Programming Assignment 3: Sally's Awesome Shirts, Inc.

COSC 1020: Computer Science 1

Fall 2023

Dates

Assigned: 19 October 2023

Due: 3 November 2023

Contents

1 Learning G

In order to complete th

- Apply the skills le
- Create a menu-dri
- Decompose progra
- Store data in parallel vectors

2 Project Description

SASI continues to grow to the point that orders must be divided into regional record-keeping. Now you may have multiple text files (containing sales from different regions) which Sally may need to load, display, and summarize in any combination. She asks you for a menu-driven program to let you perform multiple operations without exiting the program.

3 Specification and Requirements

3.1 Specification

The major changes between this project and the last are the menu-driven interface and the method of storing order data in parallel arrays. Individual order validation and cost calculation have not changed.

Menu-driven user interface

The user will interact with your program through a menu-driven interface. The menu must provide the following user options (and request the indicated character to select them):

- (U) Upload a regional sales data file
- (A) display All loaded data details
- (M) display a summary by print Method
- (C) Clear all data
- (Q) Quit

Your program should p er corresponding to the option they want. Both program should ignore any "extra" characters

If the user selects one c vior (see below); if they type an invalid option, both cases, the program should "return to the m prompting for another selection. This should c gram should end.

The work of printing th displayMenu() function described in the require

A brief description of ea

(U) Upload a region

This option prompts the he file. If the file cannot be opened, an error mes file can be opened, the file is read line-by-line a rk done in Project 2).

If a line contains valid (see below); if the line contains invalid data th lid lines should be kept and printed once the wh

The output printed during this option should be the same as that of Project 2 (including a summary of the costs of valid orders read from this file). The only significant change from the work of Project 2 is that the valid orders are stored in the program.

The work of this menu options will be performed by the uploadFile() function described in the requirements section.

(A) display All loaded data details

This option prints the order details of each order currently loaded into the program. The program should only perform this behavior if at least one valid order has been loaded; if this is not the case then the program should print an error message explaining the problem and return to the menu.

The output printed during this option should be the same as the line-by-line output from Project 2. Note that, because the program only stored valid lines of data, this option will only encounter valid records (and you will not need to re-check their validity).

The work of this menu option will be peformed by the allDetails() function described in the requirements section.

(M) display a summary by print Method

This options prints a summary total of the costs of all orders currently loaded into the program. The program should only perform this behavior if at least one valid order has been loaded; if this is not the case then the program should print an error message explaining the problem and return to the menu.

The output printed during this option should be the same as the summary output from Project 2. Note that, because the program only stored valid lines of data, this option will only encounter valid records (and you will not need to re-check their validity).

The work of this menu option will be performed by the summaryByMethod() function described in the requirements section.

(C) Clear all data

This option clears all da , menu options (A) and (M) should not be possi rror to select this menu option when no data is

(Q) Quit

This option displays a s option (both valid and invalid) which does not

Data storage in para

The program will maint es for valid shirt orders. These vectors will be l arameters to the other functions you write.

You will need a total of

- A `vector<string`
- Three `vector<int`
- A `vector<char>`
- A `vector<string>` to store the message
- Three `vector<int>` to store the shirt counts for all three sizes
- Three `vector<string>` to store the shirt color; ink color; and customer name and email for the order.

Most of the required functions receive these vectors as inputs through reference parameters, which is how they are used and/or modified. You should make these vectors local variables of `main` and **not** global variables.

Shirt order file name and format

The program determines the region which an order belongs to by examining the filename that order is loaded from. The program will keep track of five region names: `North`, `South`, `East`, `West`, and `Other`.

When the file contains orders from one of `North`, `South`, `East`, or `West`, then the filename will begin with the region; file names that contain none of these should be categorized as `Other`.

The string member function `find(string&, int)` may be useful in determining which region the file describes. This function searches for a substring (the first argument) beginning at or after the indicated position (the second argument) and returns either (a) the position where the substring appears, or (b) the value `string::npos` if the substring does not appear. For example:

```
string filename = "North-Data.txt";
if ( filename.find("North", 0) != string::npos ) { /* This happens */ }
if ( filename.find("West", 0) != string::npos ) { /* This does NOT happen */ }
```

The contents/file format is unchanged from Project 2. That is, the first line of the text file contains column headers, and each following line contains a single shirt order with data. The order of this data is repeated here for reference:

- **Order Date** is always written as `yyyy/mm/dd`
- **Print method** is a single character (either upper- or lower-case).
- **Message** begins and ends with a double-quote, and may contain spaces and/or punctuation
- **Number M**, **Nu**
- **Shirt color** is a s
- **Ink color** is also
- **Customer name** name and email (which we will treat as a

One example of a single

```
2023/06/26 S "Hello                                    d.com
```

Data validation

The rules for validating before, you may assume no data type errors but rules.

