Weather Data

You will write the class WeatherData to process 100 years of weather data from a single weather station. It should contain the following constructor:

```
public WeatherData(Scanner s)
```

This should load the data provided by the Scanner into your WeatherData class. The data should be loaded one line of text at a time using the Scanner's nextLine method. The first line of text should be discarded as it is a header. Each line of text, stored in a String, should be split using the `.split(",")` method. This will create an array of Strings. The relevant Strings are index 1, which indicates the date for the data on this line, index 2, which will indicate if the day contains multiple days of precipitation (or will be blank if it is just a single day of data), index 3, which indicates a multiple day precipitation total (or will be blank if it is a single day of precipitation), index 4, which indicates the single day precipitation total (or will be blank if the day contains multiple days of precipitation), index 5, which indicates the day's snowfall total, index 7 which indicates the day's high temperature (and may be blank if this was not recorded), and index 8 which indicates the day's low temperature (and may be blank if this was not recorded). All other indexes may be ignored.

Every index in the split array that is not blank will be surrounded by double quote characters. The substring method may help you remove them.

You may assume that each row has a valid date in the form YYYY-MM-DD and that the dates are in chronological order, but there may be missing dates. Temperatures are always whole numbers but precipitation and snowfall may contain decimal points. The `Integer.parseInt` and `Double.parseDouble` methods may help you convert the corresponding String to numeric data.

The constructor will set up data structures to support the methods below. The use of collections will help the methods run quickly. **The constructor must run in under 1 minute or you may receive no credit for your submission.** Each method will be called many times for testing. **For each of your methods, each call must execute in under 1 second or no credit will be awarded for that method.** There will be a **10% penalty** (1 or 2 points) for a given method if it is too slow (which will be tested by performing many method calls) and there will be a 2 point bonus for the whole assignment if none of your methods are slow.

The class should have the following methods, shown with their corresponding documentation and point values. The provided starter code will have empty implementations.

```
/**
 * Determine whether the given temperature was ever seen as a high temperature
 * in the data provided to the constructor. (10 points)
 *
 * (HINT: This is a membership question. What data structure have we seen that
 * can help us answer this question?)
 *
 * @param degrees Temperature (same units as data file)
 * @return true if high temp, false otherwise
 */
public boolean highTemp(int degrees)
```

```java
/**
 * Determine whether the given temperature was ever seen as a low temperature in
 * the data provided to the constructor. (10 points)
 *
 * @param degrees Temperature (same units as data file)
 * @return true if low temp, false otherwise
 */
public boolean lowTemp(int degrees)

/**
 * Determine the total amount of snowfall recorded in the given year. (20
 * points)
 *
 * (HINT: What data structure would allow us to correspond an amount of snowfall
 * with a year? How much snowfall is recorded in a year not found in the file?)
 *
 * @param year
 * @return
 */
public double totalSnowfallForYear(int year)

/**
 * Determine the average (mean) total precipitation recorded for the given
 * month. Be sure to include multi-day precipitation amounts. (Assume that all
 * of the precipitation occurs on the date of the multi-date range - never
 * divide it across months.) (20 points)
 *
 * @param month
 * @return
 */
public double averagePrecipitationForMonth(int month)

/**
 * Return the most common (most often observed) high temperature seen in the
 * given month. If there are two or more temperatures that are both seen the
 * most number of times, return the lowest high temperature. (20 points)
 *
 * @param month Month
 * @return highest most common high temperature seen in that month
 */
public int lowestMostCommonHighForMonth(int month)

/**
 * For the given low temperature, find the highest high temperature seen with
 * that low. (20 points)
 *
 * @param degrees Low temperature
 * @return Highest high ever seen for that low temperature
 */
public int highestHighForLow(int degrees)
```