Indian Institute of Technology, Kharagpur

Department of Computer Science and Engineering

Assignment 3: C++ Programming, Spring 2015-16

Software Engineering (CS 29006)

Assignment Date: 24-Feb-2016 **Submission Deadline:** 23:55 hrs, 02-Mar-2016

Instructions

- Assignments in this set should be coded in C++.
- Zero marks for a submission if it does not pass the plagiarism test or if you copied from someone in the class.

---

1. You need to develop a `Date` data type in C++ to deal with calendar dates ranging between 01/Jan/1950 and 31/Dec/2049. The dates should support the operations for a Date type as defined in the following class definition (in `Date.h`). Further dates are formatted for i/o using the `DateFormat` class.

```
#include <iostream>
#include <exception>
using namespace std;

typedef enum Day {
    D01 =  1, D02 =  2, D03 =  3, D04 =  4, D05 =  5,
    D06 =  6, D07 =  7, D08 =  8, D09 =  9, D10 = 10,
    D11 = 11, D12 = 12, D13 = 13, D14 = 14, D15 = 15,
    D16 = 16, D17 = 17, D18 = 18, D19 = 19, D20 = 20,
    D21 = 21, D22 = 22, D23 = 23, D24 = 24, D25 = 25,
    D26 = 26, D27 = 27, D28 = 28, D29 = 29, D30 = 30,
    D31 = 31 };

typedef enum WeekDay {
    Mon = 1, Tue = 2, Wed = 3, Thr = 4, Fri = 5, Sat = 6, Sun = 7 };

// A week starts on Monday and ends on Sunday. Weeks are numbered in a year from 1 to 52
// (or 53). Week 1 (W01) of a year starts contains the first Thursday of the year. Hence
// W01 of 2016 starts on 04/Jan and ends on 10/Jan as 07/Jan is the first Thursday.
// 01/Jan, 02/Jan, and 03/Jan therefore belongs to the last week of 2015.
typedef enum WeekNumber {
    W01 =  1, W02 =  2, W03 =  3, W04 =  4, W05 =  5,
    W06 =  6, W07 =  7, W08 =  8, W09 =  9, W10 = 10,
    W11 = 11, W12 = 12, W13 = 13, W14 = 14, W15 = 15,
    W16 = 16, W17 = 17, W18 = 18, W19 = 19, W20 = 20,
    W21 = 21, W22 = 22, W23 = 23, W24 = 24, W25 = 25,
    W26 = 26, W27 = 27, W28 = 28, W29 = 29, W30 = 30,
    W31 = 31, W32 = 32, W33 = 33, W34 = 34, W35 = 35,
    W36 = 36, W37 = 37, W38 = 38, W39 = 39, W40 = 40,
    W41 = 41, W42 = 42, W43 = 43, W44 = 44, W45 = 45,
    W46 = 46, W47 = 47, W48 = 48, W49 = 49, W50 = 50,
    W51 = 51, W52 = 52, W53 = 53 };

typedef enum Month {
    Jan =  1, Feb =  2, Mar =  3, Apr =  4, May =  5, Jun =  6,
    Jul =  7, Aug =  8, Sep =  9, Oct = 10, Nov = 11, Dec = 12 };

typedef unsigned int Year;
```

```
// This class defines the format in which a date is output or input
class DateFormat {
public:
    // The parameters below takes designated values with the specified format semantics
    // dateFormat:
    //     0: No date. This is valid for output only.
    //     "d": single digit date in one digit, double digit date in two digits (7, 23)
    //     "dd": all dates in two digits with single digit dates with leading 0 (07, 23)
    // monthFormat:
    //     0: each month in its full name (February, November). This is valid for output only.
    //     "m": single digit month in one digit, double digit month in two digits (2, 11)
    //     "mm": all months in two digits with single digit months with leading 0 (02, 11)
    //     "mmm": each month with first three letters of its name (Feb, Nov). This is valid for output only.
    // yearFormat:
    //     0: No year. This is valid for output only.
    //     "yy": year in last two digits (1961 as 61, 2016 as 16)
    //     "yyyy": year in four digits (1961 as 1961, 2016 as 2016)
    DateFormat(const char* dateFormat, const char* monthFormat, const char* yearFormat);

    // Single C-string format where dateFormat, monthFormat, and yearFormat are separated by '-'
    // Example: "dd-mmm-yy", "d-mm-yyyy", etc
    DateFormat(const char* format);

    // Default DateFormat created as "dd-mmm-yy"
    DateFormat();

    ~DateFormat();

private:
