Homework 5

Due Date: 3/9/15
Files to submit: **wcount.c, perimeter.c, tail.c, ReadMe.txt**

- All programs must compile without warnings when using the -Wall option
- If you are working in a group **ALL** members must submit the assignment on SmartSite
- Submit only the files requested
    - Do **NOT** submit folders or compressed files such as .zip, .rar, .tar, .targz, etc
- All output must match the provided solution in order to receive credit
    - We use a program to test your code so it must match exactly to receive credit
- All input will be valid unless stated otherwise
- The examples provided in the prompts do not represent all possible input you can receive. Please see the Tests folder for each problem for more adequate testing
- You may assume all inputs are valid unless otherwise specified
- All inputs in the examples in the prompt are underlined
- If you have questions please post them to Piazza

Restrictions

- No global variables are allowed
- Your main function may only declare variables and call other functions.

1. Write a program called **wcount.c** that counts all the words in a given file
    1. Name your executable **wcount.out**
    2. The name of the file to count the words in will be passed on the command line
    3. For this problem we will define a word as a series of 1 or more non-whitespace characters
        1. Use `isspace` to determine if a character is white space or not

    Examples
    Assume we have file named fun.txt that contains the following:
    ```
        Computer sciences classes are great!
        Even though I spend all my time doing homework :(
    ```
    ```
    1../wcount fun.txt
      There are 15 word(s).
    ```

2. Write a program called **perimeter.c** that calculates the perimeter of a polygon.
    1. Name your executable **perimeter.out**
    2. The points of the polygon will be stored in a file and this file will be passed on the command line arguments
    3. The file itself will be a **binary file containing integers**
        1. The first integer in the file is the number of points contained in the file
        2. The remaining integers are the points, with the first integer being the x coordinate and the second integer being the y coordinate.
        3. There is an edge between each adjacent point and between the first point and the last point
        4. Each file contains at least 3 points
    4. The perimeter of a polygon is the sum of the lengths of all of its edges
    5. Use a double to store your perimeter and report the perimeter to the nearest 2 decimal points.
    6. You **MUST** store your points in a struct to receive credit for this problem.
    7. As an aside the example tests do not form actual polygons but assume that they do.

    Example. Assume that there is a file called example.txt. It will store the following information but in binary form. This example is just to give you a visualization of the data.
    ```
    3
    287 422
    283 -981
    781 647
    ./perimeter example.txt
    The perimeter is 3648.30
    ```

3. Write a program called **tail.c** that prints out the last N lines of a given file
    1. Name your executable **tail.out**
    2. Arguments to your program will be passed on the command line in the following order
        1. Name of the file
        2. N
    3. N will always be at least 1
    4. If N is greater than the number of lines in the file, all of the lines in the file should be displayed
    5. You may assume that no line is longer than 100 characters
        1. For an additional challenge try to solve the problem where there is no limit on the length of a line
            1. My solution to dealing with any length lines involved using the functions `ftell` and `fseek`.
    6. There is no limit to the number of lines in a file
    7. Some lines may only contain the newline character, these still count as a line
    8. My editor automatically added 1 newline character to the end of most of the test files but does not display them. Your editor may or may not display them but do be aware that they are there.

    Example. Assume the file meme.txt contains the following

```
It
is
over 9000!!!
```

```
1../tail.out meme.txt 1
    over 9000!!!
2../tail.out meme.txt 2
    is
    over 9000!!!
3../tail.out meme.txt 10
    It
    is
    over 9000!!!
```