# CS23710 Assessed Assignment 2012-2013
# Monitoring the Runners and Riders.

**David Price and Fred Long**

November 5, 2012

## 1 Introduction

PLEASE NOTE: We require that your programs are compliant with a recognised version of ANSI 'C'.

Our "official environment" in which all your programs must run, and in which we will test them, is that provided by the Computer Science computers **wiked** and **minted**.

The best way to access these machines is via the SunRay units in room B57. It is also possible to access **wiked** and **minted** from other campus networked computers running an X Window System such as that provided by **Xming**. You are likely to have to quote their fully qualified DNS names as **wiked.dcs.aber.ac.uk** or **minted.dcs.aber.ac.uk** from many locations. You can also access our machines from outside Aberystwyth **provided** you have established an AberVPN connection first.

You **must** use one of the following to develop and debug your program.

- NetBeans and the compilers and debuggers accessible from it;
- SolarisStudio (also known as SunStudio) and the compilers and debuggers accessible from it.

## 2 Monitoring Open Countryside Events

You have been contracted to produce a software system to monitor the positions and times of competitors/entrants in cross country events.

The events involve large numbers of people competing in horse riding, cycling and walking challenges.

Most of the events have multiple courses, some aimed at experienced entrants, others aimed at novices. Entrants must nominate which course they wish to enter in an event. Each entrant will be given an official number which will be a positive integer. All courses for a given event share a common pool of entrant numbers.

The courses cover a variety of footpaths, bridleways, byeways and roads. It is normal for a given track to be part of multiple courses, sometimes used in one direction for one course and another direction in a different course.

Some of the junctions between the tracks are marked (e.g. with arrows) but do not have marshals. It is not unusual for competitors to take the wrong track at a junction and thus they deviate from the official course for which they have entered.

At certain points on the course there are official checkpoints and entrants' numbers and times are recorded and passed back to the event control for safety purposes and so friends and supporters can see how everyone is progressing.

The entrants start at different times, typically a few minutes apart. Their start time is noted (that being the time they are seen at the first checkpoint for their course) as is the time they finish (that being the time they are seen at the last checkpoint for their course).

Some events, for instance long distance events such as endurance horse races, have medical check points at places on the course. As a competitor enters such a place the time is noted. They, or their horse, or both, are now checked for medical fitness. If they are found to be unfit, they are not allowed to continue on the event and control is notified. If they are fit, they are allowed to proceed and the time at which they restart is noted. The time spent in the medical check point is *not* counted as part of their event time. Thus, no one is disadvantaged if they reach a medical check when there is a long queue of other competitors already waiting.

The initial "test" deployment of your system will be used to monitor an endurance horse race in Mid Wales. We will supply you with data files to define the event. We will also supply start times, check point times, arrival and departure times at the medical check point and finish times for all competitors.

# 3 Simplification and Warning

**NOTE:** While much of the information supplied in this assignment specification is closely related to real issues of tracking the movement of entrants in competitions, various simplifications have been made to reduce the complexity of the problem.

For the purposes of this assignment, all events can be assumed to start and finish at the same place.

Any given events will run during the hours of one day only, that is, no entrant will ever start on one day and finish after midnight thus on the next day.

You may assume that there is only one unique track providing a direct link between any two junctions or checkpoints (generically referred to below as nodes).

You may assume that no event has more than 26 courses.

The node numbers mentioned below are all positive integers.

All courses start and finish at a time checkpoint.

The track numbers mentioned below are all positive integers.

All data supplied in files or manually arrives in the correct chronological order.

Our information, and any software you produce, should **NOT** be used for the monitoring of real events without you conducting a proper *risk assessment first*!

# 4 Entrant Management and Time Tracking

You are required to design, write and test two ANSI C programs to help solve various tasks that could assist an event planner to monitor an event in progress.

More details are given in the "main mission" (for program one) and "extended mission" (for program two) sections later in this document.

