COP3014-Foundations of Computer Science - Assignment #6

ALWAYS COMPLETE AND SUBMIT YOUR PROGRAM EVEN IF IT IS LATE!

Objectives:

Manage a static array: 1)Read the contents of a data file into a static array; 2)Process the data stored in an array of records; 3)Print the records store in an array to a data file;

This assignment is an extension of Programming Assignments 2, 3 and 5. You will implement a program called ***"amazon_porders5.cpp"*** to process customer purchase orders (orders) on Amazon. You will read the data stored in a data file into a static array of purchase orders records, then process each purchase order record in the array, and finally print the array of purchase order records to a datafile. The purchase orders will be stored in order records. Each order record contains nine fields, which are as follows: 1) a ten digit customer cell phone number (string, no dashes), 2) the item number (string), 3) the number of items (quantity) in the purchase order (integer),4) the cost (price) of one item (double), 5) processing plant number (integer), 6) the tax rate on the order (double), 7) the net cost of the purchase order (double), 8) the purchase order tax (double) and 9) the total cost of the purchase order (double). **Your program will have 4 functions: input, process, output, and tally_inventory.** Your main program will call (invoke) each function. Following are the descriptions of the functionality of each function:

1. The void function "***input***" will have two parameters: the order record array called 'INV", and "count". The capacity (SIZE) of INV should be initialized to 50, and "count" should be initialized to 0. The input data file will be opened and closed inside this function. The function will read all the data until the end-of-file has been reached; the cell_number (customer cell phone number), item_number (item identification number), quantity (number of items in purchase order), price (cost of one item) and processing_plant (processing plant identification number), will be read into the array of order_records (INV) from the data file. Remember, "count" should be passed by reference.

2. The function ***"process"*** have two parameters: INV and count. The function will calculate the order tax rate ***(tax_rate),*** net cost of an order ***(net_cost)***, the tax on an order ***(order_tax)*** and the total cost of an order ***(total_cost)*** using the quantity, cost of an item (price), and processing plant identification number (processing_plant) for a purchase order record (order_record). Remember, we are using an array of records. Please consider the following:

   a) The ***tax rate on an order (tax_rate)*** is simply based on the processing plant identification number ***(processing_plant)*** which is where the order was processed (0<= ***processing_plant*** <= 50 then ***tax_rate*** = 6%; 51<= ***processing_plant*** <=110 then ***tax_rate*** = 7%; 111<= ***processing_plant***<=200 then ***tax_rate*** = 8%; 201<=***processing_plant***<=500 then ***tax_rate*** = 9%; ***processing_plant*** >500 then ***tax_rate*** =11%) .

   b) The tax on an order (order_tax) is calculated by the following formula:

      order_tax = (quantity) x (price) x [(tax_rate) / 100]
      //Hint: INV[i].order_tax = (INV[i].quantity * iNV[i].price ….

   c) The net cost of an order (net_cost), which does not include tax, is calculated by the following:

      net_cost = (quantity) x price

   d) The total cost of an order (rounded to the nearest houndredth) is calculated by the following formula:

      Total_cost = net_cost + order_tax
      All tax and cost calcuations should be rounded to the nearest hundredths.

3. The function ***"output"*** will print every field of an order record stored in the array INV. The output data file will be opened and closed inside this function. The function will print every field of every order record stored in INV. The fields should be printed in the following order: 1) cellphone number, 2) item_number, 3) quantity, 4) price, 5) processing_plant, 6) tax rate, 7) order tax, 8) net cost, and 9) total cost of order. See the sections below called "Input St ream" and "Format of Output for the function output" for more information. See the section "Format of Input Data File (input filename is "purchase_data.txt")".

4. The double function ***"tally_inventory"*** which will calculate the average of the "total order cost". The average should be rounded to the nearest hundredth and returned to main. ***"tally_inventory"*** will be the last function your main program calls. The main program will print the following message after output have executed: (Note: the average appears as XXXXXX.XX to represent a double value)

   Average Total Order Cost = XXXXXX.XX

It is always a good to start with the skeleton program I provide. **See "amazon_porders5.cpp".** Remember to follow all style rules and to include all necessary documentation (consistent, indentation, proper variable names, pre/post condition, program header, function headers, and so forth.)

Electronically submit ***"amazon_porders5.cpp"*** in the Assignments area of Canvas before the due date and time. (Note: ***purchase_data.txt*** should be in the same directory as ***amazon_porders5.cpp).***

Remember, no late assignments will be accepted.

