CS23710 Assignment 2015-16 (REVISED)

Dave Price

20 November 2015

1 Introduction

This is the CS23710 practical assignment for 2015/2016 Semester 1. It will count for 50% of the total marks for the module. You should spend roughly 32 hours of effort on this assignment.

The submission section has been revised on 20th November 2015 to clarify how submission is to be made.

Deadline: The deadline for submitting the assignment is **2.00pm on Friday 11th December 2015**.

Note. Your assignment **must** be submitted electronically via AberLearn Blackboard as detailed at the end of this document.

2 Programming Environment and Implementation Requirements

You **must** use NetBeans and the gcc compilers and gdb debuggers accessible from it to develop and debug your program. In addition, we also specify that you *MUST* use **remote build host support** and you must select to use `central.aber.ac.uk` as the remote build host.

Our "official environment" in which all your user programs must run, and in which we will test them, is that provided by the environment described above.

This assignment **MUST** be solved using dynamically created structures.

PLEASE NOTE: We require you to all work using constructions which are valid in C89 (ANSI 'C' (1989)), C99 (ANSI 'C' (1999)) or C11 (ISO/IEC 9899:2011) and not to use any extra extensions which the `gcc` compiler might provide.

We recommend that you turn on any facilities that will warn you if you stray outside of ANSI 'C'.

3 Analysing Competition Results — The Overall System

3.1 Horticultural Competitions - Implemented using Dynamic Data Structures

The Ceredigion Horticultural Association supports the running of various horticultural (vegetable and fruit growing) competitions each year at the end of summer.

A horticultural competition has an unknown number of competitors!

You will only know how many people have actually entered any competition as the results arrive.

We supply two data files, one corresponding to a 2013 event and one corresponding to a 2014 event.

Each file contains the name and the date of the competition and all the results.

The number of competitors can only be discovered by detecting "end of file" while reading the data. You must not "hard code" the number of competitors into your program. Indeed, your program should be

designed so that if we fed in data from a future staging of the competition, in which there were many more competitors, it should not fail unless the operating system dynamically refused to provide enough memory to your program. Should this happen, your program should correctly trap such a problem and report this to the user.

The datafiles, named hort_2013.txt and hort_2014.txt, are linked from the AberLearn Blackboard web site for this module.

Each horticultural competition consists of three components, a cucumber growing competition, a carrot growing competition and a runner bean growing competition.

The overall winner of a competition is the competitor with the maximum total length of fruit and vegetables (note: a cucumber is technically a fruit).

The competitions have great tradition and have been run for many years. All fruit and vegetables are measured using the British Imperial system. Lengths are thus recorded in feet and inches.

[Note: There are 12 inches in a foot.]

3.2 Program Design Requirements

1. Your program must use a dynamically built binary tree data structure to hold the results of the competition. The "key" used to decide whether to insert a new node on the left or right subtree with respect to an existing node should be the total number of inches of fruit and vegetables grown. Nodes representing a competitor with less length of fruit and vegetables should be stored on the left subtree and nodes with a greater length on the right subtree. If the length of fruit and vegetables in a new node is equal to that of an existing node the new node should be placed in the right subtree.

2. Your program must define a suitable structure ("C" struct) to hold the data for a single competitor. These structures are to be used as the nodes of the tree mentioned above.

   The structure must have **nine** fields.

   The first seven fields are used:

   - one to hold the competitor's name;
   - one to hold the competitor's postal address;
   - one to hold the competitor's phone number (assume this is text);
   - one to hold a competitor number (which you allocate automatically as you read in the data for each competitor, use 1 for the first competitor, 2 for the second and so on);
   - one to hold the length of their cucumber converted to inches;
   - one to hold the length of their carrot converted to inches;
   - one to hold the length of their runner bean also converted to inches.

   The final two fields should be suitably defined as pointers to the next node in the left (shorter length) subtree and the right (longer length) sub tree respectively.

   Note: although the input and output lengths are in the form of feet and inches, the data structure MUST only store the lengths converted to inches.

3. Your program must dynamically request storage space for new nodes using the functions malloc() or calloc().

4. You MUST use functions as defined below as part of your solution to the problem.

5. When you run your program, it should process one file of data, produce the results as required and then exit.

4 Your Task — Part One

4.1 Input Data

Your program should commence by requesting the name of the data file from the user.

Your program should then open the specified file.

The program should read the data for a competitor, place this in a new structure (using memory space acquired using malloc() or calloc()) and then insert this new structure in the appropriate place within your dynamically built binary tree of structures.

Remember to also allocate each competitor a "competitor number" and store that in the correct place within that competitor's structure.

Your program should repeat reading in data on competitors until end of file is encountered.

NOTE: no line of text input will exceed 79 characters in length.

4.2 Data File Input Format

The file will commence with two lines of text, the first being the title of this specific competition and the second containing the date of the competition.

The input for each competitor will follow and will consist of twelve items. Firstly, the name of the competitor on a single line. Secondly, the postal address of the competitor on a single line. Thirdly the phone number of the competitor on a single line. A fourth line of input will provide one integer and one real number representing the cucumber length in feet and inches. The number of feet will be given as an integer, the number of inches as a real number with one decimal place. A fifth line of input will provide the carrot length in the same format. A sixth line of input will provide the runner bean length in the same format.

A printout of the contents of the data file for the 2013 competition is given at the end of this assignment specification.

4.3 Data Output

You must write a function which has a single parameter, namely a pointer to a single competitor structure, and which prints out the data in that structure in the exact format as that used for a single competitor as shown in the example below.

You must now write a function which traverses your tree in the direction **in-order**, (i.e. left sub-tree, node, right subtree). When traversing the tree, your function should call your node printing function and thus produce output of identical layout to that shown below.