## 4.1 Units used in our Data and for Measurements and Calculations

All times given in our data files will be represented using a "24 hour clock" format. As an example, 13:24 would be used to represent 24 minutes past one o'clock in the afternoon.

All time durations used in the program data will be given as a simple integer in minutes.

In output from your programs the same format for times and time durations should be used.

Your program should consider that "the current time" commences as the time in the event description file and is then later updated to match the time associated with the last item of checkpoint data that has been read in.

### 4.2 Updating an Entrant's Location as time goes by

Your programs should be keeping a record of all entrants' locations (presumed or actual) as time goes by. Their location can either be:

- not yet started;
- a time checkpoint;
- a track number (assuming an entrant went the correct way when leaving the last checkpoint);
- a medical checkpoint;
- course completed correctly;
- excluded from the course for taking an incorrect route;
- excluded from the course for medical safety reasons.

## 5 The Data Files Provided

We provide three sets of data files for each event.

The first set of data files describes an event and the courses for an event.

The second set of data files defines the entrants, their allocated number and which course they intend to complete.

The third set of data files give the times that entrants reach the various checkpoints on the course.

### 5.1 Data Files - The Courses for an Event

All the courses for an event make up a "graph" where the links are some form of track and the nodes are one of three types.

Any one course will consist of a selection of nodes and thus the interconnecting tracks.

The first file will have three lines of information. The first line will have the title of the event which will be no more than 79 characters. The second line will have the date as free format text, but no longer that 79 characters. The third line will be the start time in the format specified earlier.

The second file will define the full set of nodes. The file will have multiple lines, one per node. There will be two values per line, a positive integer, being the node number, then some spaces and then two characters defining the node type. The characters are:

- CP - for a normal checkpoint where single times are recorded;
- MC - for a medical checkpoint where both an arrival and a departure time is recorded;

- JN - for a junctions between tracks where no time is recorded;

The third file will define the full set of tracks. The file will have multiple lines, one per track. There will be four values on each line, all integers separated by spaces. The first integer will be the track number. The next two integers will be the node numbers at each end of the track. The third number will be the number of minutes within which any entrant should be able to safely complete a journey along that track.

The fourth file defines all the courses for this event. All items on each line will be separated by one or more space characters. The file will have multiple lines, one per course. An initial single capital letter will be used as the course identifier; this will be followed by an integer specifying how many nodes make up this course and that will then be followed by other integers which are the node numbers for that course.

## 5.2 Data Files - The Entrants for an Event

We will supply one file defining all the entrants. The file will have multiple lines, one per entrant.

Each line of information will commence with a positive integer being their competitor number, then some spaces, then a single capital letter indicating the course they are registered for, then more spaces and finally their name as multiple words. No name will occupy more than 50 characters including any embedded spaces.

## 5.3 Data Files - The Times at Time and Medical Checkpoints

This type of file will have multiple lines, each line representing one entrant reaching a time or medical checkpoint. The lines will be in chronological order.

Entrants arriving incorrectly at time or medical checkpoints, which are not the next node on their course, will be excluded.

Entrants may also be excluded for medical safety reasons at medical checkpoints.

Entrants who have been excluded should be recorded as such in your system. It should be assumed that they will be removed from the course safely and they should no longer be expected at any future checkpoints. Your system should record whether their exclusion is because they took the incorrect course or whether it was for a medical safety reason.

There are five types of line in this type of file.

One file may have times for many different time and medical checkpoints. We will typically provide several files of this type.

### 5.3.1 Data Lines for Time Checkpoints

There are two types of line representing an entrant reaching a time checkpoint. The fields in each line will be separated by spaces.

If an entrant arrives at a correct time checkpoint for their course, the line will commence with a single capital letter **T** followed by an integer for the node number, an integer for the entrant number and then a time in the format described earlier.

If an entrant arrives at an incorrect time checkpoint for their course, the line will commence with a single capital letter **I** followed by an integer for the node number, an integer for the entrant number and a time in the format described earlier. The entrant will be excluded at the checkpoint.

