ECOR 1606 Winter 2014 - Assignment #4

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.**
- **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 #3 part 2 ("*a32.cpp*") as follows:

Part 1:
For practice with conditionals and logic, your program must now deal with the correct number of days per month, rather than the simplifications of Assignment #3. In other words, you must now support:

- The correct number of days in each month (February has 28 or 29 days; April, June, September, and November have 30 days; the rest have 31 days.)
- You will need to manage leap years correctly:
    - 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).

It is recommended that you complete Part 1 before starting Part 2.

Part 2:

- Change your code so that you have just one declaration for all your "int" variables. (You may use multiple lines, if needed, but the word "int" should appear just once.) If you have any "double" or "bool" variables, make the same change for those.

- Change the sentinel "while" loop to an infinite "for" loop with a "break".

- Change any assignment statements to use ++, --, +=, -=, *= or /= where possible.

- For practice with output formatting, modulus, and conversion between double and int:
    - After each valid input, output the date entered in dd-mm-yy format with leading zeros, if applicable (e.g. 10-01-14 for Jan 10th, 2014; 02-03-04 for Mar 2nd, 3004, etc…).
    - After each gap is calculated, output the gap time in days.
    - These dates and gaps must all line up nicely, as in the sample executable.
    - At the end, output the shortest and longest gaps in days, and the average gap length in days, hours, minutes, and seconds, rounded to the nearest second.

A sample executable has been provided. Your program should behave just like this one.

Call your C++ file "*a41.cpp*". Submit "*a41.cpp*" as Assignment 4; A#4 – Question 1.

Question 2

In this question, you are to produce tables of integer values, the number of unique ways to choose three items from this value (see explanation of *nChooseR* below), its cube root, and an indication of whether the number is a perfect square or not (a number is a perfect square if its square root is an integer). See the sample executable for the exact table format.

Your program starts by reading in the first integer value (negative, zero, or positive, and in the range of -999 to 999). It then asks the user for the number of rows in the table, and then the number of decimal places desired (for the cube root). It produces a table as per the description above. The first row in the table is for the given *value*, the second row for *value*+1, etc., up to the required number of rows.

After printing the table, your program asks the user for the next integer value. The user should enter the same value as the previous time for your program to stop. If the user enters a different start value, you will produce another table. The number of rows must be between 1 and 100, or possibly less, as the largest value cannot exceed 999. The number of decimal places must be between 2 and 6.

The number of unique ways that *r* elements can be chosen from *n*, where *n* is greater than or equal to *r* and both *n* and *r* are positive integers can be determined through the formula below (Remember *n!* means n factorial per chapter 1 of the lecture notes). The formula can be optimized by recognizing that all the terms in *(n-r)!* in the denominator are common to the *n!* term in the numerator.

$$nChooseR = \frac{n!}{(n-r)! * r!} = \frac{n * (n-1) * (n-2) * (n-3) *\ldots* (n-r+1)}{r!}$$

Note that negative numbers cannot be a perfect square, but do have a cube root. Also, you may only apply *nChooseR* with *r* = 3 to positive numbers 3 or larger.

Your program must use five "for" loops: one for the sentinel loop, one each to check the start value, rows, and decimal places; one to print each row of the table. The first four are infinite "for" loops with "break"s, and the last one is a regular "for" loop.

Your program must also use a Boolean variable to keep track of whether it's the first time inside the sentinel loop (in this case, there is no previous value with which to compare). Here is some pseudo-code to give you an idea of how to use this variable:

firstTime is set to true when the program starts
….
if (*firstTime*) then
  set *firstTime* to false
else if (*value* entered is the same as last time) then
  break out of for loop
endif

As usual, a sample executable has been provided. Your program should behave just like this one.

Call your C++ file "*a42.cpp*". Submit "*a42.cpp*" as Assignment 4; A#4 – Question 2.