CS201 – Summer 2016 - Sabancı University
Homework #4
Due: August 15, 2016, Monday, 19:00 (Sharp Deadline)

Data Mining Application

Brief Description

In this homework, you will write a program that reads information from 3 text files and generates a sorted sales report as an output file. The input data files contain information about items, merchants and their monthly sales information for a year. Your program will produce a report showing the total sales of each merchant per item sorted by item name.

Input Files

Your program will read several text files:

1) **Items** database file: Each line in this file consists of two fields: first one is *itemID* and second one is *itemName* in the following format:

```
itemID itemName
```

They are separated by a space, but *itemName* may consist of one or more words separated by a space. (**Hint**: consider using `getline` and `substr` together).

2) **Merchants** database file: Each line in this file consists of two fields: first one is *merchantID* and second one is *merchantName* in the following format:

```
merchantID merchantName
```

They are separated by a space, but *merchantName* may consist of one or more words separated by a space.

3) **Sales** database file: This file contains the monthly sales information for a year (let's say 2015). It contains a line for each item as follows:

```
******itemID******
```

There can be any number of such lines for a given *itemID*. You cannot assume that there will be only one entry for an item (example: there can be two `******1******` in our test cases).

The value of this line contains the *itemID*, which you will have to extract out of this line.

After this line, there will be **<u>any</u>** number of lines containing the sales information for this item. Each line contains 3 values: *merchantID*, *saleMonth*, *saleAmount*.

An example of the sales file is given below:

```
******1******
1 12 100
2 11 10
2 1 100
4 1 120

******3******
1 5 100
3 7 10
1 2 200
2 1 150
```

For sample input files, you may check out the items.txt, merchants.txt and sales.txt files provided in this homework zip package. Note that *itemID* and *merchantID* values in the *sales.txt* file will be given on *items.txt* and *merchants.txt* respectively.

You can assume that there will be only ***1*** space between each word and/or number in text files. So the merchant or item names may consist of multiple words but each word will be separated by 1 space character. All text data in the files will be uppercase letters.

The structure of the input file, the associated rules, assumptions and the things that you cannot assume, which are all explained above, are fixed such that you cannot change them or make any other assumptions. No format check for the contents of the input file is needed. You may assume that all data are in correct format.

The names of the input and output files are all **keyboard** inputs. If the user enters a **wrong file name** that does not exist, then your program should display an appropriate error message and **ask for a new input file name** until the file is found and **successfully opened**.

Processing

You can use a **`struct`** data structure to encapsulate data of a single entry *itemName*, *merchantName*, and *totalOrder*. It is up to you to decide which fields are to be included in this struct.

You can use a **`vector`** to store the list of entries. Of course, the element type of this vector will be the previously defined data **`struct`**. Please note that you **CANNOT** make any assumptions about the number of items in given *sales.txt* file. It can contain only some or all of the items in the *items.txt* file.

The flow of your program may be as follows:

– Your program first asks the file names from the user for the items, merchants, sales input files and the output file. Then, the user enters these file names as input. After that, your program opens the files and tests for failure, asks for a new filename in case of any failure.

– Then your program processes the sales file. As mentioned above, there are several lines for item sales information in the sales file. An item information group starts with **`******`** followed by the *itemID* and then **`******`** (Example: **`******2******`**)

– After such a line with *itemID*, your program reads several lines, each of which consists of *merchantID*, *month* and *orderCount* values. You will then use the *merchants.txt* file to match the merchant *merchantID from sales file with the line in merchants.txt file to find out merchantName*. You will also need to read the *items.txt* file to find the *itemName* corresponding to the *itemID*. You need to add *orderCount* to *totalOrder* to update the total number of sales of a product by a merchant. After this step you have the following 3 values stored in the `struct` for a single item's sale data:

  itemName, MerchantName, totalOrder

After your program finishes reading and processing all of the sales results from the input files and storing the necessary data in the vector, it should sort the vector by the following criteria:

- You may modify and use one of sorting algorithms provided by the book and/or discussed in the lectures. If you want to develop your own sorting algorithm, of course you may, but this will be more difficult.

– You should sort the vector first by *itemName* in ***ascending order***; if two *itemName*'s are same, then you should sort it by *totalOrder* in ***descending order;*** if two *totalOrder* values are also the same, then you should sort it by *merchantName* in ***ascending order***.

– Finally display a message that informs the user about the end of the process and the name of the output file.

