ECOR 1606 Winter 2014 - Assignment #5

Notes:

- **Program comments (including name / student number in the first line) and proper formatting (i.e. indentation) are required for full marks on all programs!**
- **If you use a sample solution as your starting point in any question, please indicate this in your comments. ** Note that if you start from a sample solution, right click the file name and choose "save as" to save the file with the correct formatting. If you cut and paste while displaying the file in your browser, the formatting will likely be wrong. ****
- **Programs that do not compile and run in the Dev-C++ lab environment (ME4390/ME4377/MC5010) get a maximum of 25%.**

Question 1

Enhance your solution (or the sample solution) to assignment #4 part 2 ("*a42.cpp*") as follows:

You are to add four functions to your program. The function prototypes are below. You must follow these prototypes exactly.

```
// Test and returns whether value is a perfect square.
bool isSqr(int value);

// Returns cube root of value regardless of sign of value.
double cubeRoot(int value);

// Returns value of the nChooseR algorithm for any given N and R.
// (Also returns zero if valueR>valueN.)
int nChooseR (int valueN, int valueR );

// Returns an integer value from the user between minimum and
// maximum, inclusive.
// We assume that the calling function has output the initial
// request for input.
// This function will check the value entered and output an error
// message until a value in the correct range is entered.
int getInt(int minimum, int maximum);
```

You are to change your main program so that you use these functions. *isSqr* and *nChooseR* must be used in the for loop that prints the table. *getInt* must be used three different times: to get the start value, the number of rows, and the number of decimal places.

A sample executable has been provided, but the only difference from the one for Assignment #4 part 2 is in the error messages output by *getInt* (as *getInt* doesn't know the name of the variable being input, only its valid range).

Call your C++ file "*a51.cpp*". Submit "*a51.cpp*" as Assignment 5; A#5 – Question 1.

Question 2

Enhance your solution (or the sample solution) to assignment #4 part 1 ("a41.cpp") as follows:

You are to add five functions to your program. The function prototypes are below. You must follow these prototypes exactly.

```
// returns false if the year is before 2010.  Otherwise, returns
// true if the given year is a leap year, false otherwise.
bool isLeap(int year);

// returns the number of days in the given month of the given
// year, 0 if the month is not between 1 and 12, inclusive, or if
// the year is not greater than or equal to 2010
int daysInMonth (int month, int year);

// returns the day number of the given year for the given day and
// month, i.e. 1 for Jan 1, 31 for Jan 31, 32 for Feb 1, up to 365
// or 366 for Dec 31, or 0 if the day is invalid (not
// between 1 and the number of days in that month of that year),
// or if the month is invalid (not 1 to 12, inclusive), or if the
// year is invalid (not greater than or equal to 2010)
int dayNumber (int day, int month, int year);

// returns the number of days between the start of 2010 and the
// start of the given year, or 0 if the year is before 2010
int daysInPreviousYearsSince2010 (int year);

// returns the project day number (Jan 1, 2010 is day 1, etc.),
// or 0 if the date is invalid (day not between 1 and the
// number of days in the month of that year; month not between 1
// and 12, or year before 2010).
int projectDayNumber (int day, int month, int year);
```

Hints for function *isLeap*:

The general rule is that a year is a leap year if it is divisible by 4. There is an exception to the rule (years which are divisible by 100 are NOT leap years) and an exception to the exception (years which are divisible by 400 are leap years). Your function must, of course, deal correctly with all of this. Be sure that your function returns *false* for any year prior to 2010.

Hints for function *daysInMonth*:

Use 1 for January, 2 for February, and so on. When the month is February your function must make use of function *isLeap* in deciding how many days to return. Your function must return 0 if the month or year is invalid, i.e. month not between 1 and 12, or year before 2010.

Hints for function *dayNumber*:

Your function must return 1 if the date is January 1st, 2 if the date is January 2nd, and so on. Basically it has to add up all of the days in the preceding months and then include the preceding days in the same month. It must use function *daysInMonth*. Again, your function must return 0 if any of the inputs is invalid, i.e. day not between 1 and the number of days in the month that year, month not between 1 and 12, or year before 2010.

Hints for function *daysInPreviousYearsSince2010*:

This function returns the total number of days in all of the years between 2010 and the year prior to the one provided (i.e. year-1). Your function must return 0 if the year is 2010 (as there are no previous years), 365 if the year is 2011, and so on. Your function must also return 0 if the year is before 2010. This function must use *isLeap*.

Hints for function *projectDayNumber*:

This function must return 1 if given January 1, 2010, 2 if given January 2, 2010, and so on. It must use *dayNumber* and *daysInPreviousYearsSince2010*. Again, your function must return 0 if any of the inputs is invalid, i.e. day not between 1 and the number of days in the month that year, month not between 1 and 12, or year before 2010.

A good approach would be to start with the first function (*isLeap*). Write just that function and a main program to test your *isLeap* function. Once that is working, write the next function (*daysInMonth*), and rewrite your main program to test *daysInMonth*, etc.

Once you have all your functions working, insert your main program from Assignment #4, Question #1. Your main program must call *daysInMonth* to check the input, and *projectDayNumber* to calculate the days from the start of project. While your main program does not call the other three functions, you must include them as they are used by the two functions you do call.

A sample executable is not provided, as the behaviour is unchanged from Assignment #4, Question #1.

Call your C++ file "*a52.cpp*". Submit "*a52.cpp*" as Assignment 5; A#5 – Question 2.