4.4 Data Output Format

```
Competition: Capel Sion Horticulture.  Date: 6th September 2013.
NAME       competitor number      Cucumber        Carrot   Runner Bean  Total Length
================================================================================
Margaret Mouse     1            1ft  0.5in    2ft  3.2in    1ft  4.5in    4ft  8.2in
Donald Duck        3            2ft  0.4in    2ft  4.5in    1ft  6.2in    5ft 11.1in
Bert Hill          2            1ft  7.2in    1ft  8.2in    2ft 11.7in    6ft  3.1in
```

and so on for the other competitors.

5 Your Task — Part Two

You should create a second program, based on your first program, but this time also including new functions as described below.

You must design and implement a new tree traversing function, based on that created for part one, but this time one of the parameters to the function should be of the type "pointer to function".

The function pointed to by the second parameter, (which I call a "node action function") should be called by the tree traversing function once for every node in the tree.

You should also produce two node action functions.

- One of these functions can be very similar, possibly identical, to the node printing function produced for part one.
- The second node action function should print out contact details for a single competitor in the format shown below.

If your new tree traversing function is called with the actual parameter set to the node printing function then the output should be identical to that produced in part one of this assignment.

If your new tree traversing function is called with the actual parameter set to point to the contact details function then the output should be as shown below.

5.1 Data Output Format

```
Competition: Capel Sion Horticulture.  Date: 6th September 2013.
Competitor Contact Details.

Competitor Name: Margaret Mouse
Postal Address: Skirting Board House, Mosehole, Devon. DV1 2ST
Telephone: Southern 9565

Competitor Name: Bert Hill
Postal Address: 14, Priory Lane, Birmingham, West Midlands. B20 1RU
Telephone: Central 2010

Competitor Name: Donald Duck
Postal Address: Village Pond, Duckington-by-Sea. S12 1QQ
Telephone: Marine 1234567
```

and so on for the other competitors.

6 Effort Allocated to this Assignment

You are likely to need to spend up to 32 hours of your time to complete this assignment.

6.1 Guidance as to overall mark allocation

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

A perfect solution to "Your Task — Part One" above, could earn a "middle 2(1)" grade. A student aiming for higher marks should also complete "Your Task — Part Two".

7 Assessment Criteria

This assignment will be assessed using the "Assessment Criteria for Development".
See the Student Handbook, Appendix AA, via the URL:
`http://www.aber.ac.uk/compsci/Dept/Teaching/Handbook/appendices2.htm`

The usual requirements for coding projects apply, namely that programs should include comments that add real value and do not just, in essence, duplicate code. Programs should have good layout and must use meaningful names for variables, functions, structures and other identifiers.

8 The Material You Must Submit

The submission must be made electronically using Blackboard. There will be appropriate links in the Assignments part of the course materials for module CS23710.

You must submit (via Blackboard) *three* things.

1. NetBeans Project Export of Program One (Your Task — Part One).

   Your NetBeans Project Export **MUST** be made using NetBeans. You should make it very clear which version of NetBeans has been used and on what computing platform. A stated earlier, you **MUST** use remote build host support and use `central.aber.ac.uk` as the build host.

   You MUST make sure that your export includes all of the source code of your program addressing features required by "Your Task — Part One". **Note:** we will be judging the quality of the layout of the program you have produced.

2. NetBeans Project Export of Program Two (Your Task — Part Two).

   Your NetBeans Project Export **MUST** be made using NetBeans. You should make it very clear which version of NetBeans has been used and on what computing platform. A stated earlier, you **MUST** use remote build host support and use `central.aber.ac.uk` as the build host.

   You MUST make sure that your export includes all of the source code of your program addressing features required by "Your Task — Part Two". **Note:** we will be judging the quality of the layout of the program you have produced.

3. A Design, Compile / Run Document

   You MUST submit a document (in PDF format). This document must include:

   - A short section of between two and four sides of A4 in length describing the design decisions you made. Include diagrams if you wish to help explain your design.
   - If you have used any pieces of code taken from elsewhere, *including* any code from case study examples used during the module, you *must* include comments at the appropriate places in your program code and include references and a bibliography in your document.

     If you have drawn understanding from published books or web pages, you *must* include 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.

- The document must also include a section showing the output produced when attempting to compile your program and any errors or warnings that are produced. This can be a collection of "screen shots'
- The document must also include a section showing the output produced by your program when processing both our supplied data files. This can be a collection of "screen shots".

9 Submission Information

The deadline for submitting all parts of the assignment is **2.00pm on Friday 11th December 2015**.

The submission must be made electronically using AberLearn Blackboard. There will be an appropriate links in the Assignments part of the course materials for module CS23710.

By making your submission on Blackboard it will be deemed that you are declaring the submission to be your own work.

NOTE: This is an "individual" assignment and must be completed as a one person effort by the student submitting the work. Please take note of the statement on Unacceptable Academic Practice which can be found in Section 5 of the Department Handbook. See:
`http://www.aber.ac.uk/compsci/Dept/Teaching/Handbook/handbook.htm`

10 Example Data File - filename hort_2013.txt

```
Capel Sion Horticulture.
6th September 2013.
Margaret Mouse
Skirting Board House, Mosehole, Devon. DV1 2ST
Southern 9565
1 0.5
2 3.2
1 4.5
Bert Hill
14, Priory Lane, Birmingham, West Midlands. B20 1RU
Central 2010
1 7.2
1 8.2
2 11.7
Donald Duck
Village Pond, Duckington-by-Sea. S12 1QQ
Marine 1234567
2 0.4
2 4.5
1 6.2
```