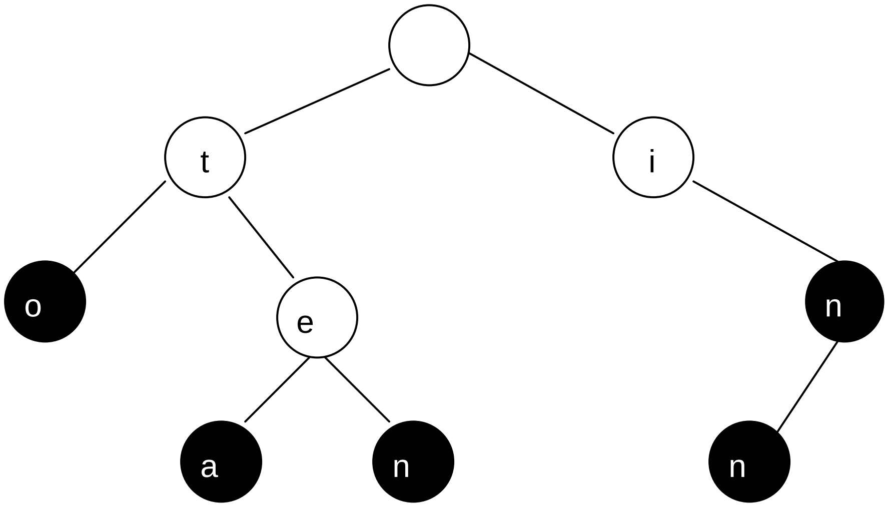
Programming Project 4 (50 points)
Due: Thursday, March 14 by midnight

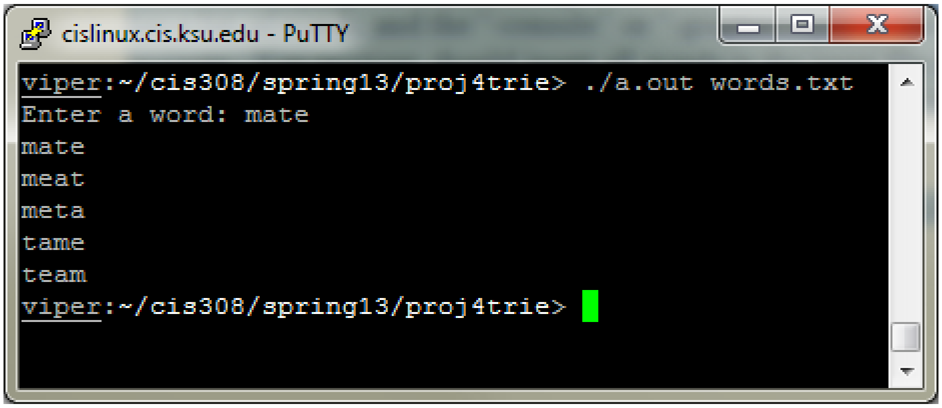
Background:

In this assignment, you will implement a trie data structure. A trie is a kind of tree where the root represents the empty string. Each node contains a character search key, and the string represented by a node is formed by concatenating all characters on the path from the root to that node. Not all such strings are words in the dictionary – instead, each node has a special mark for whether the string it represents is a word. Below is a picture of a trie that contains words “to”, “tea”, “ten”, “in”, and “inn”. Nodes with words contained in the dictionary are black, and nodes with words not in the dictionary are white.



Assignment Description:

You are to write a program in C that first builds a trie from a file of words supplied as a command-line argument. You should then ask the user for an input word, and print all *anagrams* of that word given your dictionary file. Recall from Project 2 that an anagram is a rearrangement of the characters in a word, where the rearrangement is also a valid word (and not just a jumble of letters). Here is a sample run of the program:



```
viper:~/cis308/spring13/proj4trie> ./a.out words.txt
Enter a word: mate
mate
meat
meta
tame
team
viper:~/cis308/spring13/proj4trie>
```

Implementation Requirements:
Your assignment must meet the following requirements:

- Your program should have a struct for representing a trie. It should also use an enum for defining a bool type (true/false).
- Your program should contain at least the following separate functions: buildTrie (read each word in the input file and add it to a trie), add (add a new word to a trie), search (return whether a word is in the trie), printAnagrams (finding all permutations of a word and printing only those which are in the trie), and the main function.
- You are welcome to use the model solution (or your own solution) from Project 2 when writing your printAnagrams function.
- You should release all allocated memory when the program ends

Documentation:
Your program must include a comment block at the top of every file, as well as at the top of each function. The function comments should include a brief description of what the function does, and explain any function arguments and return values. You may use the comment block below as a template:

```
/*********************************************
* Name: (YOUR NAME)                          *
* Date: (THE DUE DATE)                       *
* Assignment: Project 4: Tries and Anagrams  *
**********************************************
* (WRITE A DESCRIPTION OF THE PROGRAM)       *
**********************************************/
```

Submission:
First, you will need to create a zip file of your project. To create a zip file in Unix, put all your code for this project (probably just one .c file) in a directory called “proj4”. Change directories to one back from the proj4 directory. To create a zip file of your project called “proj4.zip”:

```
zip proj4.zip proj4/*
```

It should list all the files that it included in the zip file.

To create a zip file in Windows, again put all your code for the project in a directory called “proj4”. Then, right-click on the proj4 folder and right-click, select “Send To”, and then select “Compressed (zipped) file”. This will create a zip file with your code called proj4.zip.

To submit your project, find the `proj4.zip` file that was created above. Then, go to “`Files and Content->Modules->File Dropbox`” on K-State Online, and upload the `proj4.zip` file. **Put your name and Project 4 in the description box.**

Grading:

Programs that do not compile will receive a grade of 0. A grading breakdown for programs that do compile appears below:

Correctly reads dictionary file and builds a corresponding trie	24
Format of project (struct for trie, enum for bool, buildTrie/add/search/printAnagrams/main functions)	10
Finds all permutations of input word and prints those that are words in the trie	10
Memory is released at the end of program	5
Documentation and submission	1
Total	**50**