Cs352 — Homework #6
Tries — continue

March 4, 2004

Due Time: 3/11/04 (9:00PM).

Turnin ID: *cs352_assg6*

Submission in pairs is allowed.

In this homework you are requested to continue the implementations of tries, this time while working with files. Your program should support the following commands, which are expansion of the ones from hw5.

c — Create an empty trie for S. Here we assume that S is empty (i.e., contains no elements). After each command is entered, the program should return with the corresponding prompt, as does the example program in the course directory. Delete all data previously stored at the trie from the trie (and free memory cell you used).

i w — Insert the word w into the trie for S. For example, `i hello` should insert the word "hello" into S.

d w — Delete the word w from the trie for S. For example, `d hello` should delete the word "hello" from S. (make sure to free the unused cells).

p print all words currently stored in the tree, in lexicographically increasing order (again — check the example program for the exact format).

s filename Save the tree to the file named `filnemae.dat`. See details below regarding the format of the file.

l filename Load the data from the file `filename.dat`. Erase all data that was in the tree before.

q Quit the program. Before exiting, make sure to free all cells your program used.

Attempting to insert an excising word, or attempt to delete a non-existing word, calling an undefined command, or specifying a word that contains a letters which is out of range, should caused an error message — see the example program.

BTW - (actually goes without saying) changing the value of N (the size of the alphabet) should not effect the correctness of the program.

In the 'inset' and 'delete' commands, you can assume that there is a single blank character separating the command from the word.

The program should halt if the "malloc" command returns NULL. It should print an error message (you can pick any meaningful message).

File format

The first line of this file contains the number of words stored in the trie, and the number nodes (separated by a blank). The next lines contains words that stored in the tree, in an alphabetic lexicographic increasing order. Each line contains one word, and the line contains no other characters (excluding CR). For example:

```
3 8
aaa
aaaaa
bb
```

Bonus (25 points)

You can increase the speed of the algorithm in several ways. The program is more efficient when all nodes are close to each other in memory, so the probability of paging is lower. This can be archived at the beginning of the "load" process, by trying to allocate a big contiguous area in memory, that can fit all nodes in the files, treat is area as an array of struct, and fill its cells in the process of loading. Then there is no need for a single malloc operation for each node of the tree, However, recall that you cannot always assume that there such a large array available, and then your program should treat use the "old fashioned" malloc.

Also note that during the insertion process of many words, we usually do need to search the tree from the roomt, when during the process of inserting a new word. For example, think about the insertion of the words `abcdabcdabcdabcc, abcdabcdabcdabcd`, which a differ only in one character. Can you use this idea also to create a more efficient way to store the words in the file, which would both reduces the size of the file, and fasten the loading time ?