CIS 1057 Fall 2020 Lab 11

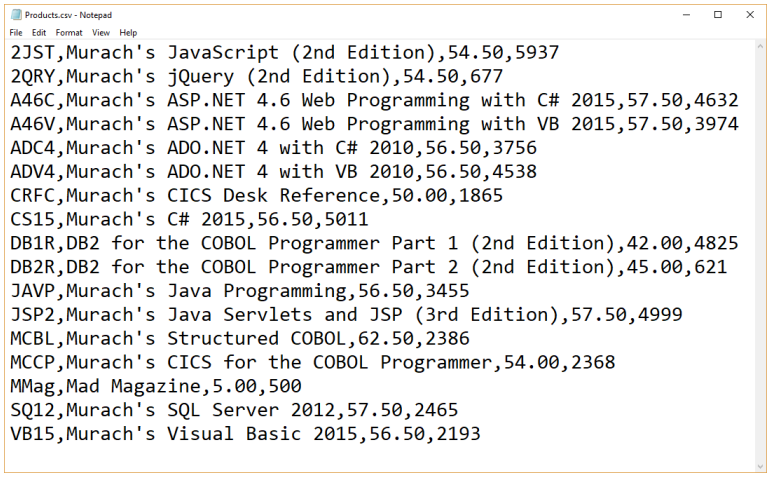
This lab requires you to read two data files, print their data to the screen, process the data in the files, print a report to screen and write a report to a file, and update the products in inventory. Create a c file that imports the stdio, stdlib and string libraries and add a main function. Define constants: MAX_PRODUCTS = 25, MAX_LINE_ITEMS = 200 and MAX_READ_CHARS = 200.

Reading products

The first file, products.csv, is a comma separated values file. This means that the individual fields for a record are separated by commas and each record is on its own line. To read the file, you need something to contain the data in your application. A struct can be used to contain all the product data in one data structure. Your main function will declare a vector of product structs and an integer to record how many products were read from the file.

The products.csv file

The image below shows the product.csv file contents. The first datum is a string product_code and is the unique identifier for a product. The second datum is a string description that describes the product. The third is the double price of a product. The fourth is an integer quantity that records the number of products available for sale.



The product_t struct

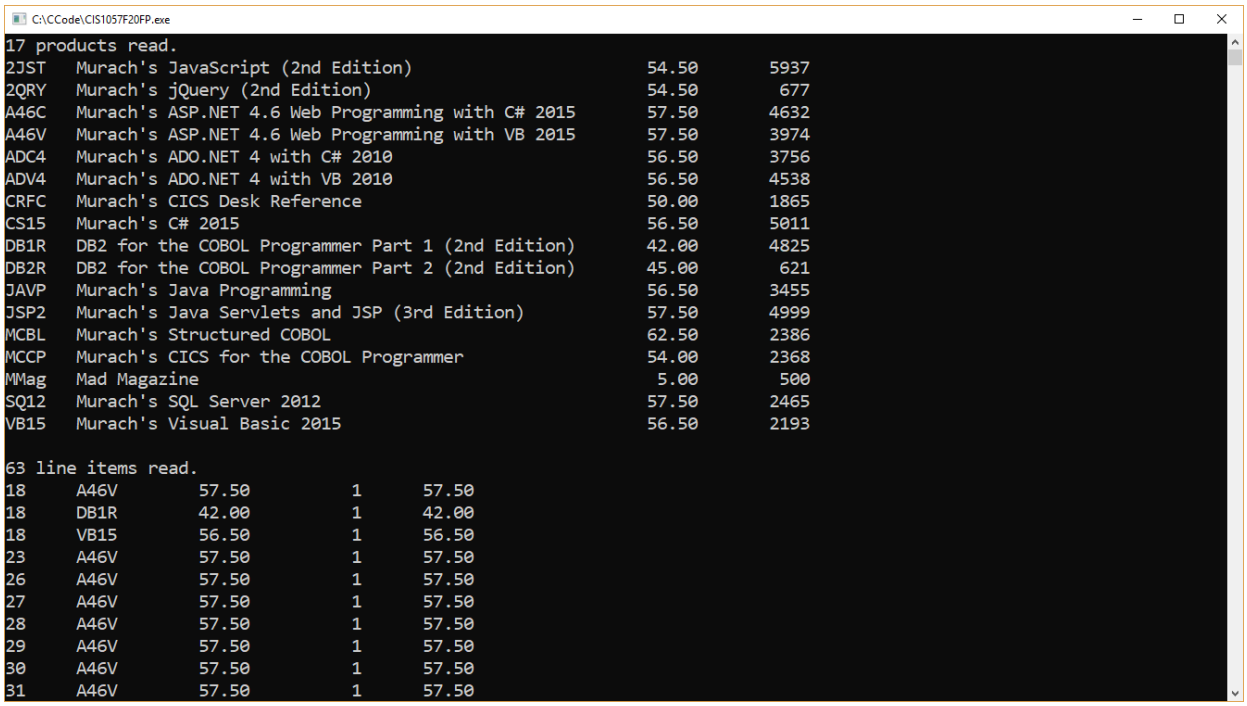
The product_t struct needs members to contain the data for the file described above. It should have a string product_code (size of 10 characters), string description (size of 50 characters), a double price and an integer quantity as members. Make sure that you make the struct a data type so you can declare it as: product_t p;

