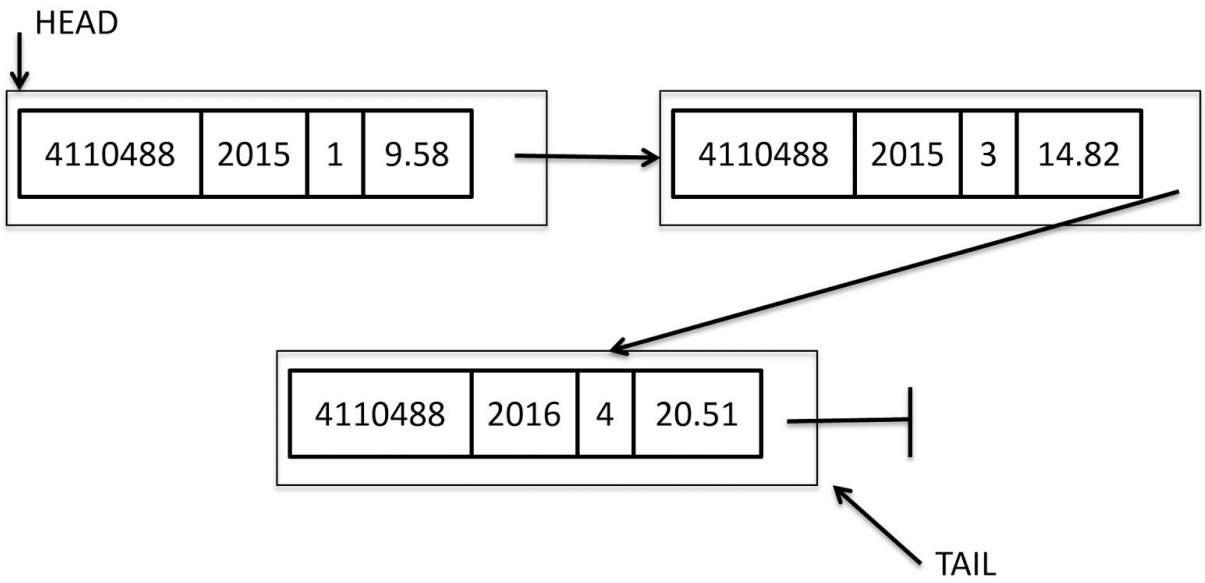
Your program should read these two files and generate a file results.dat with the queries and their results. More on processing the queries in the next homework. Regardless, temps.dat might look like this:

```
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
```

