Lab 1 revision 1 – Introduction to C++

Computer Science 2413 – Data Structures – Fall 2018

Due by 11:59 pm CST on Friday, 31 August 2018

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

User Request:

"Create a simple program to read and write NVRA data."

Objectives:

1	Create and correctly submit a C++ source code file (named "`VoteR.cpp`").	*5 points*
2	Use the standard C++ input and output streams (`cin` and `cout`) to read and write data.	*5 points*
3	Use C++ to do minor processing of input.	*5 points*
4	Use proper coding style, documentation, and formatting.	*5 points*

Description:

According to the Oklahoma State Election Board, "[t]he National Voter Registration Act (NVRA) was passed by the United States Congress in 1993. The NVRA requires states to make voter registration opportunities for federal elections available through the mail and when people apply for or receive driver licenses. The NVRA also requires government agencies providing public assistance, disability services, military recruitment and other government services to provide voter registration services. Election officials are required by the NVRA to conduct a voter list maintenance program to remove the names of ineligible voters from the official lists of eligible voters."[1] Relevant to this statute, the board publishes monthly NVRA statistical reports. 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 *VoteR*, a program that reads, processes, and outputs data from these reports. VoteR version 0.1 (which satisfies the requirements of this lab) will read through a file of NVRA data (using redirected `cin`), perform simple processing on the data, and send the resulting output to `cout`.

Operational Issues:

Your program will read the NVRA data file—a text file in which data fields are separated by commas. The data file will be organized as follows:

The first row consists of column headers and should be ignored. The remaining lines of the file contain the data. Each data row contains data on a single NVRA record. VoteR will not know in advance the number of data rows present in the file. However, for this lab, you only need to read, process, and output one line at a time, so it should not be necessary for VoteR to know the total number of rows in advance. The data in these rows will be in 24 columns (that is, separated by 23 commas) per row. Most of the columns contain data that should be internally represented as integer data but three columns contain data that should be internally represented as strings. These are the fourth, twelfth, and thirteenth columns.

As VoteR reads the data file, there is two correctness checks that it must make on the data. First, VoteR must ensure that the record ID (found in the first column) of each vote is unique (not repeated). If VoteR encounters a duplicate record ID while reading the data file, it must provide an error message indicating

[1] https://www.ok.gov/elections/About_Us/National_Voter_Registration_Act/

that it encountered a duplicate and the line number at which the duplicate was found, then move on to reading, processing, and outputting the next line in data file. The exact format of the error must be as follows:

`Duplicate record ID r at line n.`

Naturally, *r* should be replaced by the duplicated record number and *n* should be replaced by the line at which the duplicate was found, counting the first data line (not the header line) as line 1. Lines at which duplicates are encountered *do* increment the line count.

Second, if the record ID is not a duplicate, VoteR should verify that all integer values in the data are greater than or equal to zero (that is, that none of them are negative). If any of the integer values are negative, VoteR must provide an error message indicating that it encountered an invalid data value and the line number at which the invalid data was found, then move on to reading, processing, and outputting the next line in data file. The exact format of the error must be as follows:

`Invalid data at line n.`

Naturally, *n* should be replaced by the line at which the invalid data was found, counting the first data line (not the header line) as line 1. Lines at which invalid data is encountered *do* increment the line count, but do not have their record ID added to the list of unique IDs already read in.

As the data is read in, line by line, and processed to check for duplicates and invalid values, VoteR will send it back out again, albeit in a slightly altered form. Rather than comma separated, the output data will be separated by semicolons. Lines containing duplicate record IDs or invalid data values will not be output, only errors messages will be output for those lines.

Implementation Issues:

You must create and submit C++ source code to complete this lab (as well as all other labs and projects in this course).

You will use the standard C++ input and output streams (`cin` and `cout`, respectively) for all input and output for this assignment. To test your code on the data file, you will want to tell VoteR that you wish to redirect `cin` to read from your data file. This can be done from the configurations within your VoteR project. Note that you do not need to consider the standard C++ error stream (`cerr`) for this assignment.

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 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 Vote project to the appropriate dropbox in Canvas by **11:59 pm CST on Friday, 31 August 2018**.