These validation rules a

- **Order Date** - Al f March (or earlier) are considered expire
- **Print method** - Project 1), orders may specify 'n' (for "N ; any other character is invalid.
- **Message**
  - Messages will begin/end with a double-quote character; these quotes are **not** part of the message. The length of the message must be 48 characters or fewer (same as Project 1).
  - If printing method is 'n' then the message must be empty (`""` in the text file)
  - If printing method is not 'n', then the message must have at least one non-whitespace character.
- **Number of shirts** - Each count is between 0 and 14400 and the sum of all three must not be zero.
- **Shirt color** - Valid colors are "White", "Black", "Green", "Blue", "Yellow", or "Red". Simplifying from Project 1, valid colors will always be spelled with a capitalized first letter.
- **Ink color**
  - Valid colors are "White", "Black" or "None" and must not be the same as the shirt color.

  - If the print method is 'n' then the ink color **must** be "None"
  - If the print method is not 'n' then the ink color **must not** be "None"
- **Customer name and email** - Your program should handle this data as a single sring (with spaces) consisting of everything following the ink color up to the end of the line. In particular, the customer name may have any number of words, so you should not try to separate/parse this in any way.

Calculations

The rules for cost calcuation have not changed from Project 1; they are summarized here for your reference.

Printing types and costs

- Silk-screen has a set-up cost of $111.00 and a per-unit cost of $1.59
- Iron-on has a set-up cost of $0.00 and a per-unit cost of $2.35
- Embroidered has a set-up cost of $12.00 and a per-unit cost of $7.99
- None has a set-up

Message length costs

- Silk-screen per-un
- Iron-on per-unit c
- Embroidered per-
- *None must have a*

Shirt size costs and shir

- Medium shirts cos
- Large shirts cost $
- Extra large shirts

If the shirt color is any $0.22

Order discounts

- Orders of at least
- Orders of at least

Printed output

Your program will print a formatted table of output describing the data it reads from the file. The beginning of this output should be a "header" displaying column titles for the data to follow.

Your program should display the following data for **every** line it reads from the file:

1. The order date
2. The printing method
3. Up to five characters of the message; if the messge is longer than five characters, print the first five followed by ellipses
4. The length of the message
5. The number of medium; large; and extra large shirts

6. The shirt color
7. The ink color

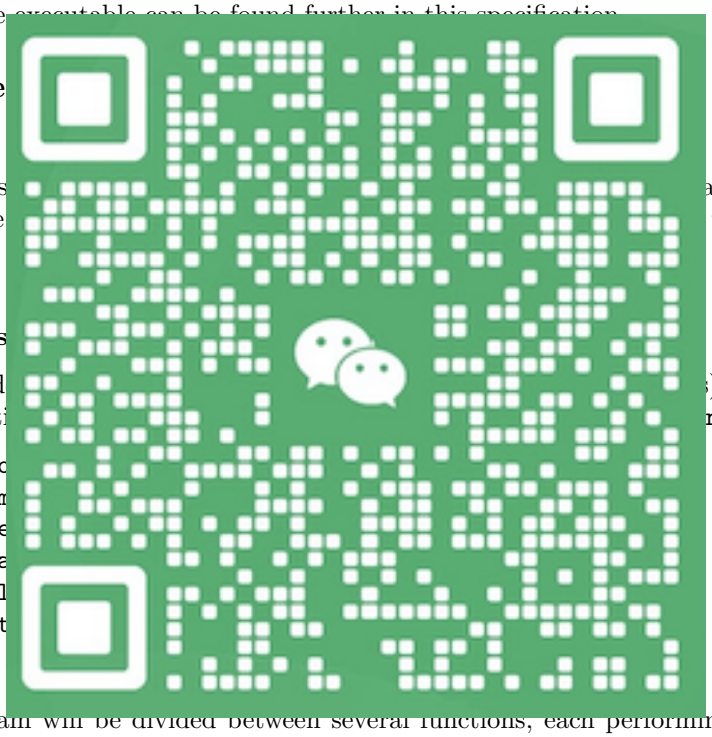
The last column of output is the final cost of the order. If the order is valid, your program should calculate the final cost and print it here.

If the order is invalid, your program should leave the final cost blank. Instead, it should print an error message describing the errors in the row. If the row is invalid in more than one way, then more than one error message should be printed (one per line). After printing the appropriate error message(s), print the contact information for the order.