Reading the products

Write a function called read_products that accepts a vector of product_t pa and the filename to read and returns the number of products read. The call to read a line should read MAX_READ_CHARS characters, which should be the size of the string buffer to store a line read from the file.

Printing products

Write a function called print_product to print a product to screen. It should accept product_t and return nothing. Write a function called print_products that calls repeatedly calls print_product to print all products to screen. It should accept a vector of product_t and the number of products in the vector. The output should look like the image below. Note that the columns are a fixed width (Hint: the first column's format is %-6s).



In main

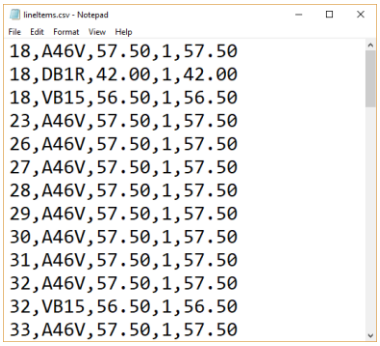
Write code in main to declare a vector of product_t called pa with MAX_PRODUCTS elements. Declare an integer pa_count to receive the number of products read from the file and call read_products to read the file. Print the number of records read to screen. Print the vector of product_t to screen. **Test your product functions and fix any errors.**

Reading line items

The second file, lineItems.csv, is also a comma separated values file. You need to write a struct and three functions.

The lineItems.csv file

The image below shows the lineItem.csv file contents. The first datum is an integer invoice_id and is the unique identifier for the invoice that contains the line item. The second datum is a string product_code and is the unique identifier for a product. The third datum is a double price of a single product. The fourth is an integer quantity that records the number of products purchased. The fifth is the double total of all product in the line item.



