Final Programming Proficiency Test

Due by the end of term, 5:30 pm on Wednesday, December 18, 2019.
No late submissions will be accepted.

1 Instructions

This take-home test is in lieu of a regular final exam. It is due at the end of the final exam period.

The two in-class hour exams tested your understanding of object-oriented programming and C++ concepts. This is a test of your ability to put that knowledge into practice. The two hour exams and this test will each count for 17% of the course grade.

Because this is an exam, you are expected to complete it on your own. You are *not* permitted to work in groups, to ask others for help, nor to post questions to internet help sites. However, you may use the internet to view course materials and other references. All of the exams will be put through a program to detect collusion and plagiarism. You are of course free to ask questions of the instructor or TA for any parts that are not clear.

2 Problem

A company sells many different kinds of parts. Each part has three attributes: its name, the quantity on hand, and the selling price. The company's inventory is kept in an unsorted file of parts.

The data in the inventory is manually updated from time to time. Unfortunately, this can result in multiple entries for the same part, sometimes with different prices.

Your job is to write a program to repair the inventory as follows:

1. After the repair, there should be only one entry for each part.

2. If the original file contains multiple entries for the same part, that part's quantity on hand in the new file should be the sum over those entries of the quantities on hand in the old file. The price of that part in the new file should be the maximum price of that part in the old file.

3. The repaired file should be sorted alphabetically by part name.

3 Requirements

The input file is free-format and consists of a sequence of whitespace-separated parts. Newline characters are treated as whitespace but are otherwise ignored. A part consists of a triple of whitespace-separated fields: a string part name, an integer quantity on hand, and a double price. The part name does not contain embedded whitespace.

set precision

The output file should have each part on a separate line. The fields comprising a part should be separated by a single space. The price should be printed as a decimal number with two digits after the decimal point.

Your program should have three classes, `Part`, `Inventory`, and `Controller`, and one exception class `IOerror`.

Part Contains three fields describing a part: name, quantity, and price.

Inventory Contains a list of parts, represented by a `vector`, as well as functions for reading and writing inventory files and for repairing the parts list.

Controller Controls the repair process: Creates an empty `Inventory` object, reads the old inventory file into it, sorts it by part name, and creates a new `Inventory` object to hold the repaired inventory. Next, it makes a pass over the old inventory list, merges together each block of entries for the same part, and pushes the resulting parts onto the back of the new inventory object. Finally, the new repaired inventory list is written back to disk. Reports any errors encountered.

The user command `repair` takes two command line arguments, the input file name and the output file name.

The `main.cpp` file should read the command line arguments, instantiate the controller class with them, and call a function `go()` in the controller. It should also catch an `IOerror` exception and call the `what()` function in that class to print an informative message before terminating the program.

You should use `std::sort()` to sort the inventory. For this, you must write a functor `CompareByName` that will order two `Part` objects alphabetically based on a string comparison of their names.

You should *not* use `tools.hpp` and `tools.cpp` in this test.

Your code should follow all of the good coding practices taught in this course, including (but not limited to) the General OO Principles on slide 4 of lecture 24.