Output

After the sorting operation is finished, the vector of structs should be written to a file. The name of the output file should also be an input from the user.

The format of the table is described below and **you must follow this format precisely**.

First, you should print out the *itemName* and put ten (10) "*" in the line below. After that you should list all the sales of that item in a sorted way (as specified above). One line of data should be in this order:

  orderTotal MerchantName

Below you will find part of an example output file:

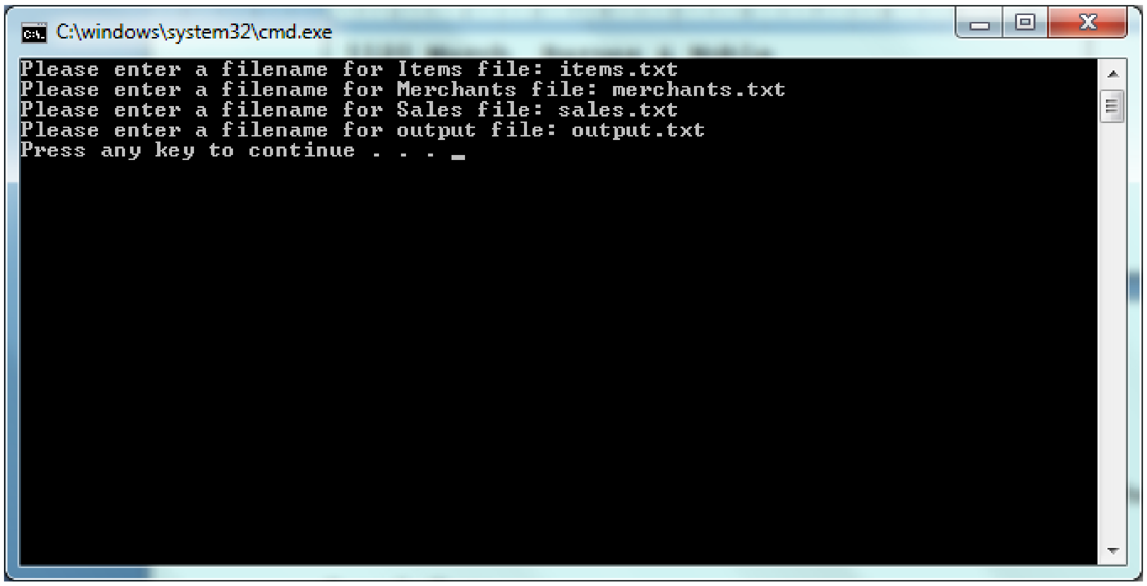
```
CANON MG2450 INKJET MULTIFUNCTION PRINTER
**********
250  TEKNOSA
120  BIMEKS
120  METRO GROSS MARKET
110  MEDIA MARKT
MACBOOK PRO RETINA 13 INCH
**********
2140  MEDIA MARKT
120  BIMEKS
120  TEKNOSA
SAMSUNG GALAXY S7
**********
2075  METRO GROSS MARKET
270  TEKNOSA
130  BIMEKS
```

For each entry, you should print these values in a sorted manner. In the zip package of this homework you will find sample input and output files.

