University of Exeter

College of Engineering, Mathematics

and Physical Sciences

ECM2433

The C Family

Continuous Assessment

Date Set:	30th January 2018
Date Due:	22nd February 2018
Return Date:	15th March 2018

This CA comprises 15% of the overall module assessment.

This is an **individual** exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

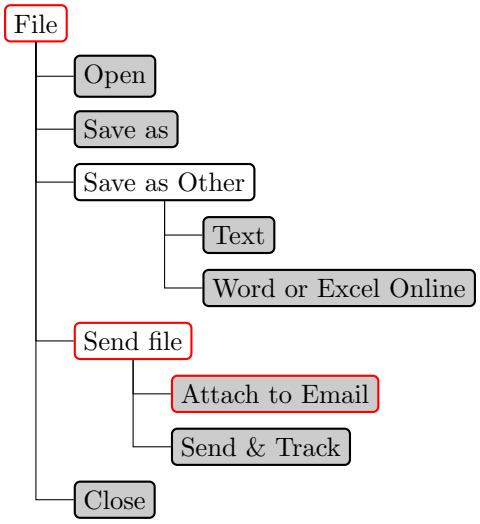
This is the first coursework for this module and tests your skills and understanding of programming in C.

Please Turn Over

Problem Statement

Many interactive computer systems allow the user to select options or functions from a set of menus. These menus are hierarchical, so the user selects an option from the main menu and is presented with a sub-menu, from which they make a further selection. This sub-menu may itself have sub-menus. Eventually the user will select an option that performs some task rather than opening another sub-menu.

The menu structure can be represented as a tree, with each (sub-)menu represented as a node, and the eventual tasks as leaves. For example, here is part of the Adobe Acrobat Reader menu structure, represented as a tree:



The shaded boxes represent leaves, and an example menu selection is shown by the red outlines on selected nodes: first the user selected "File", then the "Send File" submenu, and then the "Attach to Email" option which, because it is a leaf node in the tree, causes some action to occur.

You must write an ANSI-standard C program to read in the structure of a menu from a text file, store it as a tree and then display the menu structure by printing it out to the screen.

The Task

On the study resources website on ELE (`http://vle.exeter.ac.uk`), you will find two example data files, with the output that your programs should produce in each case.

Your executable program must be called `menuTrees` and must accept one parameter from the command line. This parameter is the name of the file to be processed, i.e. the file that contains the data for constructing the menu tree. Thus, if the data file is called `testExample1.txt`, then the command

```
menuTrees testExample1.txt
```

should cause the program to process this test file.

Each data file contains rows in one of two formats. Format "A" defines a node by assigning it a unique number and a text label; format "B" links a child node to its parent (e.g. a sub-menu to the main menu). The first character of each row should be "A" or "B" to indicate the format of the rest of the data in that row. Rows in format "A" will have the following structure:

start position	name	datatype	length	description
1	dataFormat	character	1	Always has the value 'A'.
2	nodeID	integer	4	An integer between 1 and 9999 inclusive. The unique identifier of this menu option.
6	textLabel	character	≥ 1	A string representing the menu option's text label.

Rows in format "B" will have the following structure:

start position	name	datatype	length	description
1	dataFormat	character	1	Always has the value 'B'.
2	childNodeID	integer	4	An integer between 1 and 9999 inclusive. The unique identifier of the child.
6	parentNodeID	integer	4	An integer between 0 and 9999 inclusive. The unique identifier of the parent.

Here are the contents of a valid file in this format, which represents the example Adobe Acrobat Reader menu structure shown on page 1:

```
A0001File
A0002Open
A0003Save As
A0004Save as Other
A0005Text
A0006Word or Excel Online
A0007Send File
A0008Attach to Email
A0009Send & Track
A0010Close
B00010000
B00020001
B00030001
B00040001
B00050004
B00060004
B00080007
B00090007
B00100001
```

The "A" records will always come first. If the parent ID of a given menu item is "0000", then it is a top-level menu (so in the given example "File" is the only top-level menu).

