Lab 1 – Introduction to C++ and Microsoft Visual Studio

Computer Science 2413 – Data Structures – Fall 2016

Due by 11:00 pm CST on Thursday, 1 September 2016

This lab is individual work. Each student must complete this assignment independently.

User Request:

"Create a simple program to read and write merchant vessel data."

Objectives:

1	Create and correctly submit a Microsoft Visual Studio project.	*5 points*
2	Use the redirected standard C++ input stream (`cin`) to read a file.	*5 points*
3	Use C++ to do minor processing of input, sending errors to the C++ error stream (`cerr`).	*5 points*
4	Use the redirected standard C++ output stream (`cout`) to send output to a file.	*5 points*
►	Use proper coding style, documentation, and formatting.	*5 points*

Description:

Merchant vessels are large ships used to move cargo or passengers by sea. For this lab, you will put together several techniques and concepts you have learned in CS 2334 (or from a similar background) and some new techniques to make *Merchant Vessel Program* or *MVP*, a program that reads, processes, and outputs data on merchant vessels. MVP version 0.1 (which satisfies the requirements of this lab) will read through a data file on merchant vessels[1], perform simple processing on the data, and send the resulting output to another file.

Operational Issues:

Your program will read the merchant vessel data file—a comma-separated value (csv) text file—via redirected standard input. The data file will be organized as follows:

The first two rows (lines) of the file will contain comma-separated lists of the headers for the columns of data in the rows that follow. On the first row there will be eight headers separated by 24 commas. On the second row there will be 25 headers separated by 24 commas. Each header will be one or more alphabetic characters and/or symbols but will not contain commas. Note that the first row contains more commas than headers as each header is meant to label three data columns.[2]

Each row after the first two is a data row. Each data row will contain data on the merchant vessels of a single country[3] except for the final row which contains worldwide totals. MVP will not know in advance the number of data rows to read. However, for this lab, you only need to read, process, and output one line at a time, so it should not be necessary for MVP to know the total number of rows in advance. The data in these rows will also be comma separated, with 25 data fields (24 commas) per row.

1 The data in this file was derived with minor revisions from data found on the Maritime Data & Statistics webpage of the United States Maritime Administration (https://www.marad.dot.gov/resources/data-statistics/).

2 Compare the csv file to the associated Portable Document Format (pdf) file to see the coverage for each header.

3 Actually, it's a bit more complicated than that. Most rows are for individual countries but there are a few other entries as well, such as the row for ships of "Unknown Registry." Also, some countries have data in more than one row. However, for the purposes of this lab, we can and will treat all primary data rows as if they have data for a single country.

The first field in each data row will be the name of the country. The remaining 24 fields will be numeric values. The first three of the 24 numeric fields contain country totals for the number of merchant vessels registered in that country, the gross tonnage of those ships, and the deadweight tonnage of those ships, in that order, while the remaining 21 contain values for ships by type (for example, "Container" or "Dry Bulk"). For each type there are three data fields—number of ships of that type, gross tonnage of those ships, and deadweight tonnage of those ships, in that order.[4] Note that some of the numeric fields may be empty. An empty field must be interpreted as a data value of zero.

Consider the following data row:

```
Bahrain,12,211079,256916,3,158329,179952,,,,9,52750,76964,,,,,,,,,,,,
```

This data row indicates that the country of Bahrain has 12 total merchant vessels registered to it with a total gross tonnage of 211,079, a total deadweight tonnage of 256,916, and so on.

As MVP reads the data file, there are a couple of correctness checks that it must make on the data.

First, MVP must ensure that there are 25 fields for each data row. As mentioned, most of these fields may be blank (fields 2 – 25) but the first is not allowed to be blank[5], and all fields should be present (that is, there should be 24 commas in each row).

Second, MVP must ensure that for each country, there is a match between the country totals listed in the first three numeric fields and the sum of the values for the individual ship types within that country. For example, the data row shown above indicates that Bahrain has a total of 12 registered merchant vessels, as mentioned previously. This value of 12 from data field two should equal the sum of the values from data fields 5, 8, 11, 14, 17, 20, and 23, which are the data fields that hold the numbers of ships of each type. In this case, they match (12=3+0+9+0+0+0+0).

If MVP encounters either of these errors while reading the data file, it must provide an error message indicating which error it encountered and at which line number, and move on to reading, processing, and outputting the next line in data file.

As the data is read in, line by line, and processed to check for errors, MVP will send it back out again, albeit in a slightly altered form. Rather than comma separated, the output will be lined up in columns using spaces. In particular, each data field will be written into a field of a given width based on the number of characters it contains, filling in the remaining characters with spaces. The first field will have a width of 30 characters and be left justified (that is, the name will appear at the left edge of the column and the extra spaces will be at the right) whereas each of the remaining 24 fields will each have a width of 20 characters and be right justified (that is, the numeral will appear at the left edge of the column and the extra spaces will be at the left).

