CptS 121 - Program Design and Development

Programming Assignment 5: Practice and Fun with Pointers

Assigned: Wednesday, February 26, 2014
Due: Wednesday, March 12, 2014 by midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Apply and implement pointers in C
- Pass output parameters to functions

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements and apply top-down design principles for a problem
- Apply repetition structures within an algorithm
- Construct while (), for (), or do-while () loops in C
- Compose C programs consisting of sequential, conditional, and iterative statements
- Eliminate redundancy within a program by applying loops and functions
- Create structure charts for a given problem
- Open and close files
- Read, write to, and update files
- Manipulate file handles
- Apply standard library functions: fopen (), fclose (), fscanf (), and fprintf ()
- Compose decision statements ("if" conditional statements)
- Create and utilize compound conditions
- Summarize topics from Hanly & Koffman Chapter 6 including:
  - What is a pointer?
  - What is an output parameter?

III. Overview & Requirements:

For the following small problems, make sure that you create a separate MS Visual Studio project for each problem!!!

1. (60 pts) Call this project Computations. Note this problem is similar to Hanly & Koffman's project 6.5. Write a program to determine the answers to the following questions:

1. Is the value a multiple of 7, 11, or 13?
2. Is the sum of the digits odd or even?
3. Is the value a prime number?

The program should read integers from a file called "numbers.txt" until all integers have been read. As each integer is read from the file, your program should answer the above three questions for that particular integer. The answers to the questions may be displayed to the screen. To determine functions that you need for this project, you should start with a structure chart (you do NOT need to turn this in, however it could be within a

comment block in your program if you please). From the structure chart you should be able to determine a reasonable top-down design for this problem (don't forget however that you must provide at least the function described above to solve this problem. You should notice from the structure chart that you probably will need a function to read a single integer from a file, a function for answering the questions, etc. Note that the function that answers the question may accept a single integer value (the number from which to answer the questions) and three output parameters/pointers (one for each answer to each question). Please have fun with this problem and display your knowledge of proper top-down design! Keep in mind that you are practicing using pointers, so the more that you can use them in your functions the better off you will be in the future!

2. (55 pts) Call this project MathematicalModels. Note this problem is similar to Hanly & Koffman's project 6.4. The table below summarizes three commonly used mathematical models of nonvertical straight lines.

Mode	**Equation**	**Given**
Two-point form	$m = (y_2 - y_1) / (x_2 - x_1)$	$(x_1, y_1), (x_2, y_2)$
Point-slope form	$y - y_1 = m(x - x_1)$	$m, (x_1, y_1)$
Slope-intercept form	$y = mx + b$	m, b

Design and implement a program that permits the user to convert either two-point form or point-slope form into slope-intercept form. Your program should interact with the user as follows (use input is shown in bold):

Select the form that you would like to convert to slope-intercept form:
1) Two-point form (you know two points on the line)
2) Point-slope form (you know the line's slope and one point)
3) Exit
==> **2 (this is where the user enters a value)**

Enter the slope=> **4.2**
Enter the x-y coordinates of the point separated by a space=> **1 1**

Point-slope form: y - 1.00 = 4.20(x - 1.00)

Slope-intercept form: y = 4.20x - 3.20

Select the form that you would like to convert to slope-intercept form:
1) Two-point form (you know two points on the line)
2) Point-slope form (you know the line's slope and one point)
3) Exit
==> **1**

Enter the x-y coordinates of the first point separated by a space=> **4 3**
Enter the x-y coordinates of the second point separated by a space=> **-2 1**

Two-point form:

$$m = \frac{(1.00 - 3.00)}{(-2.00 - 4.00)}$$

Slope-intercept form: y = 0.33x + 1.66

Select the form that you would like to convert to slope-intercept form:
1) Two-point form (you know two points on the line)
2) Point-slope form (you know the line's slope and one point)
3) Exit
==> **3**

You are once again given the freedom to determine the appropriate functions required to solve this problem. Use a structure chart to determine how to perform a reasonable top-down design. Note that you must accepts pointers as parameters in at least two of the functions that you implement. I have provided a possible top-down design below (you may ignore these functions and write your own):

get_problem - Displays the user menu, then inputs and returns as the function value the problem number selected.
get2_pt - Prompts the user for the x-y coordinates of both points, inputs the four coordinates, and returns them to the calling function through output parameters (i.e. pointers).
get_pt_slope - Prompts the user for the slope and x-y coordinates of the point, inputs the three values and returns them to the calling function through output parameters.
slope_intercept_from2_pt - Takes four input parameters, the x-y coordinates of two points, and returns through output parameters the slope (m) and the y-intercept (b).
intercept_from_pt_slope - Takes three input parameters, the x-y coordinates of one point and the slope, and returns as the function value the y-intercept.

Other possible functions include: *display2_pt ( ), display_pt_slope ( ),* and *display_slope_intecept ( )*

IV. Submitting Assignments:

1. Using the Angel tool https://lms.wsu.edu submit your assignment to your TA. You will "drop" your solution into the provided "Homework Submissions" Drop Box under the "Lessons" tab. You must upload your solutions as <your last name>_pa5.zip by the due date and time.
2. Your .zip file should contain two project workspaces. One for each project above. Within each project you must have at least one header file (a .h file), two C source files (which must be .c files), and project workspace. Delete the debug folder before you zip the project folder.
3. Your project must build properly. The most points an assignment can receive if it does not build properly is 75 out of 125.

V. Grading Guidelines:

This assignment is worth 125 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 115 pts (60 pts -> project1, 55 pts -> project 2) for adherence to the instructions stated above (see the individual points above)
- 10 pts (5 pts / project) for appropriate top-down design of functions and good style