Exercise 5 – Making Decisions

Objective

The major objective is to practice decision making within a C program and to consolidate all material covered so far.

Reference Material

This is based on the *Making Decisions* chapter and language material from the *Data Types* chapter and the *Expressions* chapter.

This practical session is located in the following directory:

Windows Directory:	**c:\qacprg\decision**
Windows Solution directory:	**c:\qacprg\decision\solution**
Linux Directory:	**/home/user1/qacprg/DECISION**
Linux Solution directory:	**/home/user1/qacprg/DECISION/Solution**

Overview

Questions 1 and 2 are standalone. Questions 3 and 4 are followed up in a later exercise.

Practical Outline

1. For Windows, open the Visual Studio Solution **numbers2.sln,** for Linux edit **numbers2.c**. Write a program that asks the user for a number, reads it in and reports if the number entered was positive, negative or zero.

2. For Windows, open the Visual Studio Solution **leap.sln**, for Linux edit **leap.c**. If a year is exactly divisible by 4 but not by 100, the year is a leap year. Write a program that asks the user for a year and reports either a leap year or *not* a leap year. (*Hint*: x % y is zero if x is exactly divisible by y.) Test with the following data:

Leap years	*Non-leap years*
1996, 1984	1997, 2001
2000, 1964	1900, 1967

3. For Windows, pen the Visual Studio Solution weekday.sln For both Windows and Linux take a look at the code template provided in weekday.c. Complete this

program, to ask for a date in DD/MM/YYYY format and print out the day of the week for this date.

There is a formula, called Zeller's Congruence, which calculates the day of the week from a given day, month and year. Zeller's formula is:

```
z=(1+d+(m*2)+(3*(m+1)/5)+y+y/4-y/100+y/400) % 7
```

where `d`, `m` and `y` are day, month, year and `z` is an integer (0 = Sun .. 6 = Sat).

However, with the following adjustments *before* use in the formula:

If month is 1 or 2 and year is a leap year, subtract 2 from day.

If month equals 1 or 2 and year is not a leap year, subtract 1 from day.

If month is 1 or 2, add 12 to month.

Your program should print out the name of the day (e.g. Monday), e.g.:

1/1/1980	Tuesday	9/8/1982	Monday
25/12/1983	Sunday	31/5/1989	Wednesday
2/2/1990	Friday	29/2/1992	Saturday

Enhance your program to display whether the day is a weekday or on the week-end and how many days there are in that month. Make sure the latter checks for leap years! (Solution in weekday2.c)

Optional

4. Still working in the **weekday.sln** Visual Studio Solution, add checks to **weekday.c** to validate the date that the user types in. For example, your program should reject obviously wrong dates such as 32/4/1993 or 1/15/1993. As a second step, your program should catch incorrect dates such as 31/4/1993 or 29/2/1993.

   NB: 2000 Compliancy!!!

   In question 2, it is stated that a year is a leap year if it is exactly divisible by 4 but not by 100. There is an exception to this rule. Years exactly divisible by 400 are leap years. The year 2000 is a good example.

   You may like to add this refinement to your program. Use the following test dates:

31/12/1999	Friday	1/1/2000	Saturday
28/2/2000	Monday	29/2/2000	Tuesday

A solution for this question may be found in the **solution\val_date.sln** Visual Studio Solution.