The line_item_t struct

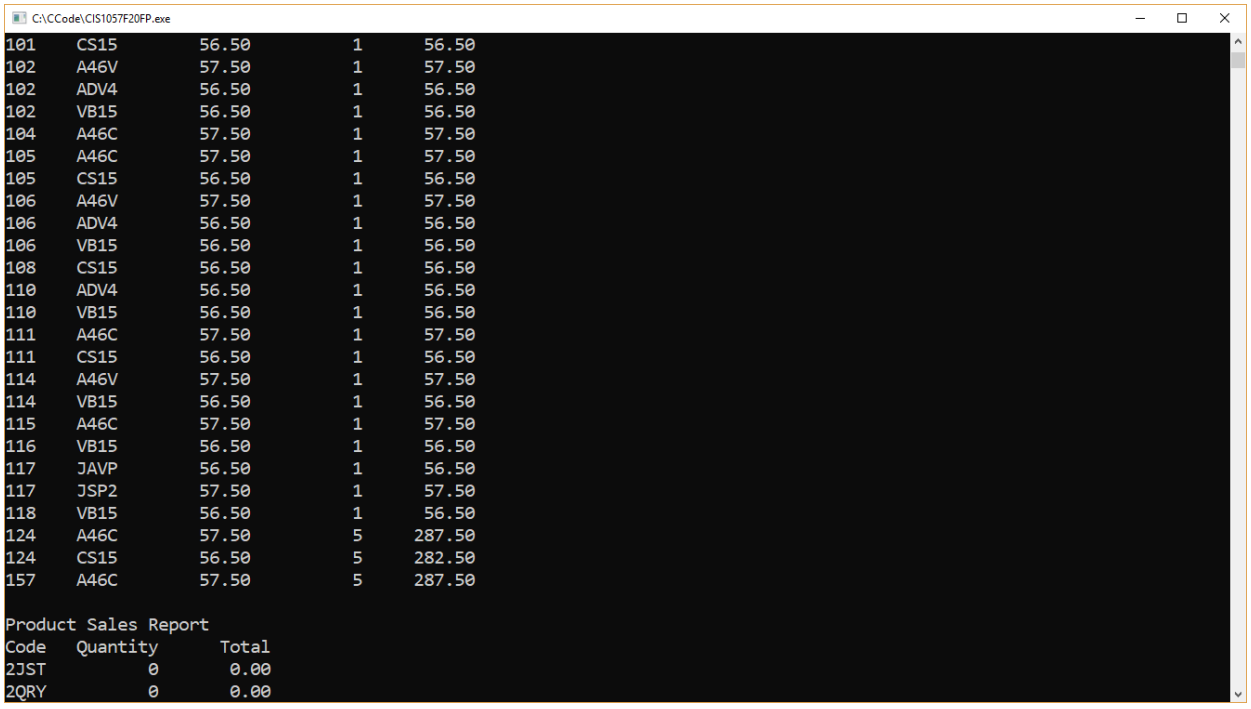
The line_item_t struct needs members to contain the data for the file described above. It should have an integer invoice_id, a string product_code (size of 10 characters), a double price, an integer quantity and a double total as members. Make sure that you make the struct a data type so you can declare it as: line_item_t l;

Reading the line items

Write a function called read_line_items that accepts a vector of line_item_t la and the filename to read and returns the number of line items read. The call to read a line should read MAX_READ_CHARS characters, which should be the size of the string buffer to store a line read from the file.

Printing products

Write a function called print_line_item to print a line item to screen. It should accept line_item_t and return nothing. Write a function called print_line_items that calls repeatedly calls print_line_item to print all line items to screen. It should accept a vector of line_item_t and the number of line items in the vector. The beginning output is shown in the screen output above and the last lines are in the image below.



In main

Write code in main to declare a vector of line_item_t called la with MAX_LINE_ITEMS elements. Declare an integer la_count to receive the number of line items read from the file and call read_line_items to read the file. Print the number of records read to screen. Print the vector of line_item_t to screen. **Test your line item functions and fix any errors.**

Processing the data, printing, and writing the report

This is the challenging part of this lab. You need to write code to generate a report that tallies the number of each product sold and the revenue for each item.