    char*    dateFormat;     // C-string specifying format of date
    char*    monthFormat;    // C-string specifying format of month
    char*    yearFormat;     // C-string specifying format of year
};

// This class defines a date within 01/Jan/1950 and 31/Dec/2049
class Date {
public:
    // CONSTRUCTORS
    Date(Day d, Month m, Year y)        // Construct a Date from (d, m, y)
        throw(
            invalid_argument,           // One or more of the arguments d or m is/are invalid (27, 13, 13)
            domain_error,               // (d, m, y) as a triplet does not define a valid date (30, 2, 61)
            out_of_range);              // Date is out of range (4, 7, 2053)

    explicit Date(const char* date)     // date in string format -- to be parsed as static format member
        throw(invalid_argument, domain_error, out_of_range);
                                        // "13-03-77" for "dd-mm-yy"
                                        // "2-7-2018" for "d-m-yyyy"

    Date()                              // Default Constructor - construct 'today' (get from system)
        throw(domain_error, out_of_range);

    Date(const Date&);                  // Copy Constructor

    // DESTRUCTOR
    ~Date();                            // No virtual destructor needed

    // BASIC ASSIGNMENT OPERATOR
    Date& operator=(const Date&);

    // UNARY ARITHMETIC OPERATORS
    Date& operator++();       // Next day
    Date& operator++(int);    // Same day next week
    Date& operator--();       // Previous day
    Date& operator--(int);    // Same day previous week
```

```
    // BINARY ARITHMETIC OPERATORS
    unsigned int operator-(const Date& otherDate);    // Number of days between otherDate and current date
    Date operator+(int noOfDays)                      // Day noOfDays after (before) the current date
                                                      // (sign of noOfDays decides ahead or behind)
        throw(domain_error, out_of_range);

    // CAST OPERATORS
    operator WeekNumber() const;   // Cast to the week number of the year in which the current date falls
    operator Month() const;        // Cast to the month of the year in which the current date falls
    operator WeekDay() const;      // Cast to the day of the week of the current date

    bool leapYear() const;         // True if the year of the current date is a leap year

    // BINARY RELATIONAL OPERATORS
    bool operator==(const Date& otherDate);
    bool operator!=(const Date& otherDate);
    bool operator<(const Date& otherDate);
    bool operator<=(const Date& otherDate);
    bool operator>(const Date& otherDate);
    bool operator>=(const Date& otherDate);

    // BASIC I/O using FRIEND FUNCTIONS
    // These functions use Date::format to write or read
    friend ostream& operator<<(ostream&, const Date&);
    friend istream& operator>>(istream&, Date&);

    // Format Function
    static void setFormat(DateFormat&);
    static DateFormat& getFormat();

private:
    // Format to represent Date
    static DateFormat format;

    // Data members for Date
    // Need to fill up
};
```

(a) Implement `DateFormat` and `Date` classes. All exceptions as specified must be properly raised. **[5 + 20 = 25 Marks]**

Grading Guideline:

Completeness & Correctness	*70%*
Code Quality (simplicity, readability, efficiency, re-usability, standard library usage, robustness)	*20%*
Comments	*10%*

(b) Write a test function `void TestDate()` (in `Testdate.cxx`) that tests all methods of both classes and checks for all exception situations. `void TestDate()` should handle *all* exceptions properly. It must not crash under any circumstance. **[20 Marks]**

Grading Guideline: Based on percentage of methods and exception situations coded.

(c) Submit the test report for your implementation for `void TestDate()`. **[5 Marks]**