Homework 3

Due: May 27, 2024 **Points:** 100

All programs are to be submitted through Gradescope. Gradescope will compile and execute your program, sometimes with varying inputs or invocations. You can see your score once executions are done. You can submit your program multiple times, up to the deadline — at that point, the score you have will *usually* be the grade for that problem. I say "usually" because we reserve the right to look at your submission and deduct points if you do not use proper programming style.

You have to name your program as given in the problem. Otherwise you will get an odd error message indicating there is a problem with Gradescope.

Also, your output must match Gradescope's *exactly*, including blanks and tabs — so pay attention to the example output!

1. (*25 points*) Write a program that prints a tic-tac-toe board. Each square is to be 7×7, with "#" marks. The "X" is to be 5×5, composed of "X"s, and placed in a square the coordinates of which are given as input.

   Your program first prompts for, and reads, two integers separated by blanks or tabs between 1 and 3 inclusive; if either is not, print an error message giving both numbers and stating it is not a valid square and prompt for two numbers again. If you encounter an end-of-file, end the program. For example:

```
> 2 4
2,4 is not a valid square; the numbers must be between 1 and 3 inclusive
> 2,4
Illegal character in input ","
>
```

   The upper left square has co-ordinates (1, 1) and the lower right corner has coordinates (3,3).

   On exit, your program returns 0.

   Here is an example input and output; the input is in red and the output in black.

```
> 1 1
       #       #
 X   X #       #
  X X  #       #
   X   #       #
  X X  #       #
 X   X #       #
       #       #
#######################
       #       #
       #       #
       #       #
       #       #
       #       #
       #       #
       #       #
#######################
       #       #
       #       #
       #       #
       #       #
       #       #
       #       #
       #       #
```

Call your program “ttt3.c”.

2. (*40 points*) Write a program that reads words from one or more files named on the command line and prints the words in sorted order (use ASCII ordering), and a count of how many times each word appears in the input. The easiest way to do this is to use a linked list. A node in the list might look like:

```
struct node
char *word; /* pointer to word */
int count; /* number of times word occurs */
struct node *next; /* pointer to next entry in linked list */
```

You *cannot* make any assumptions about the maximum length for a word, so you must allocate space for each word as well as for the node.

A word is a maximal string of digits, letters, and underscores.

When you print your words, print the count in the first 5 spaces then a blank, and then the word. In other words, use the format string `"%5d %s"`.

If your program cannot open a file for reading, print the file name followed by a colon and a space (“: ”) and then append the system error message. The best way to do this is to use the library function *perror*(3).

Your program is to return an exit status code that is the number of files named on the command line that could not be opened. So if all files can be opened, the exit status code is 0; if five such files cannot be opened, the exit status code would be 5.

Here is an example input and output. Suppose the following is the preample to the United States Declaration of Independence, and is in the file “declindep”:

```
The unanimous Declaration of the thirteen united States of America,
When in the Course of human events, it becomes necessary for one
people to dissolve the political bands which have connected them
with another, and to assume among the powers of the earth, the
separate and equal station to which the Laws of Nature and of
Nature’s God entitle them, a decent respect to the opinions of
mankind requires that they should declare the causes which impel
them to the separation.
```

Run your program as:

```
wordsorta declindep
```

The first 20 lines of your output will be:

```
    1 America
    1 Course
    1 Declaration
    1 God
    1 Laws
    2 Nature
    1 States
    1 The
    1 When
    1 a
    1 among
    3 and
    1 another
    1 assume
```

```
1 bands
1 becomes
1 causes
1 connected
1 decent
1 declare
```

Call your program “wordsorta.c”.

3. (*10 points*) Write a second version of the program in the question above, or modify your answer, but this time sort in lexicographical (dictionary) order, not ASCII order. In lexicographical order, upper case letters are treated as though they were lower case. For example, “The” and “the” are considered the same word, so your output would show “the” occurring twice.

   Using the input file in question 2, here are the first 20 lines of output:

   ```
   1 a
   1 America
   1 among
   3 and
   1 another
   1 assume
   1 bands
   1 becomes
   1 causes
   1 connected
   1 Course
   1 decent
   1 Declaration
   1 declare
   1 dissolve
   1 earth
   1 entitle
   1 equal
   1 events
   1 for
   ```

   If you look at the word “The’, you will see it occurs 11 times in the text, because 10 of them begin with a lower case “t” and one with an upper case “T” (see the output in question 2).

   Call your program “wordsortl.c”.

4. (*25 points*) The program *revfile.c* is supposed to read in up to 1023 lines of input, the maximum input line length being 1023 characters. It should then print the lines in reverse order. But it doesn’t; it crashes. Please identify the problem, and fix it.

   Here is sample output for the *fixed* program. The standard input is taken from the file in question 2:

   ```
   them to the separation.
   mankind requires that they should declare the causes which impel
   Nature’s God entitle them, a decent respect to the opinions of
   separate and equal station to which the Laws of Nature and of
   with another, and to assume among the powers of the earth, the
   people to dissolve the political bands which have connected them
   When in the Course of human events, it becomes necessary for one
   The unanimous Declaration of the thirteen united States of America,
   ```

   Call your fixed program “revfile.c”.