Clearing vectors to zeros

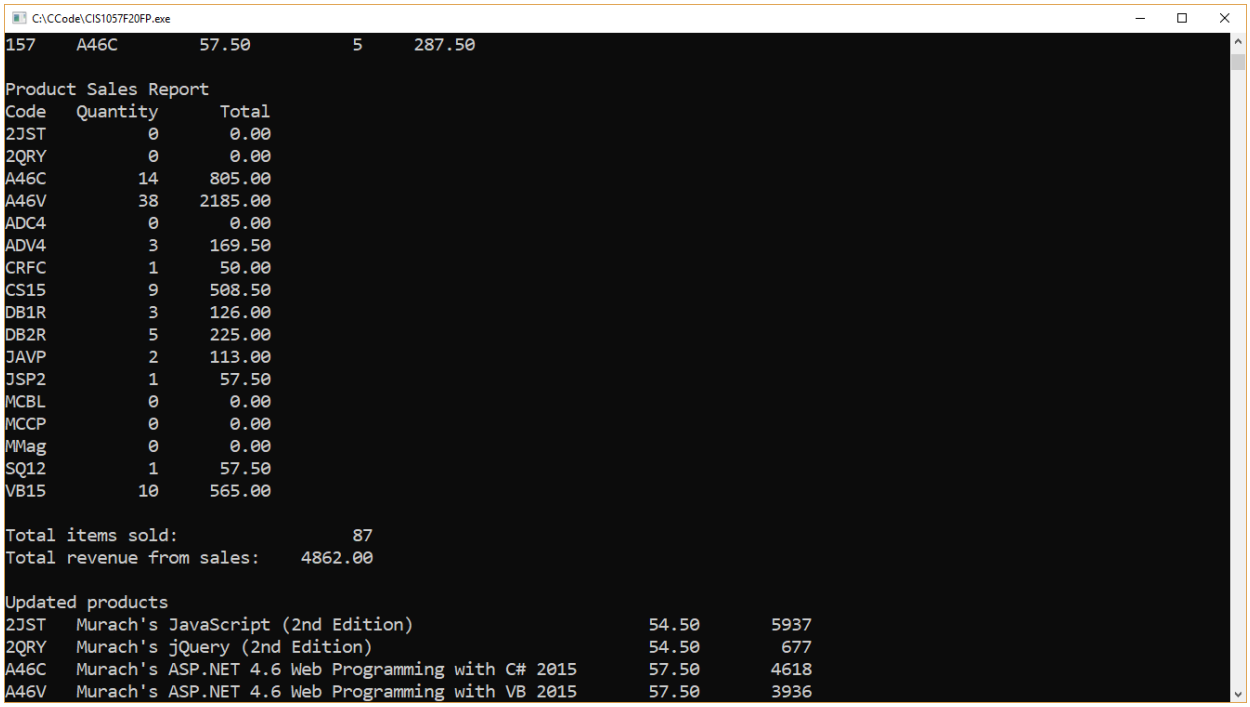
Write a function called zero_quantities to accept a vector called quantity_sums of integer and an integer called pa_count that contains the number of elements in the vector that writes zeros to all elements in the vector. Write a function called zero_totals to accept a vector called total_sums of double and an integer called pa_count that contains the number of elements in the vector that writes zeros to all elements in the vector.

Calculating the report

Write a function called calc_report that accepts a vector of product_t called pa, an integer called pa_count to contain the number of products in pa, a vector of line_item_t called la, an integer called la_count that contains the number of line items in la, a vector of double called total_sums and a vector of integer called quantity_sums. The total_sums and quantity_sums vectors have the same number of elements as pa. There is pa_count elements in each of these. Write a nested loop to calculate the sum of each product sold and the total revenue for each product. Remember that product_code is a string and cannot be compared with ==. I suggest iterating through line items in the outer loop and products in the inner loop.

Printing the report

Write a function called print_report that accepts the vector of product_t pa, an integer pa_count that contains the number of products read, a vector of double total_sums and a vector of integer quantity_sums. Declare and initialize an integer called tot_quantity to aggregate the sum of quantity_sums and a double to aggregate the sum of total_sums. The image below shows the output for this function. Write a title for your report and the column titles to screen. In a loop, print the product_code, total quantity and total price to screen for each product, and aggregate the sums for quantity_sums and total_sums. Write the lines with tot_quantity and tot_price to screen. **Test this function and calc_report and fix any errors.**



Writing the report

Write a function called write_report that write the same data as above to a file called report.txt. Hint: copy print_report and paste it and modify the code to write to a file. This is easier than you think (printf to fprintf).

