CISC 3130 **Ari Mermelstein**

Homework #4:

In this assignment you will be asked to implement a simulator for a hardware store. (This will contain a number of classes and a lot of moving parts, so please ask a lot of questions.) This assignment will require you to use the STL vector or STL list (your choice), in addition to the STL queue. You will also need to use your object oriented programming skills to create clean, maintainable code.

(**N.B.** If your code is not documented, you will not receive credit. This means that if you write any line of code that isn't completely trivial, you must explain it to me.
You may not use any code or any function that I have not discussed in class. Any fancy algorithm you use must be written by you or discussed with me beforehand. I have been getting some interesting code for which ascertaining its correctness is almost impossible.)

You will create the following classes. I will provide prototypes below:

An Item class that models the name and price of something that is sold in a hardware store. For example, a hammer that costs $10.50.

This class must support the following operations:

Item(**const string&** name, **double** price);
A constructor that takes in the name and price as input.
Item();
An empty constructor. (This is so the vector or list gets initialized properly)

Of course, you can implement these as one constructor.

string getName();
This returns the name of the item
double getPrice();
This returns the price of the item.

ostream& operator<< (**ostream&** os, **const Item&** item);
This prints out the representation of an item.

You may implement this class (and only this class) inline as only a .h file if you want.

A ShoppingCart class that contains a vector (or a list) of Items.

The ShoppingCart class must implement the following functions.

ShoppingCart();
This creates an empty shopping cart.

void addItem(**const** Item& item);
This adds an item into the shopping cart.

double grandTotal();
This returns the total price of all of the items in the shopping cart.
int numItems();
This returns the number of items in the shopping cart.

ostream& operator<<(**ostream&** os, **const** ShoppingCart& sc);

This prints a representation for a ShoppingCart.

A Shopper class, which contains a name and a ShoppingCart. It must implement the following functions.

Shopper(**const** string& firstName, const string& lastName);
This creates a shopper with this name and with an empty ShoppingCart.

void addItemToCart(**const** Item& item, **int** numItems=1);
This adds *numItems* items of the same type to the shopping cart. If no second parameter is specified, only one item should be added.

int amountOwed();
This is the amount of money in cents that Shopper needs to pay the cashier in the hardware store. It is the total cost of the items in the cart plus an additional 8.875% tax. The

price should be rounded to the nearest cent. (reminder: the ceil function in the <cmath> header file, rounds up a double to the nearest integer).

ostream& operator<<(**ostream&** os, **const** Shopper& shopper);
Prints the representation for a shopper.

And finally, a HardwareStore class, that models the actual store. Each store has a vector of Items that is its inventory (which you can turn into its own class if you want to, for extra credit), an amount of money taken in by the store, and a vector of queues. (i.e. A bunch of checkout lines).

It must have the following functions:

HardwareStore(**int** numRegisters);
Creates a HardwareStore with the specified number of registers. (i.e. the number of queues).

void addShopperToLine(**const** Shopper& shopper);
Adds a shopper to the line that currently has the least number of people waiting on it.

void processShopper();
Processes the shopper at the front of the longest line.
void checkoutAllShoppers();
Processes all Shoppers still on line for all of the lines.

double totalRevenue();
Returns the amount of money the HardwareStore has taken in so far. (At the end of the program, it will represent the total revenue of the store.)

Your main program will read in 2 files: an inventory file and an event file.

The inventory file will have a number of lines of the form:

itemName itemCost

The event file will have a number of lines of either the form

P (which means process) followed by no additional information.

Or

S *n item1 numberOfPiecesOfItem1 item2 numberOfPiecesOfItem2 … itemN numberOfPiecesOfItemN*

You must write a main program that will read in the inventory file and store that information in the HardwareStore's inventory vector. Then, it will read in the event file and process the shoppers in it.

You must submit the following files:
Item.h (Item.cpp optional)

ShoppingCart.h	ShoppingCart.cpp
Shopper.h	Shopper.cpp
HardwareStore.h	HardwareStore.cpp
(Inventory.h	Inventory.cpp if you decide to go this route)
StoreSimulator.cpp	

Sample files are included on the course website.

Please add some more entries to both files.