ICS 212 Homework 7

"Untabifying" files

Write a program that removes tabs from a given text file and replaces them with spaces. Keep track of how many lines you process and how many tabs you replace and report those totals as output. Write the changed file contents to a new file with an added `.untab` extension.

Your program will require two pieces of information from the user: the name of the input file to process and the tabstop size to simulate while replacing tab characters. You can prompt the user for these details when the program starts. (But see below for an alternate approach for extra credit.)

Then, open the input file for text reading and read in its contents. Remember that each line will end with a '\n' character, except possibly for the last line. (In other programming languages, you may have read input files in line-by-line or all at once. Though this is still an option, this approach is harder to do safely in C. You may want to consider processing the file character-by-character instead.)

You will replace any tabs ('\t' characters) with spaces. Use the user-entered tabstop value to do this. Remember how tabstops work: you do ***not*** just replace each tab character with the given number of spaces! Instead, you need to know what character you are on in the given line. For example, consider a tabstop of 4 given the following short lines as input:

```
a\tb     ->    a   b       // tab (\t) is replaced with 3 spaces
aa\tb    ->    aa  b       // 2 spaces
aaa\tb   ->    aaa b       // 1 space
aaaa\tb  ->    aaaa    b   // 4 spaces
```

The modulo operator (%) might be handy when calculating how many spaces to add for a given tab.

Do not change the original file's contents. Instead, output the "untabified" contents to a file with the same filename plus: `.untab`

Your program must be robust. It should not hang or crash no matter what input the user gives you or if anything is wrong with the files (input file does not exist, cannot be read, or if you cannot create or write to the output file). You will want to check the return values of the I/O functions you call to make this happen.

You will need to use some new functions in stdio.h. You can see the documentation in the K&R appendix or here. In particular, you should examine these constants and functions:

```
EOF, FILE, fopen, fclose,
fprintf, fscanf, fgetc, fgets, fputc, fputs
```

You may also want to consider stdlib.h's `atoi` function, and string.h.

It is very unlikely that you will need to use all of these functions to complete the assignment. Just choose which ones seem appropriate for what you need to do.

The following is an example of what such a program's output might look like:

```
UNTABIFY - replaces tabs with spaces.
Enter the name of a file to untabify: my_grocery_list.txt
Enter the tab-stop to use [default is 8]: 4

Read 17 lines from my_grocery_list.txt and replaced 9 tabs.
Output written to:  my_grocery_list.txt.untab
```

Good luck!

Command Line Arguments *(optional/ +1 point EC)*

For those users comfortable using a command line interface, it would be nice not to have to wait for input prompts but instead specify all details when starting the program.

Modify your program so that:

- If the user enters no command line arguments, the program runs as above, prompting the user for details as normal.
- If one command line argument is given, treat it as the name of the input file to use and use 8 for the tabstop value.
- If two arguments are given, treat the first as the input file name and the second as the tabstop value.

Again, remember that your program must not crash no matter what kind of arguments the user gives you.