COP 3223 Assignment #6

Program A: Averaging (averaging.c):

Help! The "-" key on my keyboard is broken. (Shhh...don't point out the flaw in my story that I just typed it there.) I need to average a list of numbers that I will input, but I can't input negative numbers without being able to push the minus key. So instead, I've created an alternate way to enter this list.

Input a list of numbers in my special format, then output their average. My format consists of the following: each number will start with a "n", "p", or "e". If it starts with an "n" or "p", that character will be followed by a space and then a positive integer, the value of this number. If the character is an "n", then the number should be negative; if the character is a "p", the number should be positive. If the character is an "e", then this is the end of the list (nothing will be after it).

Your program should have and use the following function (written by you):

```
// Precondition: value will be a pointer to where the input value is
//    to be stored.
// Postcondition: returns true if a number was read, or false if
//    it was the end of the list. The int pointed to by value will be
//    set to the number input by this function, made negative or
//    positive depending on the character before it.
int read_number(int* value);
```

Sample Run (user input in bold and italics):

```
p 1
p 2
p 10
n 3
e
2.50
```

Sample Run (user input in bold and italics):

```
p 100
p 40
n 20
n 123
p 5
e
0.40
```

Hint: don't forget, scanf takes pointers: don't pass it a pointer to a pointer.

<u>Program B: Twos and Threes (twosandthrees.c):</u>

Tyler loves the numbers 2 and 3 because they're the only two prime numbers that are adjacent. One of his favorite ways to spend his free time is counting how many factors of 2 and 3 various lists of numbers have (don't make fun of his life choices, different people like different things). But he's found that, as his lists get bigger, he's spending too much time doing math and not enough time enjoying the factors. So he's asked you to write a program that will count these factors of 2 and 3 for him.

Write a program that will read in a list of numbers, and after reading each number, output the current count of factors of 2 and 3 seen so far. The list will be terminated by a 0 (zero).

Your program should have and use the following function (written by you):

```
// Precondition: value is a positive integer. two_count and
//   three_count are pointers to ints that hold current counts of how
//   many factors of 2 and 3 have been seen so far, respectively.
// Postcondition: two_count and three_count will be updated to
//   include counts of 2s and 3s found as factors of value.
void update_counts(int value, int* two_count, int* three_count);
```

<u>Sample Run (User input in bold and italics):</u>

```
2
So far, there have been 1 factors of 2 and 0 factors of 3.
3
So far, there have been 1 factors of 2 and 1 factors of 3.
6
So far, there have been 2 factors of 2 and 2 factors of 3.
0
See you next time!
```

<u>Sample Run (User input in bold and italics):</u>

```
64
So far, there have been 6 factors of 2 and 0 factors of 3.
27
So far, there have been 6 factors of 2 and 3 factors of 3.
36
So far, there have been 8 factors of 2 and 5 factors of 3.
1
So far, there have been 8 factors of 2 and 5 factors of 3.
0
See you next time!
```

Program C: Case Fixer (casefixer.c):

Andrew has a text file full of titles. However, he doesn't like to use his shift key. Instead, he often uses the caps-lock key. This would be fine (though extra work for him), except sometimes he forgets to turn it on/off at the right times. As such, his list of titles sometimes end up with the wrong capitalization. Can you help him fix this? And while you're at it, output line numbers, too.

Read in the tiles in a text file ("casefixer.txt"), and output the titles with proper capitalization (to standard output). To be clear, a title is a sequence of printable characters (letters, numbers, and spaces) with no line breaks. Each title is separated by a line break ('\n'). Proper capitalization of a title is the first letter being capitalized and the rest being lower case. (We will ignore any special rules about "a" or other short words and stick with this simplified definition for this assignment.)

Your program should read until the end of the file. The easiest way to do this is to check the return value of `fscanf` after attempting a read with it. If it returns `EOF` (a built-in constant), then no character was read because the end of the file was reached (and you should stop). Otherwise, the next character was read (as normal).

Your program should have and use the following function (written by you):

```
// Precondition: ch points to the character to process. char_in_word
//   and line_num point to counters of what their names describe.
// Postcondition: returns true if the character was a line break
//   (meaning following this will be a new line), or false otherwise.
//   Fixes the character pointed to by ch to have proper case.
//   Updates the two counters based on this character.
int fix_caps(char* ch, int* char_in_word, int* line_num);
```

You may also find it helpful to use the following built-in functions:

```
// Returns the upper-case version of c.
char toupper(char c);
// Returns the lower-case version of c.
char tolower(char c);
```

Hint: you do NOT need to use anything we haven't learned yet (like strings) to do this. Just think about doing this one character at a time. fscanf/scanf with "%c" will read one character each time, allowing you to process a single character at a time. These can be passed one at a time to fix_caps.

"casefixer.txt" for the below sample:

```
This line has proper case
THIS LINE HAS IMPROPER CASE
this line also has improper case
tHE qUICK bROWN fOX jUMPS oVER tHE lAZY dOG
```

Sample Run:

```
1  This Line Has Proper Case
2  This Line Has Improper Case
3  This Line Also Has Improper Case
4  The Quick Brown Fox Jumps Over The Lazy Dog
```

Deliverables:

Please submit three separate .c files for your solutions to these problems via WebCourses by the designated due date:

Program A: **averaging.c**
Program B: **twosandthrees.c**
Program C: **casefixer.c**