Exercise 8 – Arrays

Objective

The major objective is to use arrays and strings (arrays of char). A further objective is to consolidate looping and the use of functions.

Reference Material

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

Windows Directory:	**c:\qacprogex\arrays**
Windows Solution directory:	**c:\qacprogex\arrays\solution**
Linux Directory:	**/home/user1/qacprg/ARRAYS**
Linux Solution directory:	**/home/user1/qacprg/ARRAYS/Solution**

Overview

The first two questions deal with arrays of integers, with the second question leading on from the first. Questions 3 and 4 deal with arrays of characters that represent strings. The following questions include two questions on more advanced array manipulation (using arrays of `int`s) a question for experienced 'String Manipulators'.

There is an optional part to Question 7 which uses the library function `sprintf()` and 2D `char` arrays.

Practical Outline

1. Open the Visual Studio Solution **prt_arr.sln**, and take a look at the code template provided in **prt_arr.c**. Notice the following function prototype:

```
void print_array(int [], int);
```

The 1st parameter is an array of integers to print, and the 2nd parameter is the number of array elements. The `main` function calls `print_array` twice:

```
int main(void)
{
     int my_array[5] = { 7, 6, 5, 4, 3 };
     int other[7] = { 2, 4, 6, 8, 10, 12, 14 };
     print_array(my_array, 5);
     print_array(other, 7);

     return 0;
```

```
}
```

Write the `print_array` function, to print out the values in all elements of an array of integers. When you run the program, it should produce the following output:

```
7 6 5 4 3
2 4 6 8 10 12 14
```

Note: If you prefer, simply write a loop within `main` that produces the same output. When you have fully tested your program, redesign using a function `print_array`, as described above.

2. As explained in the course notes, C does not support whole-array assignments. Open the Visual Studio Solution **cpy_arr.sln**, and take a look at the code template provided in **cpy_arr.c**. Using this code template, write a function called `copy_array` that takes two `int` arrays of the same size (the third `int` parameter) and copies one to the other, element by element. The prototype for this function is:

```
void copy_array(int [], int [], int);
```

The code contained in **cpy_arr.c** is as follows:

```
int main(void)
{
    int a1[5] = {3, 5, -1, 7, 6};
    int a2[5] = {2, 2, 2, 2, 2};
    copy_array(a2, a1, 5);
    print_array(a2, 5);

    return 0;
}
```

This outputs: `3 5 -1 7 6`

3. Open the Visual Studio Solution **slen.sln**, and take a look at the code template provided in **slen.c**. Write a function that has the following prototype:

```
int slen(char []);
```

`slen` returns the length of a given string, i.e. the number of `chars` up to, but not including the `'\0'` character.

(*Hint*: this function will need to loop, looking at each element of the array until it finds a `'\0'`, counting as it goes.)

4. Open the Visual Studio Solution **scpy.sln**, and take a look at the code template provided in **scpy.c**. Write a function that has the following prototype:

```
void scpy(char [], char []);
```

`scpy` returns nothing, but copies the string specified in the second argument into the string specified in the first argument, character by character.

Optional:

5. Open the Visual Studio Solution **comp_arr.sln**, and take a look at the code template provided in **comp_arr.c**. Design and code the function `comp_array` with the prototype:

```
int comp_array(int [], int [], int);
```

The function returns a `-1` if the two arrays passed as parameters are identical. Otherwise, it returns the position where the arrays first differ. The third `int` parameter indicates how many elements are to be checked, e.g.:

```
comp_array({2,2,2,2,3,4}, {2,2,2,7,8,9}, 6);
```

This returns 3.

6. Open the Visual Studio Solution **swap_arr.sln**, and take a look at the code template provided in **swap_arr.c**. Design and code the function `swap_array` with the prototype:

```
int swap_array(int [], int [], int);
```

The function returns a boolean, i.e. true to indicate that the two arrays have been swapped and false to indicate that the arrays are identical (and didn't need swapping). The code template **swap_arr.c** contains definitions of `print_array` (question 1), `copy_array` (question 2) and `comp_array` (question 5), which you may find useful!

7. Open the Visual Studio Solution **str_find.sln**, and take a look at the code template provided in **str_find.c**. Write a function that has the following prototype:

```
int str_find(char [], char []);
```

`str_find` attempts to find the second string argument within the first string argument. It returns the index at which the second is found in the first. If not found, `-1` is returned.

```
str_find("Welcome to QA.", "ome");
```

This returns 4.