#### 5.3.2 Data Lines for Medical Checkpoints

There are four types of line representing one entrant arriving, departing, or being excluded at, a medical checkpoint. The fields in each line will be separated by spaces.

If an entrant arrives at a medical checkpoint which is part of their course, the line will commence with a single capital letter **A** followed an integer for the node number, an integer for the entrant number and a time, in the format described earlier, for when the entrant arrived.

If an entrant arrives at an incorrect medical checkpoint for their course, the line will commence with a single capital letter **I** followed by an integer for the node number, an integer for the entrant number and a time in the format described earlier. The entrant will be excluded at the checkpoint.

Each departure line will commence with a single capital letter **D** followed an integer for the node number, an integer for the entrant number and a time, in the format described earlier, for when the entrant departed from the medical checkpoint.

If an entrant is excluded for medical safety reasons the line will commence with a single capital letter **E** followed an integer for the node number, an integer for the entrant number and a time in the format described earlier for when the entrant was excluded at the checkpoint.

### 5.4 Alternative Data Files

The filenames for the data we provide must not be "hard coded" into your program in any way.

It should be possible for us to provide new data files and then use your program to process the data purely by us providing the names of the new files.

Indeed, we may choose to deliberately have some extra sets of data files, of exactly the same format, which we choose to use when we evaluate your program, which quite deliberately are **NOT** available to you during development!

While any files we would use would be of the same format to those described above, we would have different numbers of courses, tracks, nodes and entrants.

## 6 Your Missions

For the first part of your assignment, "the main mission", you will be assuming that all entrants take the correct route and all finish their planned course.

For the second part of the assignment, "the extended mission", you must tackle a more realistic scenario. Some entrants will take "the wrong turn" at some junctions and thus arrive at checkpoints where they are not expected and in addition, some entrants will fail medical checks. In both cases, the entrants become excluded from the course and therefore do not finish.

### 6.1 Your Main Mission

For this section we assume that all competitors take the correct course and that the courses do not contain any medical checkpoints.

You are required to write a program that first reads in the files defining an event and then reads in a sequence of other files giving the times at which competitors reach the various time checkpoints and arrive and depart from the medical checkpoints.

You are required to have the following features in your program.

When first started, the program must prompt for the name of:

1. a file which contains the event name, date and start time;
2. a file which contains the nodes;
3. a file which contains the tracks;
4. a file which contains the courses;
5. a file which contains entrants/competitors.

Your program should now allow the user to:

- query the current location/status of an individual competitor;
- ask how many competitors have not yet started;
- ask how many competitors are out on the courses;
- ask how many competitors have finished;
- manually supply times at which individual competitors have reached time checkpoints;
- read in a file of times at which competitors have reached checkpoints;
- produce a results list of competitors including their courses/times.

**Note:** as mentioned above, the checkpoint time file for this mission will not include any entries for medical checkpoints.

## 6.2 Your Extended Mission

In this section you must deal with a slightly more realistic scenario where entrants:

- take the wrong tracks at junctions, and thus arrive at the wrong checkpoints;
- or fail checks at medical checkpoints.

In both cases, the entrants are excluded.

You are required to produce a completely separate **second** program, (developed from your first program) but which is now capable of dealing with the more complex data files. If people are excluded your system should note this and then include those people in two separate sections of your final results.

Your program should retain all the features of your first program and in addition, should provide features that enable a user to:

- supply times at which a competitor entered or left a medical checkpoint;
- produce a list of competitors who were excluded for taking the wrong course;
- produce a list of competitors who were excluded at medical checkpoints.

**Note:** as mentioned above, the checkpoint time file for this mission **will** include entries for medical checkpoints as well as time checkpoints.

Any other output produced by your program should allow correctly for the existence of the medical checkpoints and the various ways that entrants might be excluded.

