**1. Temperatures**

Redo exercise 3 of lab #6 now using the following structured datatype:

struct Month {  
    string name;  
    double temperature;  
}

A value of type Month is made of two parts, the name part (a string) which is the name of the month, and the temperature part (a double) which is the average temperature for this month. You are now to use only one array holding Month values.

**2. More temperatures**

Write a program which reads a serie of temperatures (double values) and then display the list of all temperatures read. Each temperature should be displayed together with its number of occurences (the number of time the temperature has been read). The temperatures must be read from a file. Here is an example of execution of the program (user input is in bold):

Welcome to the Temperature Counter  
Enter the filename: temperatures.txt  
20.0 : 4  
22.0 : 5  
23.0 : 1  
21.5 : 3

Here is the content of the file [temperatures.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab07/temperatures.txt):  
  
20.0 22.0 23.0 20.0 21.5 22.0 20.0 21.5 20.0 22.0 21.5 22.0 22.0  
  
You are to use the following structured datatype:

struct Temperature {  
    double value;  
    int occurences;  
};

A value of type Temperature is made of two parts, the value of the temperature (like 20.0) and the occurences of the temperature (its number of occurences like 4). You are to store Temperature values inside an array. Because you don’t know in advance the number of different Temperature values to store, you must use an array of arbitrary size (big enough) and handle a variable to store the number of Temperature values inside the array. The temperatures are not to be displayed in any special ordering (like increasing order). You can use the file provided and you can assume the file is not empty.  
  
**Supporting file**

* [temperatures.txt](file:///\\home\marc\danang\2016-bis\cse142\labs\lab07\temperatures.txt)

**3. Displaying grades**

Redo exercise 6 of lab #6 now using the following structured datatype:

struct Student {  
    string name;  
    double marks[MAX\_MARK];  
};

A value of type Student is made of two parts, the name part (a string) which holds the name of the student, and the marks part (an array of double) which holds the marks of the student. You are now to use only one array holding Student values.

**4. Birthday dates**

The first part of this exercise asks you to implement a struct named date, stored in two files named date.h and date.cpp. A value of type date stores two integer values, the month and the day in that month. We don’t need to store the year in a date value. Besides your date struct, you should implement the following functions to manipulate a date. For all functions shown below, you may assume that any parameter values passed are valid:

* int getMonth(date d) : returns the month of the date d, an integer between 1 and 12. For example, if d represents August 17, this method should return 8
* int getDay(date d) : returns the day of the month of d, an integer value between 1 and the number of days in that month (which will be between 28 and 31). For example, if d represents August 17, this function should return 17.
* void setDate(date &d, int month, int day) : modifies the state of the date d to represent the given month and day. You may assume that the parameter values passed are valid.
* void setDate(date &d, date newdate) : modifies the state of the date d to represent the same month and day as in newdate. Notice that functions in C++ can be overloaded (there will be two different functions with the same name setDate)
* string toString(date d) : returns a string representation of d in a month/day format. For example, if d represents March 24, it returns ”5/24”. If this date represents December 3, it returns ”12/3”. This function returns a string and it does not print output. To write that function you will need to use an ostringstreamobject. To use that type you must include the sstream at the beginning of your files (#include <sstream>)
* bool dateEquals(date d1, date d2) : returns true when d1 and d2 represent the same date, or false otherwise.
* int daysInMonth(date d) : returns the number of days in the month of the date d (we ignore leap year so February has 28 days).
* void nextDay(date &d) : modifies the state of d by advancing it 1 day in time. For example, if d represents September 19, a call to this function should modify d’s state so that it represents September 20. Note that depending on the date, a call to this function might advance the d value into the next month. For example, the next day after August 17 is August 18; the next day after February 28 is March 1; and the next day after December 31 is January 1.

None of the functions listed above should print any output to the console. You are to write a test program to test all the previous functions.  
  
The second part of this assignment asks you to write a client program that uses the previous date structure. This program should be the final main program of the exercise. This program begins by prompting the user for today’s date and for his/her birthday, each as a month-day pair. The program uses this information to print the number of days in the month the user was born, and the number of days from today to the user’s birthday. If the user’s birthday is today, the program prints a Happy Birthday message. Below are several example logs of execution from the program:

What is today’s date (month and day)? 11 4  
What is your birthday (month and day)? 9 9  
There are 30 days in month #9  
Number of days until birthday 9/9: 309  
  
What is today’s date (month and day)? 12 15  
What is your birthday (month and day)? 12 15  
There are 31 days in month #12  
Happy birthday!  
  
What is today’s date (month and day)? 8 19  
What is your birthday (month and day)? 11 30  
There are 30 days in month #11  
Number of days until birthday 11/30: 103  
  
What is today’s date (month and day)? 10 2  
What is your birthday (month and day)? 10 1  
There are 31 days in month #10  
Number of days until birthday 10/1: 364

To figure out the number of days until the user’s birthday, represent today and the birthday as date values. By advancing one date until it reaches the other and counting, you can determine the number of days between them. Absolute days should not be used to calculate the days until the user’s birthday. You do not have to worry about leap years at all for this assignment. Assume that February always has 28 days and that the year is always 365 days long. Assume valid input (that the user will always type a month between 1-12 and a day between 1 and the end of that month).