Sample Runs

Sample Run 1

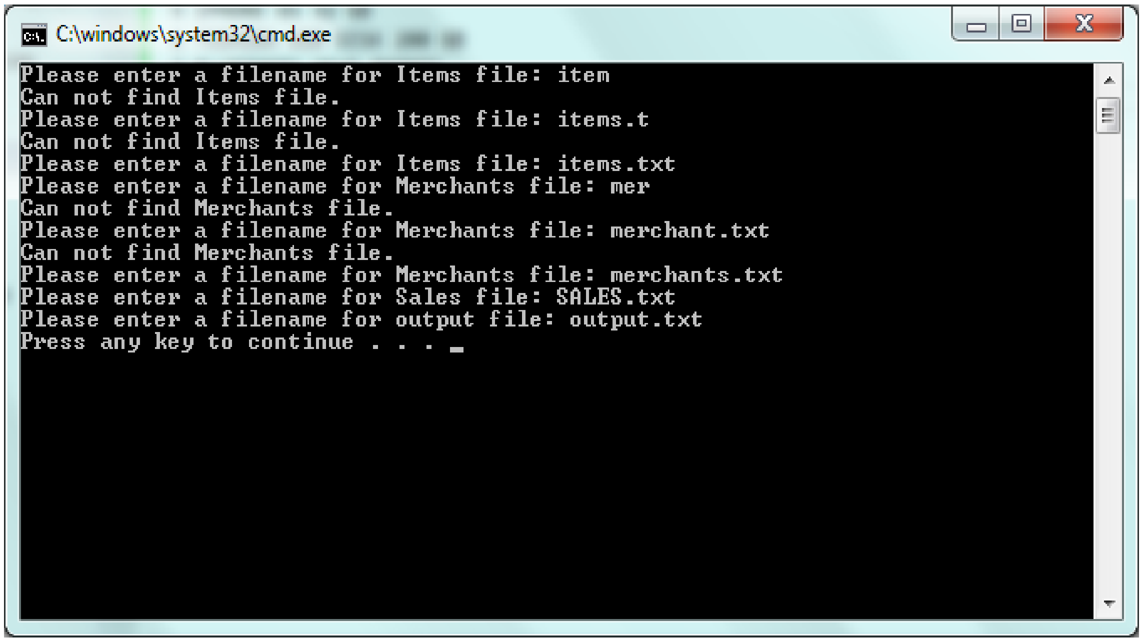
Correct Input



```
Please enter a filename for Items file: items.txt
Please enter a filename for Merchants file: merchants.txt
Please enter a filename for Sales file: sales.txt
Please enter a filename for output file: output.txt
Press any key to continue . . .
```

Sample Run 2

Incorrect input



```
Please enter a filename for Items file: item
Can not find Items file.
Please enter a filename for Items file: items.t
Can not find Items file.
Please enter a filename for Items file: items.txt
Please enter a filename for Merchants file: mer
Can not find Merchants file.
Please enter a filename for Merchants file: merchant.txt
Can not find Merchants file.
Please enter a filename for Merchants file: merchants.txt
Please enter a filename for Sales file: SALES.txt
Please enter a filename for output file: output.txt
Press any key to continue . . .
```

Important Remarks

Your program must be modular and you should avoid code duplication. Thus you have to show your ability to use functions in an appropriate way. This will affect your grade. In general, if your main function or any user-defined function is too long and if you do everything in main or in another user-defined function, you grade may be lowered.

Try to use parametric and non-void functions wherever appropriate. Do NOT use any global variables (variables defined outside the functions) to avoid parameter use.

In this homework (and in the coming ones) you are not allowed to use instructions such as “exit” and “goto”. These cause difficulty to control the flow of your programs. Thus we do not approve using them. You are also not encouraged to use “break” and “continue”. The use of “break” and “continue” prevent you from forming good readable loop conditions and hence prevent you from learning how to form good loops. Think cleverly in order not to use any of these instructions. If you don't know these commands, do not even try to learn them (we explained “break” in class).

Please remark that the only inputs are input file names, business type to be searched and user's location. No other input is allowed (such as a name for the introduction or anything else not mentioned in this homework specification). Since your submissions are processed automatically, extra inputs cause problems and delays in the processing and grading. If you do not follow this rule, your grade may be lowered.

Two common problems and their solutions

Question: I enter a correct input file name, but my program does not accept it.
Answer: Your input file should be in the same folder as the .dsp file of your project. Otherwise they may not be found while they are opened.

Question: When I compile, I receive several “template function has already been defined” errors. What should I do?
Answer: As mentioned in lectures and recitations, in order to use the tvector class, you have to include `vector.h` at the beginning of your program using **`#include <vector>`**.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS201. Office hours of TAs are at the class website. All office hours will be held in FENS L068 (assistant office hour room). Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading and Objections

<u>Careful about the semi-automatic grading:</u> Your programs will be graded using a semi-automated system. Therefore you should follow the guidelines about input and output order; moreover you should also use same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

<u>Grading:</u>

- ❑ Late penalty is 10% off of the full grade and only one late day is allowed.
- ❑ **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications (which is also bad) will also affect your grade.**
- ❑ Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- ❑ For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/su_current_courses/cs-201-spring-2008/plagiarism/ and keep in mind that

Plagiarism will not be tolerated!