### 6.3 Implementation Requirements

You are expected to make good use of the facilities of the C programming language (1989) or C99 that are supported by our compilers.

Your program **must** make use of **BOTH** *arrays* and *linked data structures* at appropriate places.

Your program **must** make good use of **functions** to modularise your code in a sensible way.

### 6.4 Comments and Layout

We do of course expect students to sensibly comment their program, making sure that all comments add real value and do not just, in essence, duplicate code. The program should have good layout and must use meaningful names for variables, structures and other identifiers.

### 6.5 Choices you are Free to Make

You need to read in the data files and store the information within suitable data structures, which you have designed, within your program.

Your program should use multiple functions, located in multiple files as you choose. As part of our assessment, we will evaluate the quality of the choices you have made.

### 6.6 Testing and Data Processing Requirements

As well as any testing you conduct to debug and validate your program, you **MUST** also analyse each of our sets of data files.

We will produce a second "further information" document clearly stating exactly how many sets of data files we have produced and their names and directories etc.

## 7 Assessment Criteria

Bearing in mind that this project concerns software development, the most appropriate "assessment criteria" are those in Appendix AA of the student handbook, namely those for "Assessment Criteria for Development".

This assignment is worth 50% of the total marks available for this module.

**NOTE: Solutions which do not tackle the extended mission could potentially be awarded a mark of up to 75% if they cannot be criticized in any manner and are accompanied by all other materials as indicated below.**

**Solutions must also tackle the extended mission for marks of 76% or above.**

## 8 The Material you Must Submit

You MUST hand in a printed version of all sections of the code of your program. **Note:** we will normally use the printed version of your code when we are judging the quality of the program you

have produced. You are thus firmly advised to make sure that the layout, when printed, looks of high quality. This is likely to mean that you should avoid using very long lines in your code.

You MUST hand in printout showing the results of attempting to compile your program and any errors or warnings that are produced.

You MUST give in printout of the output generated when running your program to process our set of data files.

You MUST hand in a short written document describing the design decisions you made. This document must be brief and between two and four sides of A4 in length. Include diagrams if you wish to help explain your design.

If you have used any minor pieces of code taken from elsewhere, or drawn understanding from published books or web pages, you should include appropriate references and a bibliography in your document. Please make sure you fully understand the department's stance on plagiarism as described in the current student handbook.

You MUST make available a machine readable version of your program on a CD or DVD or on a "USB pendrive". HOWEVER, please note that the item containing the machine readable version of the program will NOT be returned to you but will be retained by the Department. The CD/DVD/USB pendrive must also include copies of all the logfiles that were produced while processing our data files.

You MUST put all the printed written material you submit in a single folder and it MUST be possible to read, and write on, all the sheets that you hand in without us having to remove them from the folder. In particular, you must not hand in loose sheets in a bag, or put everything in plastic pockets of a folder. The "folder" does not need to be a commercially made folder, indeed, we are happy to accept sheets neatly stapled together as a "leaflet".

The CD, DVD or USB pendrive MUST be attached to your written material in a sensible manner so that on the one hand it will not get lost, but on the other hand, it is easy for us to remove and use.

NOTE: This is an "individual" assignment and must be completed as a one person effort by the student submitting the work.

You must attach, to the front of your submission, a completed and signed copy of the departmental standard assignment cover sheet which includes a Declaration of Originality.

# 9 Submission Date

Your solution to this assignment must be handed in to the Computer Science collection system between 9:00am and 4.00pm on Friday 14th December 2012.

It is not acceptable to miss any lecture for the purposes of completing assignment work, in terms of the department's monitoring of student attendance and any actions that might be taken in respect of poor attendance.

# 10 Feedback

To enable you to generate "self feedback" we plan to release our own "outline/sample solution" (which we note will not be perfect!) during December, hopefully before Xmas day. You will then be able to use this to see how we tackled the problem, especially in any areas where you had difficulties.

It is our intention to provide you with individual feedback on your work by approximately mid January.