Project 5 — Online Groceries

PART 1

CS 2370

Background

You are building a system to allow customers to order groceries. You have two files with data:

- *customers.txt*
- *items.txt*

The first file holds customer data in a single text line the following:

810003,Kai Antonikov,31 Prairie Rose Street,Philadelphia,PA,19196,215-975-7421,kantonikov0@4shared.com

The data fields are the customer id, name, street, city, state, zip, phone, and email.

The file *items.txt* has fields item_id, description, and price:

57464,Almonds Ground Blanched,2.99

Requirements

Write a program that does the following:

1. Read “customers.txt”. Put the customer records into a (global) vector.
2. Read “items.txt”. Put the item records into a vector.
3. Display a message stating how many customers there are, and how many items there are.
4. Prompt the user for a customer number. Search for the customer record.
    a. If the customer is not found, exit the program with an appropriate message.
    b. (You may assume that the user always enters a number.)
5. Prompt for the item number to purchase.
    a. Display the item description and price.
    b. Keep track a running totals of items bought, and amount spent.
    c. If the item is not found, display an appropriate message and continue.
    d. (You may assume that the user always enters a number.)
6. Continue asking for items until the user enters a 0 for the item number.
7. Display a message with the number of items purchases and the total cost.
    a. The total cost should be displayed as a typical money number with a $ sign and dollars and cents. For example, if the total is $2.00, display it as $2.00. Not $2. This means you will have to use a bit of output stream formatting.
8. (The program calculates only one customer’s order. Do not repeat the steps 4-7.)

Put all your code in a file called groceries.cpp.

Use this **main** function:

```
int main() {
    read_customers("customers.txt");        // Step 1 above
    read_items("items.txt");                // Step 2 above
    one_customer_order();                   // The rest of the steps
}
```

NOTE: You will have to declare classes and customers and items. Because they are simple data structures, and you WANT all the data to be available, you can just use a struct for each (which is the same as a class with all public members.)

Design considerations:

1. Where do you put the functions read_customers(…), read_items(…), and one_customer_order(…)?
    a. Put them is the groceries.cpp file before main().
2. What goes into the Customer class (or struct)? You can see a customer record has:
    a. Customer id (which is an integer)
    b. Name
    c. Street address
    d. City
    e. State
    f. Zip code: even though it is a number, store it as a string.
    g. Phone number: also a string -- just keep the dashes
    h. Email address
3. What goes into an Item?
    a. Item Id (an integer)
    b. Description
    c. Price: This has to be a number, and not an integer! Store it as a double.
4. How do you read in a record and create an object from it? See note below on a function called split() that we will provide for you.
5. Be sure to test your program thoroughly!
    a. Be sure the number of customers and items are correct.
    b. Not only test a "happy path" but consider all the ways things could go wrong. For example, what if the grocery item isn't found?

To Turn In

1. Your source code, in a single file called groceries.cpp
2. A UML diagram, showing the two classes you wrote. You can hand draw it (neatly!) or use a tool to draw it. A simple free online UML drawing tool is called Violet.
3. A list of the test cases you used to verify that your program works correctly.

Implementation Notes

I have provided a file, *split.h*, which contains a **split** function that returns all fields that were separated by some character as a vector of strings:

```
#include "split.h"

#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main() {
    string s = "715608,Vergil Heelis,61070 Marcy Park,Fort Worth,TX,76115,682-583-
7160,vheelis4@blogger.com";
    auto fields = split(s, ',');
    for (const auto& fld: fields)
        cout << fld << endl;
    cout << endl;
}

/* Output:
715608
Vergil Heelis
61070 Marcy Park
Fort Worth
TX
76115
682-583-7160
vheelis4@blogger.com
*/
```

You can also use this function to separate the `item_id-quantity` pairs using a dash as the split character.

FAQs

Q. How do I find the Customer or Item entries in the global arrays given a cust_id or item_id?
A. You'll have to search the vectors for them. I added the following functions to my solution:

```
int find_cust_idx(int cust_id);
int find_item_idx(int item_id);
```

NOTE: if you wish to use pointers, you may return a pointer to the items instead of the index into the vector of each.

Q. I heard global data is "evil".
A. It can be problematic, but it makes this project doable in the time allotted by keeping things simple. In production we would probably have a Grocery namespace and the global data would be

defined there, but that's an implementation detail not worth worrying about here. Namespace/static data is stored the same way as global data—only the access is different.

Q. Why didn't you make all the data of type **std::string**?
A. Well, prices must be **double**s so you can do arithmetic. Also, I want you to use **std::stoi** and **std::stod** for practice.

Q. Why is the program divided like it is?
A. The next project builds on this project. You will include *read_customers*() and *read_items*() and the find functions just as they are specified here.

Q. Why don't I display the count of customers and items inside of read_customers() and read_items()?
A. Refer to the previous question.