<u>Grade announcements:</u> Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

<u>Grade objections:</u> It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your homework from the email address provided in the comment section of your announced homework grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the homework tab to see the problem with your homework.
- Download the .zip file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

What and where to submit (IMPORTANT)

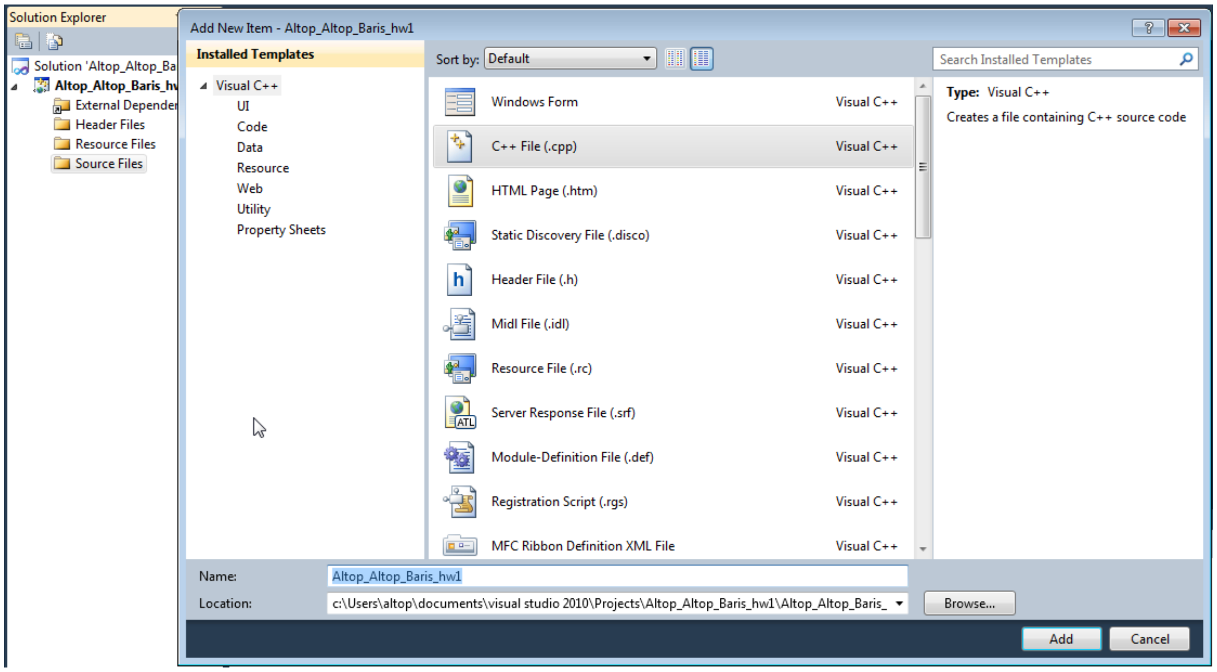
Submissions guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

<u>Add your name to the program:</u> It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).
<u>Name your submission file:</u>

❑ Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.

❑ Name your cpp file that contains your program as follows.

“**SUCourseUserName_YourLastname_YourName_HWnumber.cpp**”


Solution Explorer
Solution 'Altop_Altop_Ba
Altop_Altop_Baris_hw
External Depender
Header Files
Resource Files
Source Files
Add New Item - Altop_Altop_Baris_hw1
Installed Templates
Visual C++
UI
Code
Data
Resource
Web
Utility
Property Sheets
Sort by: Default
Search Installed Templates
Windows Form Visual C++
C++ File (.cpp) Visual C++
HTML Page (.htm) Visual C++
Static Discovery File (.disco) Visual C++
Header File (.h) Visual C++
Midl File (.idl) Visual C++
Resource File (.rc) Visual C++
Server Response File (.srf) Visual C++
Module-Definition File (.def) Visual C++
Registration Script (.rgs) Visual C++
MFC Ribbon Definition XML File Visual C++
Type: Visual C++
Creates a file containing C++ source code
Browse...
Add
Cancel

Name: Altop_Altop_Baris_hw1

Location: c:\Users\altop\documents\visual studio 2010\Projects\Altop_Altop_Baris_hw1\Altop_Altop_Baris_