Output Format for the Function "output":

Consider the following sample output table when designing and implementing the function "Output".

(The output is in the following order: cell phone number, item number, quantity, price, process plant, tax rate, order tax, net cost, total order cost)

9546321555	452-XLY	2	70.82	503	11	15.58	141.64	157.22
9546321555	742-6Z3	300	10.14	47	6	182.52	3042.00	3224.52

Input Stream:

In the assignment you will declare one **ifstream** to bind your input to the file ***"purchase_data.txt"*** Whenever a program performs ***file i/o*** you must include the **"fstream"** library. Add the following statements to your program:

For source file, *"amazon_porders.cpp"*:

Add "**#include <fstream>**" to your # include statements in your source file. Add **"include <string>** to your # include statements in your source file.

Format of the input data file(input filename is "*purchase_data.txt*"): Do not include column titles

(cell phone number, item, quantity, price, processing plant)

9546321555	452-XLY	2	70.82	503
9546321555	742-6Z3	300	10.14	47
5612971340	924-YUT	8	23.00	51
3051234567	9L3-11T	47	50.12	200
7542346622	453-TTT	92	25.25	110
3054432762	213-ABC	15	105.02	111
9544321011	321-XMZ	112	32.41	92
8776219988	444-SSS	100	12.12	86
9042224556	290-P23	297	9.62	500
7877176590	23Z-00Q	32	9.62	201
5617278899	893-42T	642	0.81	72
9546321555	008-LL5	90	45.80	18
5612971340	452-XLY	42	99.99	503
3051234567	742-6Z3	14	10.14	47
7542346622	742-6Z4	13	66.48	3
3054432762	742-6Z5	209	0.77	2
9544321011	742-6Z6	108	1.17	1
8776219988	213-ABC	67	105.02	111
9042224556	321-XMZ	500	32.41	92
7877176590	444-SSS	43	12.12	86
5617278899	290-P23	16	9.92	500
8776219988	23Z-00Q	82	9.62	201
9042224556	23Z-00Q	87	9.62	201
7877176590	23Z-00Q	445	9.62	201

Format of Output:

(cell phone number, item, quantity, price, processing plant, tax rate, order tax, net cost, total order cost)

The output file should be called "**purchase_results5.txt**", and contain the following output. Your output should not contain any titles, but the output must be in the proper order as stated in the assignment.

9546321555	452-XLY	2	70.82	503	11	15.58	141.64	157.22
9546321555	742-6Z3	300	10.14	47	6	182.52	3042.00	3224.52
5612971340	924-YUT	8	23.00	51	7	12.88	184.00	196.88
3051234567	9L3-11T	47	50.12	200	8	188.45	2355.64	2544.09
7542346622	453-TTT	92	25.25	110	7	162.61	2323.00	2485.61
3054432762	213-ABC	15	105.02	111	8	126.02	1575.30	1701.32
9544321011	321-XMZ	112	32.41	92	7	254.09	3629.92	3884.01
8776219988	444-SSS	100	12.12	86	7	84.84	1212.00	1296.84
9042224556	290-P23	297	9.62	500	9	257.14	2857.14	3114.28
7877176590	23Z-00Q	32	9.62	201	9	27.71	307.84	335.55
5617278899	893-42T	642	0.81	72	7	36.40	520.02	556.42
9546321555	008-LL5	90	45.80	18	6	247.32	4122.00	4369.32
5612971340	452-XLY	42	99.99	503	11	461.95	4199.58	4661.53
3051234567	742-6Z3	14	10.14	47	6	8.52	141.96	150.48
7542346622	742-6Z4	13	66.48	3	6	51.85	864.24	916.09
3054432762	742-6Z5	209	0.77	2	6	9.66	160.93	170.59
9544321011	742-6Z6	108	1.17	1	6	7.58	126.36	133.94
8776219988	213-ABC	67	105.02	111	8	562.91	7036.34	7599.25
9042224556	321-XMZ	500	32.41	92	7	1134.35	16205.00	17339.35
7877176590	444-SSS	43	12.12	86	7	36.48	521.16	557.64
5617278899	290-P23	16	9.92	500	9	14.28	158.72	173.00
8776219988	23Z-00Q	82	9.62	201	9	71.00	788.84	859.84
9042224556	23Z-00Q	87	9.62	201	9	75.32	836.94	912.26
7877176590	23Z-00Q	445	9.62	201	9	385.28	4280.90	4666.18