Homework 3

Due Date: Monday, May 13, 2002, ***at 11:59PM*** **Points**: 100

UNIX System

1. (*10 points*) What is the parent directory of /?
2. (*15 points*) The file /usr/share/dict/words contains a list of English words and abbreviations, one per line. How many words and abbreviations does it have? List all words in it with the trigram "gry" in them.

C Programming

3. (*30 points*) Write a program that reads words from the standard input and prints them in sorted order (use ASCII ordering). Use a linked list to do this. The structure of a node in the linked list is to be:

```
struct lnode {
      char *word;             /* pointer to word */
      struct lnode *nxt;      /* pointer to next entry in linked list */
}
```

   You will need to use the function *malloc*(3) to allocate both space for the word and space for the nodes. If a word occurs more than once, list each occurrence seperately. A "word" is a maximal sequence of letters and digits.

   Your program should print one word per line. For example:

Sample stdin	*Corresponding stdout*

`Hello, there, my old friend!`
`How are you today?`
`I am very well, thank you!`
`Goodbye ...`
	`Goodbye`
	`Hello`
	`How`
	`I`
	`am`
	`are`
	`friend`
	`my`
	`old`
	`thank`
	`there`
	`today`
	`very`
	`well`
	`you`
	`you`

4. (*30 points*) Please make two modifications to the program you wrote for part 3:
   a. Change the program so the user can name one or more files on the command line, and the program will take input from those files. If no files arenamed, the program should read from the standard input.
   b. Add a command-line option `-r` that causes the words to be printed in *reverse* order. (***Hint***: use recursion.)

.Debugging

5. (*15 points*) The program *getbit.c* (available on the class website) reads in two numbers, *n* and *b*. It returns the *b*th bit of integer *n*, where the smallest (rightmost) bit is bit number 0. Rather, it is ***supposed*** to. But it doesn't work. Please debug it.

Extra Credit

6. (*10 points*) In your program for problem 3, change the way you handle repeated words as follows. Suppose the word `hello` occurs 3 times. Instead of printing it 3 times, print it as follows:

```
hello (3)
```

Hint: You will need to change the structure of the node to include a counter.