University of Illinois at Chicago
CS 202: Data Structures and Discrete Mathematics II — Handout 10
Professor Robert H. Sloan — November 27, 2002

Program 5(b): Using Graph for Word puzzle

Due: Friday, December 6, 12:00 NOON
Warning: This deadline will not be extended.

Please use the `Graph` class to solve a word puzzle problem similar to the one sketched out in Problem 9.50 in the Weiss text. Note that you are to follow the specifications on this handout, **which are different from Weiss**.

You are indeed to write a program to determine whether one word can be changed to another by a series of 1-character substitutions. For example, BLEED to BLOOD via the sequence of 1-character substitutions:

BLEED, BLEND, BLOND, BLOOD.

Your program will take three command-line arguments. The first two arguments are the start and end word. The third command-line argument is the name of a file containing a dictionary of words to use.

(I suggest you use `usr/dict/words`.)

Your `main` program should print out an error message to `cerr` and exit with a return value if the two words have different length, or if either word is not in the dictionary.

Otherwise, if such a transformation is possible, your program should print it out, one word per line, with the input words as the first and last output lines. If such a transformation is not possible, your program should print the one line IMPOSSIBLE.

You do *not* have to find the shortest possible such transformation.

Your transformation *does* have to have the following efficiency property: it cannot be possible to just skip over one or more words in your output sequence and still get a valid output sequence.

That is, BLEED, BLEND, BLAND, BLOND, BLOOD is *not* okay.

Graph

You will of course model this problem as a graph problem. We could in fact get a fine solution using undirected graphs, but since what we have is a directed graph class, that will work just fine too.

Turing in your program

Please give us: all files you use, including the `Graph` class code and header files, a short README, and a (working!) makefile.

Please set up your makefile so that your executable is called `puz`.

(It's short for "puzzle," in case you found that name puzzling.)

Extra Credit

If you are able to use exactly the `Graph` class you turned in, you will get 17 extra points.

If you think your qualify, please tell us this in your README file.

(Sorry, but I can't give partial extra credit for "near misses" once I post my code for people to use.)