For example, the output line for Bahrain will start[6] as follows:

```
Bahrain                                        12              211079
```

Note that there are 41 spaces between "Bahrain" and "12" because "Bahrain" is seven letters long, meaning it needs to be padded with 23 spaces to its right (it's column width is 30 and 30–7=23), and because "12" is two digits long, meaning that it needs to be padded with 18 spaces to its left (it's column width is 20 and 20–2=18). So, a padding of 23 plus a padding of 18 is a total padding of 41.[7]

---

4 Again, refer to the associated pdf file to more easily see how data columns to relate to particular headers.

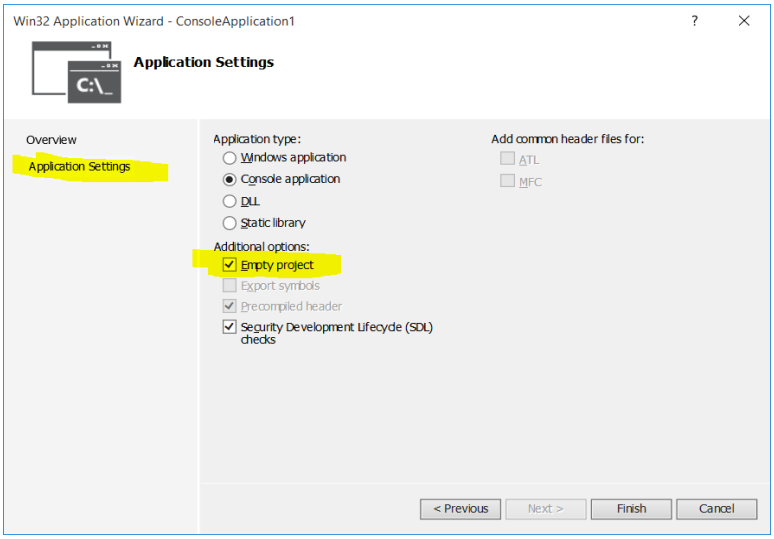
5 Note that the first header row will start with a blank field.

6 Note that only the start of the output line is shown, since all of the padding makes the output line too wide to display on a single line of this document.

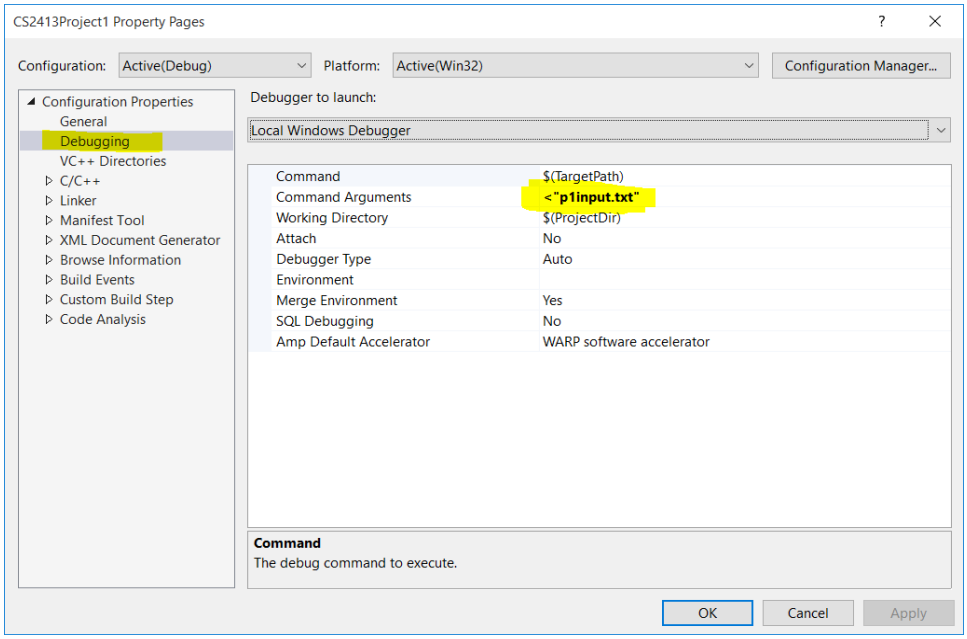
7 Note that you may not need to write your own code to carry out these pesky calculations if you make judicious use of existing C++ libraries. See "Implementation Issues," below.

Implementation Issues:

You must create and submit a Microsoft Visual Studio project to complete this lab (as well as all other labs and projects in this course). Submissions in any other format will not be graded and will receive a zero. Make sure to select "Console application" and "Empty project" when creating this project.



You will use the standard C++ input, error, and output streams (`cin`, `cerr`, and `cout`, respectively) for all input and output for this assignment. You will need to tell Visual Studio that you wish to redirect `cin` to read from your data file and redirect `cout` to send output to your output file. This can be done from the debugging configurations within your Visual Studio project. Note that you do not need to redirect `cerr` for this assignment. Note also that the example shown here is redirecting `cin` to come from a file named "`p1input.txt`" rather than "`DS_16_WorldRegistries.csv`" (which is the name of your data file). Also, this example only shows redirecting `cin` (using "<") but not `cout` (which uses ">"). Your output file must be named "`MVPoutput.txt`" and you will need to adjust your settings accordingly.



The only libraries you may use for this assignment are `iostream`, `iomanip`, and `string` (`#include <iostream>`, `#include <iomanip>`, `#include <string>`).

Note that this lab does not require you to create any classes or to use object-oriented design at all. In fact, you don't even need to create any functions except the `main` function. Nonetheless, you must use good programming style and documentation, including making your code modular, using explanatory comments for each section of code, using meaningful variable and method names, using consistent indentation, etc.

You must follow C++ conventions for compilation modules, including using a header (`.h`) file for constant variables, prototypes, etc. and a source (`.cpp`) file for implementations of functions, methods, etc.

You may write your program from scratch or may start from programs for which the source code is freely available on the web or through other sources (such as friends or student organizations). If you do not start from scratch, you must give a complete and accurate accounting of where all of your code came from and indicate which parts are original or changed, and which you got from which other source. Failure to give credit where credit is due is academic fraud and will be dealt with accordingly.

Due Date:

You must submit an electronic copy of your Visual Studio project to the appropriate dropbox in D2L by **11:00 pm CST on Thursday, 1 September 2016**.