Once all lines of the file have been processed, your program should print the following information:

- The number of valid lines and the number of invalid lines encountered (the sum of these two numbers would be the total number of lines in the file)
- A table of costs (totaled over all valid lines) separated by printing method.

The output of a sample executable can be found further in this specification



3.2 Program Re

Inputs

The primary user inputs at behavior is performed next. In the case of the to be opened.

Processing

Data storage vectors

To help you out, the d) you will need for the project. These declarati n().

```
vector<string> regio
vector<int> yyyy, mm
vector<char> printMe
vector<string> messa
vector<int> m, l, xl
vector<string> shirt
```

Required functions

The work of the program will be divided between several functions, each performing a specific task. You must implement each of the functions described here and use them to accomplish their intended task.

You may implement additional functions to support your program if you wish, but they must not replace the functions described here.

To help you out, prototypes for the required functions are provdied below

```
char displayMenu();
```

Prints the menu and receives the user's menu choice, which is returned as a character. It should clear any additional input (beyond the first character) so multiple choices can be made over time.

Parameters: none
Return value: A character representing the user's menu choice (possibly invalid)

```
bool uploadFile(string fileName, vector<string>& region,
                vector<int>& yyyy, vector<int>& mm, vector<int>& dd,
                vector<char>& printMethod, vector<string>& message,
                vector<int>& m, vector<int>& l, vector<int>& xl,
                vector<string>& shirtColor, vector<string>& inkColor,
                vector<string>& contact);
```

Attempts to open the file described by `fileName` and, if successful, reads and validates the shirt orders contained within that file. The data for each valid line of the file is *appended* to the parallel vector parameters for storage and later use.

The output of this function should be exactly the same as that of Project 2, in that it prints the details of each order as it is read; prints either the cost (of a valid order) or errors (for an invalid order); and prints a summary of the costs of valid orders (read from this file) at the bottom.

Parameters:

- `string fileName` - the name of the file to be opened and read
- References to the

Return value: Returns t irt order is read/stored to the data vectors; fals

```
void allDetails(cons
                cons                                vector<int>& dd,
                cons                                message,
                cons                                or<int>& xl,
                cons                                & inkColor,
                cons
```

Displays individual orde rs, one per line. Should only be called if these v

Parameters: Const refer
Return value: none

```
void summaryByMethod                                onst vector<int>& dd,
                                                    ing>& message,
                                                     vector<int>& xl,
                                                    ring>& inkColor,
```

Calculates running totals of the shirt cost; printing cost; and final order cost for all orders stored in the parallel vector parameters, grouped according to printing method. Displays these totals in a neatly-formatted output table. Should only be called if these vectors contain at least one shirt order.

Parameters: Const references to the parallel vectors (containing shirt order data) Return value: none

Outputs

The program will produce outputs in the same neatly-formatted style as in Project 2, either in whole (when (U) is selected) or in part (when (A) or (M) are selected. The program should also print suitable error messages if it encounters an unexpected situation.

4 Project Deliverables

The only deliverable for this assignment is the source code for your program.

Testing on cs-class

The *target machine* for this project is cs-class; this means we will grade your project by compiling and running the code on this machine. In order to receive credit for your work the source code must compile without error and run as expected on this machine.

You may choose to write your code directly on cs-class (e.g., using ssh and a command-line editor) or to write your code on a personal computer. *It is possible for compiled programs to run differently on cs-class than on your computer. It is also possible that the program will compile on your computer and not on cs-class!* Your are *strongly* encouraged to test your completed code by uploading it to cs-class, compiling there, and running the code with several different inputs (both valid and invalid).

Note that uploading your source code to cs-class for testing is not the same as submitting it for grading. See below for instructions on how to submit your project to be graded.

Submitting your cod

You will save your final submit "re268P3.cpp"). Upload this file as a sub

Academic honesty st

Your finished source co your own. Refer to the guidelines specified in th structor if you have any questions about what is

Include the following st

```
/*
 * <netid>P3.cpp
 *
 * COSC 1020 Fall 20
 * Project #3 Code
 *
 * Due on: 3 Novembe
 * Author: <netID>
 *
 * In accordance with class policies and Georgetown's Honor Code,
 * I certify that, with the exception of the class resources and those
 * items noted below, I have neither given nor received any assistance
 * on this project.
 *
 * Note that you may use without citation any help from our TAs,
 * professors, or any code taken from the course textbook.
 */
```

Note that omitting required comments/documentation (like the above) detracts from overall code quality and may result in a loss of points in your final score.