Lab 9, CSC 101

This lab provides additional exercises on file I/O and characters/strings as well as additional experience with functional decomposition. You are also asked to examine a provided program and decompose the program into logically separate functions.

Download lab9.zip, place it in your `cpe101` directory, and unzip the file.

Tally Program

Develop the following in the `tally` directory in `tally.py`.

You are to write a program that tallies (i.e., sums) the values in a file. The file contents are meant to be organized such that each line contains multiple numbers separated by whitespace into "columns". Your program must take two command-line arguments specifying 1) the name of the input file and 2) the column of numbers to tally. Your program will print the sum of all numbers in the specified column (e.g., specifying column 0 will result in printing the sum of the first number in each line whereas specifying column 12 will result in printing the sum of the thirteenth number in each line).

Your progam should account for poorly formatted files. Specifically, if a numeric value for the specified column position cannot be determined, then print an error message (as follows) and use 0 as the value in the sum.

- If the specified column is out of range, then print the line number and "out of range".
- If the specified column contains a non-numeric value, then print the line number and "not a number".

Note: The primary goal of this exercise is to decompose the program into separate functions. As such, your solution should not have all functionality in the `main` function.

Decomposition

In the `detab` directory, in `detab.c`, you will find a program that replaces tabs with an appropriate number of spaces (i.e., a tab "moves" to the next tab stop and this program assumes that tab stops are 8 spaces apart).

This program is, quite horrifically, written entirely in the `main` function. You must identify the logically separate parts of this program and move them into separate functions. (You may note that this program does not have any associated unit test cases because ... well, how would you test something like this?)

The program should continue to work after you have decomposed the problem. To compare the output file (`detabbed`) with the expected file (`detabbed.expected`) you can use the `diff` program. Type **diff detabbed detabbed.expected**. If there are no differences, then you will see no output.

Demonstration

Demonstrate each part of the lab to your instructor to have this lab recorded as completed. In addition, be prepared to show the source code to your instructor.