CSC 407/507 Programming Assignment #2: The OLDE Language Assignment
Due Date: Wednesday, February 28

Download a compiler for one of the following languages: COBOL, FORTRAN (IV if possible, FORTRAN 77 if you can't find a working FORTRAN IV compiler, DO NOT use FORTRAN 90 or later), ALGOL 60, PL/I (good luck finding a working compiler, I searched and couldn't find one for Windows, there does appear to be one available to work with gcc in Linux) or ALGOL 68. Learn enough of the language in order to implement the following assignment. Use modularity as much as possible within the limitations of the given language.

Write a transaction processing program as follows. The program will input data from 3 files: inventory, customers, transactions and generate three output files (reports): transactions processed, inventory orders, errors. The inventory file consists of product numbers, product names, number currently in stock, minimum number to stock (reorder point), and price (in the form xx.xx). The customers file consists of customer numbers, names, addresses (street, city, state or country) and the amount the customer currently owes from previous orders (in the form xxx.xx). The transactions file consists of a number of transaction records, each consisting of customer number, product number, number of that product ordered and sale code (sale codes are described below).

Your program will first need to input the inventory and customer files and store them in your arrays. You can declare your arrays to be of size 24 and 10 respectively, or larger if you prefer. After inputting both files into their corresponding arrays, the program will then process the transactions in the transaction file, one at a time as follows. If the customer number does not match a valid customer or the product number does not match a valid product, generate an entry in the error report file listing the customer number, product number and number ordered, and the type of error (invalid customer number or invalid product number). If there was no error, then process the transaction by computing the gross cost (cost of inventory item * number ordered), the discount based on the sale code (if any), and the net cost (gross – discount). Update the corresponding entry in the customer array by adding the net cost to their amount owed value. Update the corresponding entry in the inventory array to reduce the number in stock. If the number in stock drops to either reach or become smaller than the reorder point, output to the inventory orders file the item number and the quantity to order. The number that you need to order is described below. Finally, output this transaction to the transactions processed file which should list the customer's name and address, item purchased and quantity, gross cost, discount and net cost, and the amount the customer now owes (the customer may have owed money prior to the transaction). Upon completion, the program should close all files. Note that while the inventory and customer arrays have changed, you will not need to update these files (they should be updated, but we will skip this step).

For COBOL, ALGOL or PL/I, define record types for customers and inventory, create arrays of these types and access these data structures when handling each transaction. In FORTRAN, you will have to use several arrays (known as parallel arrays where the same index i is used to access each element of the same inventory item or customer).

Sales codes are: A for 10% off, B for 20% off, C for 25%, D for "buy at least 3, get 1 free", E for "buy 1 get 1 free" (if say 6 are bought, only pay for 3 of them) and Z for no discount.

The number of items to order when the current inventory drops to or below the reorder point is as follows:

If reorder point is 1, then order enough to have 3 in stock
If reorder point is 2-5, then order enough to have 6 in stock
If reorder point is 6-10, then order enough to have 12 in stock
If reorder point is 11-20, then order enough to have 25 in stock
Otherwise, order enough to have 30 in stock

For instance, if there are 8 units of an item whose reorder point is 4, and 6 are purchased, then the number in stock would drop to 2, you would then have to order 4 more to have 6 in stock.

The output of your program will be the three output files. Again, ignore the fact that the input files will be out of date after running your program because of changes made to both inventory and customer information. We just won't worry about that.

The three initial files are on the handouts webpage. Run your program on these files. NOTE: If you using FORTRAN IV (or I), then you will not have strings. In such a case, edit the files to remove any strings and use only part number and customer number in your output rather than names/descriptions/addresses.

Upon completing the assignment, submit your source code. Keep all parts of your program (including all subroutines) in a single file for your and my convenience. Remember to comment your code well! Along with the source code, submit the three output files (errors, transactions processed, orders) and a report that answers the following questions. Remember to submit all of this as hardcopy.

1. Who implemented which portions of the program in your group?
2. How hard was it to learn the language and implement the program?
3. What features of the language made your implementation difficult or awkward?
4. Were there features of the language that were adequate or even useful?
5. Were there features of the language that you used in your program that are missing or not as easy to use in a language like C++ or Java and if so, what features?
6. Did you learn anything useful about modern programming languages from this assignment?