Exercise 9 - Structures

Objective

The major objective is to practice structures and nested aggregates in general.

Reference Material

This is based on the *Structures* chapter. The practical session is located in the following directory:

Windows Directory:	**c:\qacprogex\struct**
Windows Solution directory:	**c:\qacprogex\struct\solution**
Linux Directory:	**/home/user1/qacprg/STRUCT**
Linux Solution directory:	**/home/user1/qacprg/STRUCT/Solution**

Overview

Questions 1 to 4 are related and increase in functionality. Question 1 is a simple structure. Question 2 is a redesign, which replaces one of the members with a nested structure. Question 3 and the optional question 4 introduce arrays of structures.

Practical Outline

1. Open the Visual Studio Solution **person.sln**. Create a `struct Person` with the following members:

`name`	A string (maximum 30 characters and a terminator).
`age`	An integer.
`sex`	A `char` (for `'M'` or `'F'`).

   Write a `main` function that declares a variable called `me` of type `struct Person`. Initialise this variable with your details, and in the code, increment `me`'s age member, then use `printf` to display the value of all the members of `me`.

   A solution for this question is available in the **solution\person1.sln** Visual Studio Solution.

2. Still working in your **person.sln** Visual Studio Solution, modify **person.c** as follows: replace the `age` member in `struct Person` with a member called

`birth_date`. This new member should itself be a structure of type `struct Date` (use `struct Date` in the course notes).

Change the initialisation of `me` to reflect this change.

Change the `main` you have written so that it displays the details of `me` to the screen. The `birth_date` field should be displayed in DD/MM/YYYY format.

A solution for this question is available in the **solution\person2.sln** Visual Studio Solution.

3. Open the Visual Studio Solution **phone.sln**, and take a look at the code template provided in **phone.c**. The following `struct` and variable declarations are provided; use them to write a program that displays the list of the phone numbers.

```
struct PhoneNum
{
     int area;             /* area code */
     char num[21];         /* phone number as string */
     char name[21];        /* name of person */
};

/* friends is an array of PhoneNums */

struct PhoneNum friends[5] =
{
     { 171,  "371 6657", "Mike"},
     { 171,  "983 4537", "Jo"},
     { 1753, "898320",   "QA"},
     { 1342, "123 4567", "Mary"},
     { 1462, "947 1904", "Q.E.2"}
};
```

When run, your program should produce the following output (don't worry too much about the exact format of the output!):

```
Mike -    (0171) 371 6657
Jo -      (0171) 983 4537
QA -      (01753) 898320
Mary -    (01342) 123 4567
Q.E.2 -   (01462) 947 1904
```

A solution for this question is available in the **solution\phone1.sln** Visual Studio Solution.

4. Still working in your **phone.sln** Visual Studio Solution, modify the code in **phone.c** as follows. Using the existing declarations, write a program that asks the user for a name and responds with a telephone number (i.e. it acts as an online telephone book).

   For example, if you entered `QA`, the program responds with `(01753) 898320`.

   Hints:

   Use the library function `strcmp` to compare two strings (try online help).

   To read in a string, use:

   ```
   char text[21];
   scanf("%20s", text);
   ```

   `%20s` in the scanf statement prevents `scanf` from reading more than 20 characters into the `str` array. It also places the `'\0'` `char` at the end!

   A solution for this question is available in the **solution\phone2.sln** Visual Studio Solution