CS23710 Worksheet Week Five 2000-01 - Version 2

David Price

March 6, 2001

1 Introduction

This week's practical is intended to provide practice with the use of functions.

The functions you write MUST take exactly the parameters specified in the sections listed below.

The problems in this practical build on those from the last two weeks.

PLEASE NOTE: We require you to all work using constructions which are valid in ANSI 'C' and not to use any extensions which the gcc compiler might provide. We suggest that you ALWAYS specify the appropriate flags to gcc to force it to give you warnings etc. if you stray outside of ANSI 'C'.

You might find gdb helpful in locating bugs in your programs.

2 The Problem - Data Input Format

A tri-athlon competition consists of three components, a cycling race, a swim and a final running race. The winner of the competition is the one with the shortest total time for the three races.

For each competitor, your input should consist of ten items. Firstly, the name of the competitor on a single line; this should not include any "white space", use underscore to provide gaps in the name. A second line of input should provide three integers representing the competitor's time for the cycle race in hours, minutes and seconds. A third line of input should provide the competitor's time for the swim in the same format. A fourth line of input should provide the competitor's time for the final running race in the same format.

The user should be provided with a prompt for each line of input expected.

An example of the input, together with the prompts is:

```
Competitor's name: Fred_Bloggs
Time for cycle race (hours minutes seconds): 1 35 47
Time for the swim (hours minutes seconds): 1 29 23
Time for the running race (hours minutes seconds): 1 05 14
```

Your input should be in EXACTLY the format given above.

3 The Problem - Part One

Your solution to this section MUST use an array of structures.

You are REQUIRED to use a structure to hold the values being processed by your program.

The structure should have a field to hold the competitor's name, an integer field which you use to hold the 'competitor number', a field to hold the cycling race time in seconds, another to hold the swimming race time in seconds, another to hold the running race time in seconds and a final field to hold the total time in seconds.

The competition has precisely eight competitors.

Your program should loop through the eight competitors, requesting the data for each competitor, storing each competitor's details in the array.

You are REQUIRED to write a function which takes as a parameter a pointer to a structure as defined above, and prints out the competitor's name, number, race times and total time. Each line of output should look like the examples given in the table below.

Your program should go through the array and for each competitor should then print out the names and times in a table form.

Don't forget the earlier remark about designing first and don't forget to use meaningful variable names and comments.

4 The Problem — Part Two

You are now required to extend the program from part one so that after printing out the results in the tabular form, your program then sorts all the results in the array until the fastest competitor is stored in element zero.

You MUST write a separate function to sort the array. The function should be passed a pointer to the array of structures.

(Remember my silly phrase "the name of an array is a pointer to its first element").

Using the print out function written for part one, after sorting the data your program should print out the sorted results. The output should look like...

```
NAME       competitor number  cycle time   swim time   run time   total time
-=========================================================================
Helen_Fuell        7          1H 15M 10S  1H 47M 17S  1H 05M 27S  4H 07M 54S
Bert_Hill          3          1H 23M 09S  1H 57M 05S  1H 10M 04S  4H 30M 18S
```

and so on for the other six competitors....

It is very important that you design what you plan to implement for this section before you start to write the code.

NOTE: I want you to write code to implement a simple sorting algorithm, NOT just use a function from the C libraries !

5 Effort Allocated to the Worksheet and Assessment

This week's practical is the third of a short sequence based around the same problem. This worksheet will be assessed as the next contribution to your mark for CS23710.

I do not want any separate documentation, but I do expect that the code is internally documented with comments. During assessment of the work, we will carefully consider issues such as layout, naming of variables, general readability, suitability of comments and so on as well as simple accuracy of the code. We expect you to use ANSI 'C' features and we will penalize breaches of this requirement. You should make sure the printout is adequately fixed together and clearly bears your name. You should also affix to the front a completed copy of the departments "Declaration of Originality" which is available from our web site.

You should give in a separate printout for each of the two problems specified above.

You may need to spend an hour or so of your own time to complete this worksheet in addition to your two hour practical.

6 Submission Date (for CS23710 students)

All students registered for CS23710 are required to hand in their printouts between 14:00 and 16:00 pm on Monday 12th March 2001.

The material should be placed in the appropriately labelled slot in the hand-in cabinet in the "Computer Science Advisory Office".

The demonstrators will keep a record of who attends the practical.

7 A Useful Simple Program

```
/* This is a first C Program */

struct score { char name[80];
               int racetime;
             };

makeoutput(struct score * scoreptr)
{       printf("the value is %d\n", (* scoreptr).racetime);
}
```

```
main ()
{

struct score runners [2];

strcpy(runners[0].name, "fred");
runners[0].racetime = 5;

strcpy(runners[1].name, "bert");
runners[1].racetime = 9;

makeoutput (&runners[0]);
makeoutput (&runners[1]);

}
```