Lab 1 – Introduction to C++ in Eclipse

Computer Science 2413 – Data Structures – Fall 2017

Due by 11:59 pm CST on Thursday, 31 August 2017

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

User Request:

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

Objectives:

1	Create and correctly submit an Eclipse C++ 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*
5	Use proper coding style, documentation, and formatting.	*5 points*

Description:

Eclipse, as it turns out, isn't just an integrated development environment for Java, it also works for many other languages, including C++. Also it's the name for the phenomenon when one celestial body obscures the light from another from the perspective of the observer. 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 *EclipseR*, a program that reads, processes, and outputs data on celestial eclipses. EclipseR version 0.1 (which satisfies the requirements of this lab) will read through a data file on eclipses[1], perform simple processing on the data, and send the resulting output to another file.

Operational Issues:

Your program will read the eclipse data file—a text file in which data fields are (generally) separated by spaces—via redirected standard input. The data file will be organized as follows:

The first ten rows are headers and should be ignored. The remaining lines of the file contain the data. Each data row contains data on a single eclipse. EclipseR 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 EclipseR to know the total number of rows in advance. The data in these rows will be in columns (that is, separated by spaces[2]) with 16 or 18 columns per row.

The reason that there may be either 16 or 18 columns of data per eclipse is that columns 17 and 18 are not appropriate for all eclipses. In particular, if the eclipse type (specified in column 10) starts with "P" (for partial), then columns 17 and 18 (path width and central duration) are not appropriate and are omitted.

1 The data in this file is from NASA (https://eclipse.gsfc.nasa.gov/5MCSE/5MCSEcatalog.txt). However, at the time this assignment was created, NASA was redirecting requests sent to that URL to a popular pablum website put in place for the 21 August 2017 eclipse instead (https://eclipse2017.nasa.gov/). So, this data was instead downloaded from the Inernet Archive (https://web.archive.org/web/20161226224457/https://eclipse.gsfc.nasa.gov/5MCSE/5MCSEcatalog.txt).

2 Note that there may be one or more spaces between the columns as they are intended to line up nicely when viewed with a fixed-width font and some data items contain more characters than others.

For a description of the other data columns, see the Key to Catalog of Solar Eclipses published by NASA[3]. Note that NASA's catalog key oddly omits column 2 which is entitled "Canon Plate," and counts what I am considering columns 3, 4, and 5 as a single column ("Calendar Date").

As EclipseR reads the data file, there are a couple of correctness checks that it must make on the data. First, EclipseR must ensure that there are the correct number of columns in each data row, based on eclipse type. Second, it must ensure that the catalog number of each eclipse is unique (not repeated). If EclipseR 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, EclipseR will send it back out again, albeit in a slightly altered form. Rather than lined up in columns using spaces, the data output will be comma separated.

Implementation Issues:

You must create and submit an Eclipse project file 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.

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 Eclipse 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 configurations within your Eclipse project. Note that you do not need to redirect `cerr` for this assignment. The input file must be named "`5MCSE_5MCSEcatalog.txt`" and your output file must be named "`EclipseRoutput.csv`" 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 Eclipse project to the appropriate dropbox in Canvas by **11:59 pm CST on Thursday, 31 August 2016**.

3 This should be available at https://eclipse.gsfc.nasa.gov/SEcat5/SEcatkey.html but currently is not. A copy from the Internet Archive (https://web.archive.org/web/20170118002039/https://eclipse.gsfc.nasa.gov/SEcat5/SEcatkey.html) is available (http://www.cs.ou.edu/~hougen/classes/Fall-2017/DataStructures/materials/Projects/Data/SEcatkey.html).