In main

Write code to declare a vector of integer with pa_count elements called quantity_sums and pass it and pa_count to the zero_quantities function to put 0 in all elements. Write code to declare a vector of double with pa_count elements called total_sums and pass it and pa_count to the zero_totals function to put 0.0 in all elements. Call calc_report and pass pa, pa_count, la, la_count, total_sums and quantity_sums. Call print_report and write_report to print the report to screen and write it to a file called report.txt. **Test your code and fix any errors.**

Updating and writing products

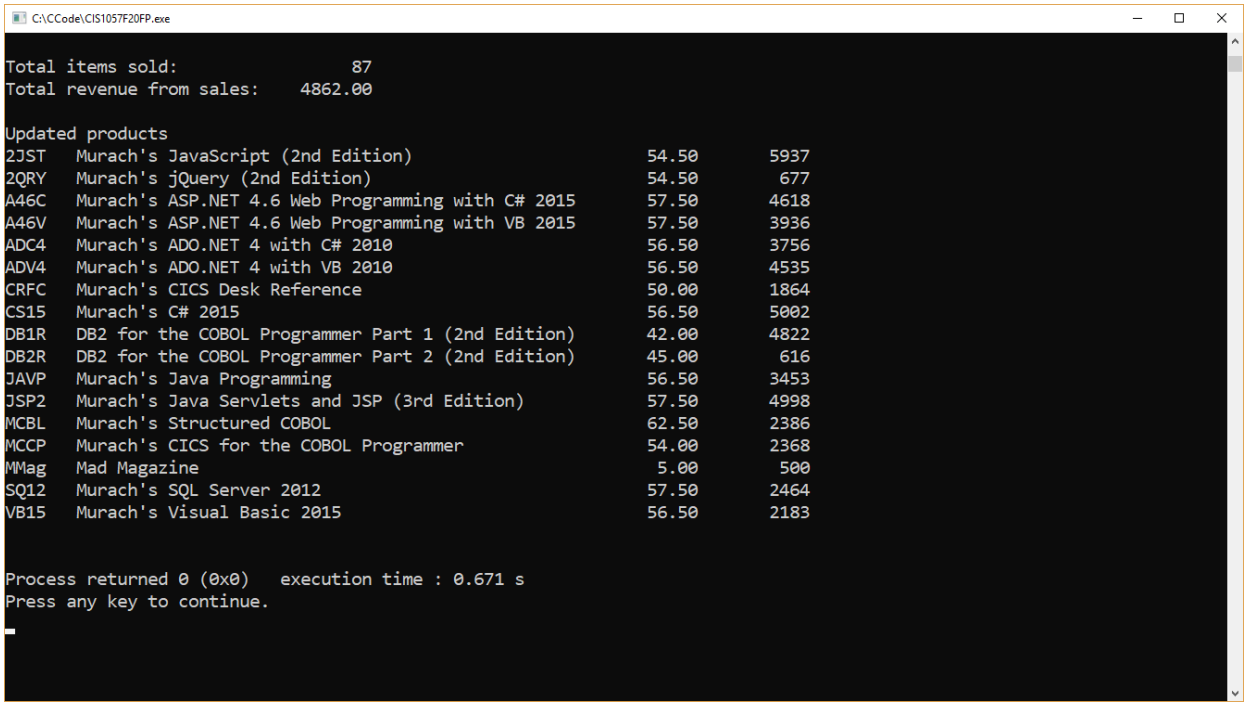
The last task is to update the products file's quantity column to subtract products sold from inventory.

Updating products

Write a function called update_products that accepts the vector of product_t pa, an integer pa_count that contains the number of products read, and a vector of integer quantity_sums. Write a loop that subtracts the products sold from quantity.

Writing to file

Write a function called write_products that accepts the vector of product_t pa, an integer pa_count that contains the number of products read and a string with the name of the output file. Write



In main

Call update_products to subtract products sold, print the vector pa to screen with print_products and call write_products to save the report to a file called newProducts.txt. **Test your code and fix any errors.**

Submit your lab to Canvas

Upload your c file to Canvas.