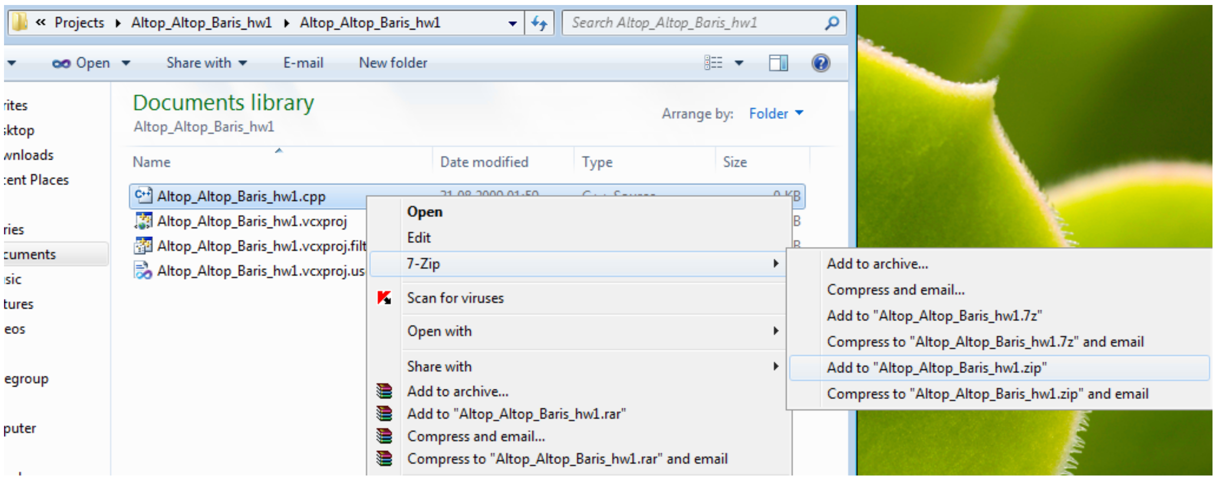
❑ Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

Cago_Ozbugsizkodyazaroglu_Caglayan_hw2.cpp

❑ Do not add any other character or phrase to the file name.

❑ Make sure that this file is the latest version of your homework program.

❑ Compress this cpp file using WINZIP or WINRAR programs. Please use "**zip**" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension.



Projects
Altop_Altop_Baris_hw1
Altop_Altop_Baris_hw1
Search Altop_Altop_Baris_hw1
Open
Share with
E-mail
New folder
Documents library
Altop_Altop_Baris_hw1
Arrange by: Folder
Name
Date modified
Type
Size
Altop_Altop_Baris_hw1.cpp
Altop_Altop_Baris_hw1.vcxproj
Altop_Altop_Baris_hw1.vcxproj.filt
Altop_Altop_Baris_hw1.vcxproj.us
Open
Edit
7-Zip
Scan for viruses
Open with
Share with
Add to archive...
Add to "Altop_Altop_Baris_hw1.rar"
Compress and email...
Compress to "Altop_Altop_Baris_hw1.rar" and email
Add to archive...
Compress and email...
Add to "Altop_Altop_Baris_hw1.7z"
Compress to "Altop_Altop_Baris_hw1.7z" and email
Add to "Altop_Altop_Baris_hw1.zip"
Compress to "Altop_Altop_Baris_hw1.zip" and email

- ❑ Check that your compressed file opens up correctly and it contains your **cpp** file. You will receive no credits if your compressed zip file does not expand or it does not contain the correct file.
- ❑ The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows.

  “**SUCourseUserName_YourLastname_YourName_HWnumber.zip**”

  For example zubzipler_Zipleroglu_Zubeyir_hw2.zip is a valid name, but hw2_hoz_HasanOz.zip, HasanOzHoz.zip are NOT valid names.

<u>Submission:</u>

- ❑ Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).
  1) Click on "Assignments" at CS201 SUCourse (not the CS201 web site).
  2) Click Homework 2 in the assignments list.
  3) Click on "Add Attachments" button.
  4) Click on "Browse" button and select the zip file that you generated.
  5) Now, you have to see your zip file in the "Items to attach" list.
  6) Click on "Continue" button.
  7) Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

<u>Resubmission:</u>

- ❑ After submission, you will be able to take your homework back and resubmit. In order to resubmit, follow the following steps.
  1) Click on "Assignments" at CS201 SUCourse.
  2) Click Homework 2 in the assignments list.
  3) Click on "Re-submit" button.
  4) Click on "Add/remove Attachments" button
  5) Remove the existing zip file by clicking on "remove" link. This step is very important. If you do not delete the old zip file, we receive both files and the old one may be graded.
  6) Click on "Browse" button and select the new zip file that you want to resubmit.

7) Now, you have to see your new zip file in the "Items to attach" list.
8) Click on "Continue" button.
9) Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Mehmet Çağrı Çalpur and Gülşen Demiröz