Data Structures: 505 22240 / ESOE 2012

Computer Assignment 2: Date class construction
Due: the week after next, 9:00pm
*Hand in online via **CEIBA***

Total score: 160

This computer homework assignment is designed to help you to learn about building C++ classes and to observe the decomposition of a complicated task into simple subtasks. You have learned how to write some functions in CA1. Now, your task is to fill in the implementation of a class that manipulates dates, which is defined in a header file called "Date.h". **DO NOT** use any of the built-in operations on dates provided in any C++ library in your solution. Fill in the private variables in "Date.h" and write your functional answers in a separate file called "Date.cpp". The overall task is broken down into subtasks as stated as follows:

1. Implement the basic helper methods listed below.

```
/** Checks whether the given year is a leap year.
  * @return true if and only if the input year is a leap year.
  * Grade: 10%
  */
public bool isLeapYear(int year) {
  ...
}
```

```
/** Returns the number of days in a given month.
  * @param month is a month, numbered in the range 1...12.
  * @param year is the year in question, with no digits omitted.
  * @return the number of days in the given month.
  * Grade: 10%
  */
public int daysInMonth(int month, int year) {
  ...
}
```

```
/** Checks whether the given date is valid.
  * @return true if and only if month/day/year constitute a valid date.
  *
  * Years prior to A.D. 1 are NOT valid.
  * Grade: 20%
  */
public bool isValidDate(int month, int day, int year) {
  ...
}
```

These methods are declared as "*public*" so we can test them from another class. Note that February contains 28 days most years, but 29 days during a leap year. A leap year is any year divisible by 4, except that a year divisible by 100 is not a leap year, except that a year divisible by 400 is a leap year after all. Hence, 1800 and 1900 are not leap years, but 1600 and 2000 are.

2. Define the internal state that a "Date" object needs to have by declaring some data fields (all *private*) within the Date class. Define the basic constructor specified below. A Date should be constructed only if the date is valid. If a caller attempts to construct an invalid date, the program should halt after printing an error message of your choosing. To halt the program, include the line: exit(0);

```
/** Constructs a Date with the given month, day and year. If the date is
  * not valid, the entire program will halt with an error message.
  * @param month is a month, numbered in the range 1...12.
  * @param day is between 1 and the number of days in the given month.
  * @param year is the year in question, with no digits omitted.
  * Grade: 15%
  */
public Date(int month, int day, int year) {
  ...
}
```

```
/** Returns a string representation of this date in the form month/day/year.
  * The month, day, and year are printed in full as integers; for example,
  * 10/17/2010 or 5/11/258.
  * @return a string representation of this date.
  * Grade: 20%
  */
public string toString() {
  ...
}
```

3. Implement the following methods.

```
/** Determines whether this Date is before the Date d.
  * @return true if and only if this Date is before d.
  * Grade: 10%
  */
public bool isBefore(const Date& d) {
  ...
}
```

```
/** Determines whether this Date is after the Date d.
  * @return true if and only if this Date is after d.
  * Grade: 10%
  */
public bool isAfter(const Date& d) {
  ...
}
```

```
/** Determines whether this Date is equal to the Date d.
  * @return true if and only if this Date is the same as d.
  * Grade: 10%
  */
public bool isEqual(const Date& d) {
  ...
}
```

```
/** Returns the number of this Date in the year.
  * @return a number n in the range 1...366, inclusive, such that this Date
  * is the nth day of its year. (366 is only used for December 31 in a leap
  * year.)
  * Grade: 15%
  */
public int dayInYear() {
  ...
}

/** Determines the difference in days between d and this Date. For example,
  * if this Date is 6/16/2006 and d is 6/15/2006, the difference is 1.
  * If this Date occurs before d, the result is negative.
  * @return the difference in days between d and this Date.
  * Grade: 10%
  */
public int difference(const Date& d) {
  ...
}
```

Hint: all the methods in the Date class can read all the private fields in “any” Date object (not just "this" Date object).

4. Implement the final missing piece of your class, a second constructor that takes a string argument.

```
/** Constructs a Date object corresponding to the given string.
  * @param s should be a string of the form "month/day/year" where month must
  * be one or two digits, day must be one or two digits, and year must be
  * between 1 and 4 digits. If s does not match these requirements or is not
  * a valid date, the program halts with an error message of your choice.
  * Grade: 30%
  */
public Date(const string& s) {
  ...
}
```

Your Date constructor definitely should not accept "11/31/2005" or "12/4" or "hey Jude".

Hint: use the STL to familiarize yourself with all the methods available to you in the string class.

The files Date.h and Date.cpp contain a skeleton and TestDate.cpp provides some test code, for a Date class, respectively. We have included a main method in the TestDate.cpp that tests some of your methods. You are welcome to modify the main method as you please, perhaps to add further tests of your own. We will not grade the main method in this assignment. **DO NOT** change the prototype (interface) of any method. However, you may want to add some utilities to some functions to facilitate the coding.