University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Assignment 2: Climbing Trees

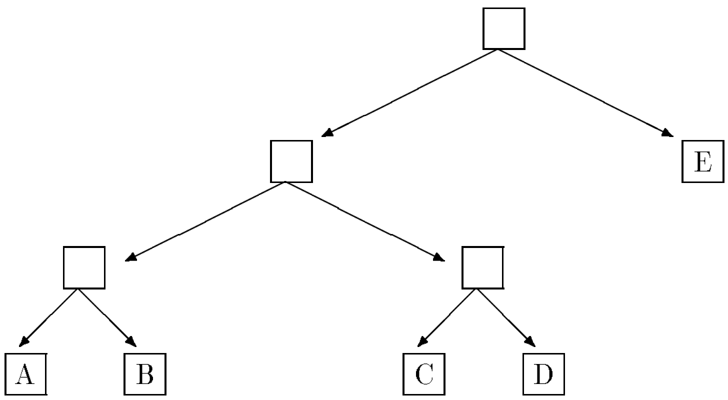
This assignment will give you further practice with file manipulation, introduce the bottom-up programming style, and prepare you for working in group projects.

We will proceed in two stages. In the first stage, you will develop a collection of simple programs, each of which must be tested individually. In the second stage, you will be told how these programs interact, and will integrate them into a useful system, which must be tested again. You will also write a report, as you did for the first assignment, and provide test input and output.

Part I – Trees and file I/O
Due: in tutorial, June 23

For each of the following programs, you are to write code which implements the required functions, and test each program carefully. Note that only the first two programs listed manipulate file data.

- **count.c** Reads a file and counts the number of occurrences of each character of the alphabet in that file.
- **fileio.c** Takes a character string of 0's and 1's, converts the string to a binary equivalent, and writes the converted string to a binary file. For example, if the character string is "0110111001101000101101" then the first three bytes of your binary file should be 01101110 01101000 10110100. Note that in this example, the last two bits were zero-padded.
- **buildtree.c** Given a list of pointers to the roots of binary trees, repeatedly combine these trees until there is only one root. For example, if the original roots were labelled A, B, C, D, and E, then your function would produce a binary tree which looks like:



- **addtree.c** Given a path description[1], to a node, display the contents of that node if it exists, or otherwise, extend the binary tree to include the node and all of its necessary ancestors.
- **traverse.c** Traverse a binary tree, starting from the root, and provide a path description for each leaf.

What to submit

For this first part of the assignment, you must only submit your test input and output, to demonstrate that each of these programs works correctly. Although your report is not due until the second part of this assignment, you will again find life much easier if you have written a description of your data and file structures, as well as an explanation of each program module.

[1] A path description is a list of branches taken, from the root to a given node, for example, in the previous diagram, the path description to node B is "left, left, right."

University of Toronto
Department of Computer Science
CSC 228 - Summer 1994

Assignment 2: Climbing Trees

Part II – Huffman Encoding
Due: in tutorial, July 7

Now we get to have some fun. In the first part of this assignment, you wrote most of the low-level code you will need to construct a Huffman encoder/decoder program. You may need to make some minor modifications, but the task should be quite straightforward. Your program should take an ordinary text file as input and produce an output file containing the same information encoded according to the Huffman coding scheme. Or, if the user specifies, your program should decode the Huffman-coded output back to plain text. The necessary character-encoding table should be kept in the same file as the encoded data.

Some points to consider:

- How will your program detect the end of the encoding table and the start of encoded data?
- How will your program know to stop decoding if the last codeword in your file ends before the end of a byte?
- Will your program work on a non-character based file? For instance, can you Huffman-encode an executable program?

What to submit

Your report should be based on the first two stages of the assignment, and indicate what changes you had to make from the first to the second version. You must include your program code as well as sample test files of input and output data as appendices.

Your output data should include the compression ratio obtained by Huffman encoding, and you should discuss, in your conclusions, the effectiveness of your program on different inputs. For example, what is the compression ratio with UNIX manual pages, with C program text, with C program text from which you have removed comments, etc.?

Finally, as for the first assignment, you must submit your executable program using the *submit* command on the CDF system. Do not submit your source files, only a single, final executable. See `man submit` for more information.