Lab 4: A Second C Program

Aim

This lab class will provide you with an opportunity to:

- learn how to create a Visual Studio project containing a C program;
- experiment with C.

Context

The context for the exercise is Easter. Given a year number from the user, when will Easter occur?

Tasks

1. Create a new C project with name `Lab4` as explained on the accompanying link on MyLO. (Although you may use any C editor, it is recommended that you learn how to use **Microsoft Visual Studio** as this is the system used for assignments).

   Integrated development environments (IDEs) allow the editing, compilation, and execution of a 'program' from within the application. Microsoft's Visual Studio (and Visual Express) is an IDE which supports programming in many languages. One of these is C. Unlike other IDEs, e.g. versions of this IDE, are freely available. This IDE is scalable. Although it may seem complicated to use at first, it will allow very complex programs (consisting of many files) to be managed and compiled easily.

2. Within the project create a C source file called `easter.c`

3. Enter the code overleaf into `easter.c`, save it, and compile and run it (removing any errors you may find) by selecting "Start without debugging" from the **Debug** menu (or pressing the <*Control*>-<*F5*> key combination). You will likely be told that the project is out of date and asked to confirm whether you would like to build it. Click **Yes**. If you don't you will not be executing the current version of the source code.

4. Experiment with the compiler by introducing errors into the program and exploring the error messages.

Note: this program uses features of C we haven't yet covered in class. You are welcome to look ahead and/or speculate on what is going on. You are also welcome to simply ignore anything you don't recognise!

Note also: this program is poorly written. There's no header comment stating the purpose of the program, its authorship, or its date. There are no meaningful

identifiers, no comments documenting the purpose of the variables or the sections of the code. This is deliberate and is a counter-example to a well-written program!

```c
#include <stdio.h>


void easter(int year, char **month, int *day)
{
    char *months[]={"January", "February", "March", April",
                        "May", "June", "July", "August",
                        "September", "October", "November",
                        "December"};
    int a, b, c, d, e, f, g, h, i, k, l, m;
    int mon;

    a = year % 19;
    b = year / 100;
    c = year % 100;
    d = b / 4;
    e = b % 4;
    f = (b + 8) / 25;
    g = (b - f + 1) / 3;
    h = (19 * a + b - d - g + 15) % 30;
    i = c / 4;
    k = c % 4;
    l = (32 + 2 * e + 2 * i - h - k) % 7;
    m = (a + 11 * h + 22 * l) / 451;
    mon = (h + l - 7 * m + 114) / 31;
    *month = months[mon - 1];
    *day = ((h + l - 7 * m + 114) % 31) + 1;
}


int main(int argc, char *argv[])
{
    int my_year, my_day;
    char *my_month;

    printf("Please enter the year: ");
    scanf("%d",&my_year);

    easter(my_year, &my_month, &my_day);
    printf("Easter in %d is/was %s %d\n", my_year,
                my_month, my_day);

    getchar();
    return 0;
}
```