CS 247 Spring 2014
Assignment 2 Question 1 --- Specifications

Q1 [70 marks] PImpl Idiom

You are to implement an *immutable* Date ADT whose data representation is hidden within a nested structure (called the 'private implementation', or 'pointer to implementation', or 'PImpl' idiom). The ADT's valid dates range from 1 January, 1900 to 31 December, 2100

Because the ADT is immutable, there are no operators for changing the value of a Date object once it is created. If a variable of type Date needs a new value, it is assigned to a new object initialized to the new value.

Objective

As per the PImpl programming idiom, the provided header file reveals nothing about the ADT's data representation except for a single reference to a `struct` that is not defined. You are to create the corresponding implementation file, named `Date.cpp`. You can add private member functions to `Date.h`, but ***you cannot add any data members or any public member functions, and you cannot declare any friends.***

Provided Code

We provide the header file (`Date.h`). It declares all of the methods to be implemented.

We also provide a main program for your solution (`DateTestHarness.cpp`). The main program is a **test harness** that you can use to test your ADT implementation by creating various dates, performing operations on those dates, and printing the dates. The test harness is not robust. (It is throwaway code.) If you enter invalid commands, it might cause the program to terminate. It should go without saying (but I'll say it anyways) that your program will be tested only on valid input commands.

Initialization

When the program starts executing, it prints a welcome message and asks for the first command from the user.

Commands

There are 9 valid commands that the test harness will recognize:

c <d#>
p <d#>
a <d#>
= <d#> <d#>
i <d#>
C <d#>
A <d#> <d#>
t
r <d#>
CTL-D

a) c <d#>

Command c initiates the creation of a new Date object assigned to variable *d#*. The user is then prompted for the day, month, and year. The ADT constructor is then called with the provided input values. If the input values result in an invalid date, the constructor throws an exception: a string-constant holding an error message to be printed.

- If the year is invalid, the exception value is "Invalid year."
- If the month is invalid, the exception value is "Invalid month."
- If the day is invalid, the exception value is "Invalid day of the month."

A year value is invalid if it is outside the date range. A month value is invalid if it is anything but the full (capitalized) name of one of the twelve months. (Yes, a friendlier ADT would accept other representations of month, but you wouldn't learn anything by coding them all.) A day value is invalid if it is outside of the date range or if it results in an invalid date (e.g., April 31).

b) p <d#>

The test harness prints the value of the Date variable *d#* (if the variable has been assigned a Date value) by invoking the streaming operator (`operator<<`). The streaming operator outputs a Date value in the following format:

```
<day><sp><monthname>,<sp><year>
```

For example, the date of the midterm is

```
26 June, 2014
```

c) a <d#>

The test harness invokes the three accessor methods on Date variable *d#* and prints the results.

d) = <d#> <d#>

The test harness calls the equality operator (`operator==`) on the two Date variables and prints the result as either `true` or `false`.

e) i <d#>

Command i invokes the three increment functions.

- If the user is prompted for the number of years to increment, the result of adding that number of years to Date variable *d#* is printed.
- If the user is prompted for the number of months to increment, the result of adding that number of months to Date variable *d#* is printed.
- If the user is prompted for the number of days to increment, the result of adding that number of days to Date variable *d#* is printed.

In all cases, a new Date value is computed and returned -- there is no change to the value of Date variable *d#*. If any of the operations computes a date whose *year* is outside of the range of legal values, then the error message "Invalid year." is printed. If the result is an invalid date within a valid year (e.g., the original date is 31 August and incrementing by 6 months produces a new date 31 February), then the value is rounded down to the nearest valid date in the given month (i.e., rounded down to 28 February -- or 29 February, if the new date is in a leap year). Assume that increments are always nonnegative.

Be mindful of the rules for leap years:

- if the year is evenly divisible by 4, then it IS a leap year, unless
- the year is also evenly divided by 100; then it is NOT a leap year, unless
- the year is also evenly divisible by 400; then it IS a leap year.

This means that 2000 is a leap year, whereas 1900 and 2100 are NOT leap years.

f) C <d#>

The test harness creates a copy of Date variable *d#* (using the copy constructor) and prints the value of the new copy as well as the value of Date variable *d#*.

g) A <d#> <d#>

The test harness uses the Date ADT's assignment operator (`operator=`) to assign the value of the second Date variable *d#* to the first Date variable. It then prints both variable values. (They should be the same!)

h) t

The test harness prints the result of calling the Date ADT's `today()` method. This method creates a new Date object using values that are returned from a system call to get the current date.

i) r <d#>

This command reads a new Date value from the input string using the streaming operator (`operator>>`) and assigns it to Date variable *d#* using the Date ADT's assignment operator (`operator=`). The input value should have the format

`<day><whitespace><monthname>,<whitespace><year>`

where `<day>` is a number between 1-31 inclusive, `<monthname>` is the full (capitalized) name of one of the twelve months, and `<year>` is a number between 1900 and 2100 inclusive. If the input does not have this format, the streaming operator should set the `failbit` of the input stream. If the input date is not a valid Date, then an exception is thrown (like the exceptions listed for the constructor).

j) CTL-D

If you hold down the control key on your keyboard while pressing the D key, you will generate an EOF (end of file) character. When the test harness's input stream sees this, it terminates the program.

Sample Execution

Below is an example partial execution. User input is shown in **bold** font.

```
Test harness for Date ADT

Command: c d0
Enter day of month (1-31): 1
Enter month (January-December): April
Enter year (1900-2100): 1900

Command: i d0
Enter number of years to increment:80
1 April, 1900 + 80 years = 1 April, 1980
Enter number of months to increment:500
1 April, 1900 + 500 months = 1 December, 1941
Enter number of days to increment:2398
1 April, 1900 + 2398 days = 25 October, 1906

Command: c d1
Enter day of month (1-31): 31
Enter month (January-December): December
Enter year (1900-2100): 2100

Command: A d0 d1
New date = 31 December, 2100, old date = 31 December, 2100

Command: p d0
Date d0 = 31 December, 2100

Command: r d1
Enter date value (day month, year): 16 May, 2300
Invalid year.

Command: r d1
Enter date value (day month, year): May 16, 2014
Invalid date value.

Command: ^D
```