1E3 Practical 7
9 March 2016

Objectives: To practice call-by-reference parameters and some problem-solving

Task#1:
Marks: 2
Summary: Write a program that converts a number (representing a number of days) into a proper date format (day, month, year); given that the input number is a day-count that starts from the arbitrary date of 1/1/1900. You must write and use a function dayMthYr, see below.
Details: For financial applications, dates are often represented by a day number, which facilitates date arithmetic. For 1E3 purposes, day number 1 is the 1st of January 1900, day number 2 is the 2nd of January 1900, and so on.
You are provided with the function that converts from date to daynumber. Therefore, you need to write and test a function that does the reverse operation – converts daynumbers to dates.

Notes/Hints/Additional Details:

- Examples:
    - 1 Jan 1901 is daynumber 366.
    - 8 July 2005 is 38540.
    - 4 Dec 1963 is 23348.
- Since you are already provided with the function that converts from date to daynumber, your job is to write the function that does the opposite, and then write a main function to test both functions (i.e. test both types of conversions and display the results on the screen).
- Hint: the declaration of the function is as follows:
  ```
  void dayMthYr (int daynumber, int& day, int& month, int& year);
  ```
  The call-by-reference parameters allow this function to place the appropriate values for day, month and year into these parameters. Nothing is returned by this function. Demonstrate it working by converting a date to and from daynumber, and checking the answer is what you started with.

Task#2:
Marks: 2
Summary: This is an extension to the previous task. Add a feature to your program to be able to perform some date arithmetic, such as "**- 3 w**" (which means 3 weeks ago).
Details: Write and test a function to be used by a program that interprets clauses such as "five weeks later", "two days ago", "a week ago", "three weeks from now" etc. For the sake of this program, you don't need to deal with such clauses yourself, rather, assume that someone else will convert these clauses into calculation specifications for you, such as: "**+ 5 w**", "**- 2 d**", "**- 1 w**", "**+ 3 w**".

Notes/Hints/Additional Details:

- Your function should be named `calc_date` and it should take a daynumber, and parameters specifying the calculation to be applied. It should return the resulting daynumber. For example:
    - `calc-date (100, '+', 10, 'd')` adds 10 days to the date whose number is 100.
    - `calc_date (1204, '-', 4, 'w')` subtracts 4 weeks from day 1204.

- Your function should handle adding and subtracting (‘+’, ‘-‘), days and weeks.
- In order to test and demonstrate that your function works properly, you need to write a program to conveniently test your calc_date function. This program should allow the user to enter a date in “dd mm yyyy” or similar form, and a specification for a calculation (not necessarily using the shorthand form – you could be more flexible), and should print the resulting date in “dd/mm/yyyy” or similar form.

Task#3 (Advanced Task):
Marks: 1
Extend calc_date to handle addition and subtraction of years. The function will need to convert the daynumber to day/month/year form, adjust the year element and convert back to daynumber form.

Start with
P7.cpp, on the web page, which contains the functions daynumber, leap_year, month_length and year_length.