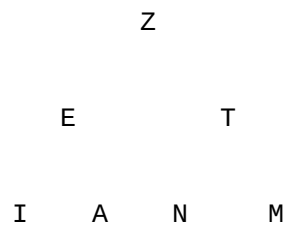
Assignment #6 – Morse Code Decoder

Date Assigned: Wednesday, November 3, 2010
Date Due: Thursday, November 12, 2010
Points: 50

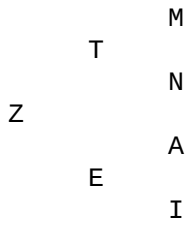
Write a C program that will decode files of Morse Code into plain English.

You are to finish implementing the Binary Tree ADT we started in class. The header can be found in the CS300Public directory on zeus as bt.h. I have changed this somewhat, so pick up the latest version. The new version is better defined and documented. Set up the Binary ADT module as all others with folders Headers, Sources, … as before. You are to include a driver called btdriver.c that tests ALL of the functions that are a part of the Binary Tree module found in the header file bt.h. You should have this module completed by Friday to give yourself enough time to finish the entire assignment.

The Binary Tree module has a btPrintTreeSideways function to print the binary tree just constructed on its side. The data content of each node in the tree is to be printed so that the spatial relationship among the nodes is shown on the output. In particular, we choose to display the tree "on its side." For example:



will be displayed this way:



Add lines to this tree by hand before you turn in your final output.

Note1: Your tree display is built by using the following rules:

1) Each datum is printed on a separate line

2) A datum at level i is preceded by 4(i-1) blanks. This means that all values at the same tree level will appear in a column on the printed output.

3) The right subtree of each node is printed on lines above the line containing the node, which in turn appears before lines containing the left subtree

Once you have completed implementing the Binary Tree ADT, you are to build a project called MorseCode that takes a Morse Code file and prints the decoded Morse Code in English. Set this module up as all others with folders Headers, Sources, … as before. You are to include a driver called **mcdriver.c** that builds a Morse Code Tree using **morsecode.txt** and then decodes a morse code file **morsemessage.txt** in the folder Testfiles. Your driver is to include and use the Binary Tree API created to build the Morse Code Tree.

I will test your Binary Tree module with the command: **./btdriver** and I will test your Morse Code module with the command **./mcdriver**.

As always,

1. You are to break up your program into appropriate .h/.c files and on the day the assignment is due, turn in a colored hard copy of each .h/.c combination (fully documented).

2. Your code is to be written in C using Eclipse 3.6. Programs written in other environments will not be graded. Submit a tarball called 06punetid.tar.gz using the submit script on zeus. Make sure to completely test your solution on zeus before submitting your final solution. This is a more complicated tarball and the tarball must extract and work correctly on zeus. The extracted tarball is to have two Eclipse projects: (a) BTDynamic and (b) MorseCode.

3. You are to use the coding guidelines from V6.0 of the coding standards.

Goals for this assignment:

1. Implement the Binary Tree ADT using dynamic memory allocation.

2. This C code is pointers, handles, and dynamic memory allocation. HARD!!!!

3. Break your program up into well thought out modules.

4. Use well thought out functions in solving this problem. Don't break code out later into a function.

5. Code and test your program one function at a time.

6. Reuse your BTDynamic code to help you implement the Morse Code project.

7. Write efficient/clean code

8. Use all of the tools in a more advanced ways including makefiles, the debugger, and valgrind. Make sure you have zero memory leaks.

9. I will put the datafile morsemessage.txt in CS300Public on zeus at some point, but just use a small datafile to test your solution. The file to be decoded contains morse code that is to be decoded into English. Each set of morse code characters representing one encoded value will be separated by one space or one or more newline characters. Whenever you read a newline, output a newline in your outputted message.

Along with printing the source code and decoded message, print out the Morse Code Tree on its side and fill in the lines by hand.

Use the file **morsecode.txt** found in the CS300Public folder to create your Morse Code Tree. Each line in the file contains a character, followed by a space, followed by the character's morse code. We will talk in class about how to build the tree.