**Programming Assignment 4**

Structures, arrays of structures, functions, header files

Program description:

Read sales and price data and compute total sales and bonuses. Use arrays of structures to hold data to process.

1. You are to read a data file (provided) containing sales data for sales people working for a company.

You are to read each line of sales data for a sales person, calculate total sales, and a bonus based on a scale.

The sales file consists of several rows of data that contain:

First Name

Last Name

ID

Product number – Number sold item pairs. These Product-Number sold pairs end with a tag of -1. There will always be a product number/number sold pair (even for the -1 sentinel), i.e., there will be no errors in the combination. However, there MAY be an error in the item number in that it is not an item carried by the company.

For example:

Pam Smith 123 2 2 1 1 3 4 -1 0

Sales person is Pam Smith, ID 123, who sold 2 of item 2, 1 of item 1, 4 of item 3. The sales data for this sales person ends with -1 (there will be no items sold of product -1, so -1 is to be a sentinel, not an invalid item number). The read for the next sales person would then begin.

1. Create a **structure** to contain the information for a salesperson:

First Name

Last Name

Company ID

Total Sales

Bonus

Declare the structure in a header file.

1. You are also to read a data file (provided) containing product information, which consists of a product number and a unit price.

For example:

1 15.95

2 12.22

Product 1 has a unit price of 15.95, product 2 has a unit price of 12.22

1. Create a **structure** to contain the information for the products:

Product Number

Unit Price

Declare the structure in the header file

1. Declare the structures for the sales and products data in the header file. Declare an array variable for each of the sales and product structures LOCALLY in main. Declare an array size for the sales data as 30; for the product data as 50.
2. Read and store the product data into the product structure array. Read until end of file. Use a counter to determine how many rows. Pass the counter in to any processing functions that loop through structure array (do not use a global variable).
3. Read and store the sales data into the sales structure array. Read until end of file. Use a counter to determine how many rows. Pass the counter in to any processing functions that loop through the structure array (do not use a global variable).
4. Calculate the total sales for each sales person while reading the data and then store in the structure. You ARE **NOT** to **store** the product/number sold pairs in the **sales array** for each sales person. The total sales calculation will involve reading the product data into the product array structure first and looking up the price based on the item number sold while reading the sales data.
5. Look up bonus values using a function. Bonus values are as follows:

$0 for Sales <= $ 500.00

$500 for Sales <= $1000.00 and greater than $500.00

$1000 for Sales <=$5000.00 and greater than $1000.00

$2000 for Sales >$5000.00

1. Read and store the product data into the structure array using a function.
2. Read and store the sales data in into the structure array using a function.
3. Process the all sales data using a function. Checking for valid product numbers. If an invalid product number is encountered in the sales data, write the invalid product number to an error file with an appropriate message of your choice. DO NOT include the product in the total sales calculation.
4. Using a function, output the computed sales data to the console (cout) and to an output data file including the following information:

First Name, Last Name, ID, Total Sales, Bonus

Include dollar signs before money values and ensure each line of data is neat and readable. All monetary values must have two decimal places.

1. Use a function to open all input and output files. Request all file names from the user EXCEPT the error file, which you may choose and hard code in the program.
2. Main should contain function calls only; no processing.
3. Close all files prior to program end. The close statements may be in main. **Three (3) points extra credit** will be awarded if the close statements are included in a function and the function is called by main.
4. All structure DECLARATIONS, global constants and function prototypes MUST be in a header file. Include the header file in the program. DO NOT CODE THE FUNCTIONS IN THE HEADER FILE. Functions MUST be coded in the .cpp file AFTER main.
5. Each function **must** have required documentation (pre and post conditions).

**Test and evaluate calculations for accuracy. Points will be taken for inaccurate calculations, improper formatting, directions not followed.**

Turn in file code \*.cpp; README file; header file and both input files. You DO NOT need to submit the executable (\*.exe). You DO NOT need to submit the output files. They will be created when I test the programs. You MAY zip all the files and submit if you choose. You MUST include all the files indicated or points will be deducted.

README must contain instructions for location of input/output files

**Points WILL BE taken if MINIMUM requirements and submissions not included.**

**1 point extra credit will be awarded if assignment is zipped for submission**

Each solution is to be uniquely your own; minimal student collaboration allowed.

See Blackboard and Syllabus for due date