The program must read the data contained in the specified file, construct the whole menu in memory and then print out the menu structure from memory in the following form (using the same example):

```
1 File
   1.1 Open
   1.2 Save As
   1.3 Save as Other
      1.3.1 Text
      1.3.2 Word or Excel Online
   1.4 Send File
      1.4.1 Attach to Email
      1.4.2 Send & Track
   1.5 Close
```

Each child node is indented by three spaces with respect to its parent. Nodes at the top of the menu structure ("File" in this case) are numbered sequentially, starting with 1. A child is numbered sequentially as a sub-number of its parent, with the first child taking the parent's number and a sub-number of 1 (e.g. "Text" is a child of "Save as Other"; the parent is numbered 1.3, this is the first child, so it is numbered 1.3.1).

Program output

The program should produce two streams of output. The menu produced by processing the input file must be sent to the **stdout** stream. The remaining output, as well as any error or warning messages,

must be sent to the **stderr** stream. The **stderr** output must list the values the program has read in from the command line and the data it has read in from the file. The total stderr output for the example shown above, assuming that there are no error or warning messages, should be like this:

```
Parameter listing:
   1: input file name: testExample1.txt

Data values read from file:
    1: File
    2: Open
    3: Save As
    4: Save as Other
    5: Text
    6: Word or Excel Online
    7: Send File
    8: Attach to Email
    9: Send & Track
   10: Close
   child 2, parent 1
   child 3, parent 1
   child 4, parent 1
   child 5, parent 4
   child 6, parent 4
   child 8, parent 7
   child 9, parent 7
   child 10, parent 1
```

Deliverables

The deliverables for this coursework comprise both electronic and paper submissions.

Electronic submission of your C program (source code) must be made via the usual (Harrison) electronic submission process for work item "ECM2433 CA1 menuTrees". The program must be in ANSI standard C, and must compile, link and run on one of the Linux machines (either `emps-ugcs1` or `emps-ugcs2`). You must also submit a shell script that compiles and links your program. The executable must be called **menuTrees**.

You must submit a printed report to the Education Office using BART, which includes the following:

- A printout of the source code for your C program.
- Instructions for compiling and linking your source code into an executable.
- A short report (maximum 3 sides of A4) describing the design of your program, particularly the representation you have chosen for the menus. The report must also describe any assumptions you have made.
- A list summarising the errors you are trapping in your program.
- The output produced by your program for each of the examples on ELE.

Note that your program will be tested on data other than that provided on ELE, and these test files may be considerably bigger than the examples given.

Submission must be made by 12pm (noon) on the date indicated on the front page of this document.

Suggested Approach

If this seems like a hard problem to tackle, I suggest you develop your program in the following steps, making sure each element works to your satisfaction before continuing with the next stage:

1. Process the command line parameters and output the stipulated messages to **stderr**.
2. Read the file specified on the command line and output the stipulated messages to **stderr**.
3. Define the data structure that will represent a menu. Look at *both* example files before doing this.
4. Populate the menu from the data read from the file.
5. Print out the menu structure in the correct format.

Marking Scheme

Criteria	**Marks**
Program basics *Does your program compile and link without errors? Does it conform to ANSI C standards? Does it trap appropriate run-time errors and take appropriate actions? To what extent does its structure conform to the standards of a well-structured C program? To what extent does your program (1) correctly read in the command line parameters, (2) correctly read the data from the file specified in the command line parameters, and (3) produce the specified output to the* ***stderr*** *stream?*	40
Creating the menu structure *To what extent does your program implement the menus as a suitably-constructed tree? Does it impose a fixed limit on the number of menu items (nodes) that can be accepted (it should not; the only limit should be the amount of computer memory available)? To what extent does your program correctly print out the menu structure to* ***stdout****?*	50
Report *Have you included clear instructions for compiling and linking your program? Does the report clearly describe the data structure used to represent the menus? Are the spelling, grammar, language and presentation of the report clear, formal and professional and appropriate to the intended audience?*	10
Total	100

Penalties

You will be penalised 20 marks if you fail to make an electronic submission of your program code and 10 marks if you have not included evidence of the messages and output produced when your program is run against the example data given on ELE.

If you implement the menus as a data structure other than a tree with links represented by pointers, then the maximum mark you will be able to achieve is 69%.