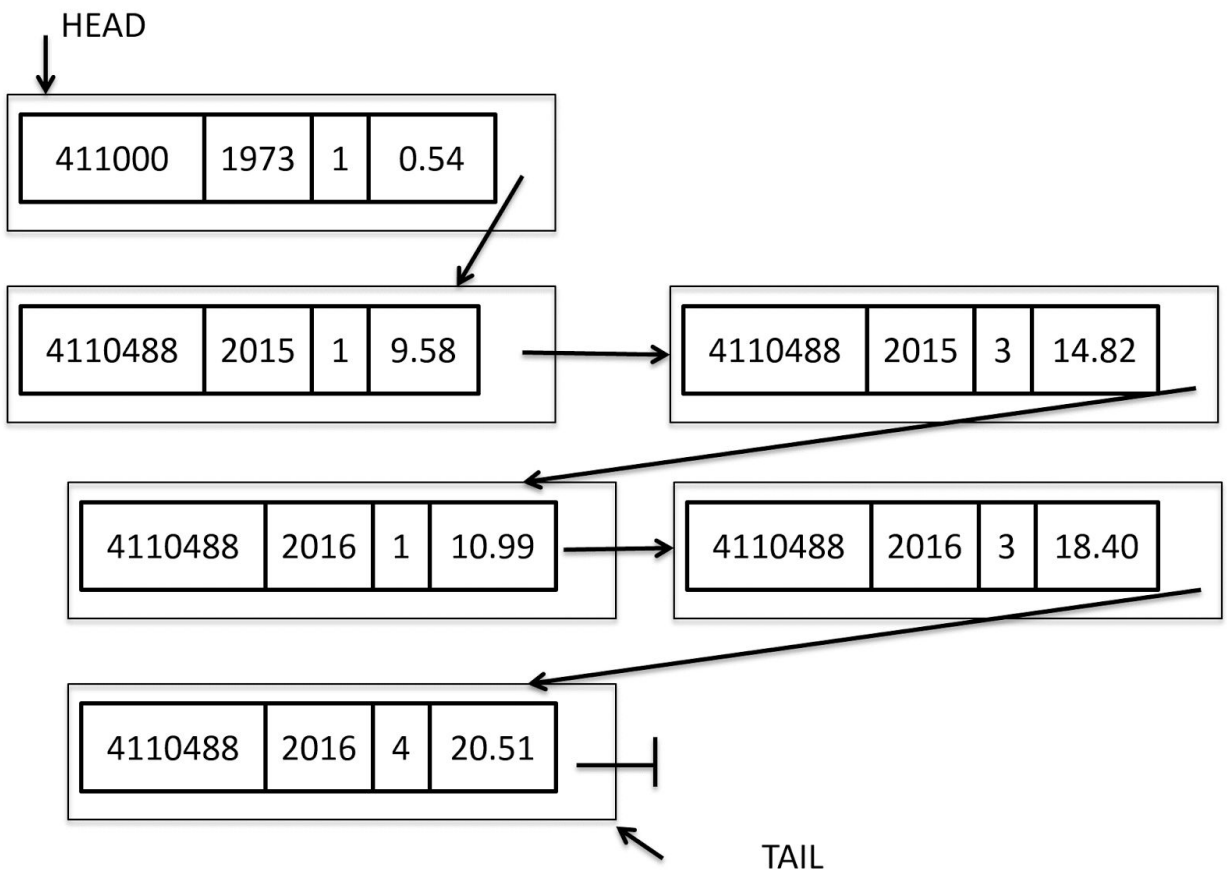
For now you will output the ordered temperature information in the results.dat file.

# Requirements

1. Use this skeleton code to create your program.
2. Your implementation has to use a linked list to store the data you read from the temperature file. As you read the data from the file, you need to insert it into your linked list such that the list is ordered by location first and 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.

3. Your program should receive the names of the input files as command line arguments. The output file created by your program should be named results.dat.
4. If the input files contain a line with an invalid format, you should output on the console an error message and finish the execution. You do not need to specify which error occurred. Just make sure that the string “Error” appears in the console. You can be more specific about the error if you want (it may be useful for your own debugging, in case you are making a mistake when validating the input data.)
    - Examples of lines with invalid format for temps.dat:

      ```
      4111000 1889 0 -4.3
      ```

```
4111000 1889 18 -4.3
4111000 2016 01 -1222.11
4111000 1889 8.3
```

5. You are required to use classes. By the end of the next homework (part 2), you must comply with the “Rule of 3”. For this part you need to implement the constructor and destructor. The constructor for your class representing temperature values should take all data items (station id, year, month, average temperature value) as parameters. Use initialization in the “membername(value)” format (zybook section 20.6)
6. Implement the overloaded output operator for the linked list and write its content to results.dat after reading in all values.

# Sample Input Files

The following zip file has sample data files: tempQueryFiles1.zip

- Data: temp-3lines.dat, temp-7lines.dat, temp-70s.dat, temp-2000s.dat

# Hints & Comments

- You may want to print out your linked list so that you can make sure that you are maintaining it in order as required. 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;
- When you round a double value to calculate the mode, make sure that you are not just truncating the value: the rounded value for 10.6 should be 11, not 10;
- It is often a good idea to develop your solution incrementally, completing and testing components of your overall program. For example, you may

want to program and test the part where you read the temperature input file first, before starting to write the code to read the query file. This will help you to avoid making the same mistake in different parts of your code;

- If you tried your solution with very large input files, you may have noticed that your solution takes some time to provide the answers. Students in the Computer science and Computer Engineering majors have the opportunity to take many other courses (data structures, databases, computer systems, operating systems, data mining, distributed systems, parallel computing) where they learn concepts and techniques to achieve efficient queries over large datasets.