Due before:	11:59 PM on Wed Sept 8 2021

CSE-443/543: High Performance Computing

Homework #1

Due: Wed Sept 8 2021 before 11:59 PM

Email-based help Cutoff: 5:00 PM on Tue Sept 7 2021

Maximum Points: 26

Not including the time to read this document, you should be able to complete the actual programming in **less than 4 hours**. If it takes you more than 4-hours to complete this project, then you are missing important skills from prerequisite courses and you will need to invest some serious time into practicing the prerequisite skills.

Submission Instructions

This homework assignment must be turned-in electronically via Canvas CODE plug-in. Ensure your program compiles successfully, without any warnings or style errors. Ensure you have documented the methods. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload just the 1 source file onto Canvas via the CODE plug-in.

1. The 1 C++ source file developed for this part of the homework.

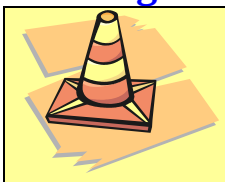
General Note: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Objective

The objective of this homework is to develop a program to process columnar data to refresh your memory on the following concepts and skills from "CSE-278: Systems 1" prerequisite course:

- Basic of I/O operations in C++
- Developing C++ program involving simple string manipulation.
- Working with command-line arguments.
- Practice developing properly styled programs

Grading Rubric:



The program submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet base case requirements or just skeleton code will be assigned zero score! Program that do not compile, **have even 1 method longer than 25 lines**, or just some skeleton code will be assigned zero score.

- **-1 Points**: for each warning generated by the compiler.
- **NOTE:** Violating CSE programming style guidelines is a compiler error! Your program should not have any style violations reported in `NetBeans` when you compile it.

- **Do not use global variables –** that is not good programming practice.
- **Base case: 6 points**: Print a single column, given index position.
- **Additional functionality #1: 5 points**: Print multiple columns given index positions.
- **Additional functionality #2: 5 points**: Filter outputs to values in first column containing given substring.
- **Additional functionality #3: 7 points**: Work with column titles instead of column index positions.
- **Formatting & Documentation: 3 points** (if your comments are poor, then you lose points in this category)
- **Delayed submission: Only 80% points:** Submission delayed **by no more than 24-hours** will be accepted for a partial credit of maximum 80% of the points.

Project overview

In this course, we will often work with columnar data files. Accordingly, this project motivates you to gain a good understanding of processing simple columnar data files in C++. Specifically, you will be working with Tab Separated Values (TSV) file and printing specified columns, optionally with the first column containing a given substring.

TSV file assumptions: The Tab Separated Values (TSV) file will contain 1 or more columns of data. Each column is separated by a tab (`'\t'`) character. Empty lines in the TSV file must be ignored. The first non-empty line is assumed to contain the titles for each column. Each non-empty line will contain the exact same number of columns. See example data in `airports.tsv` and `movies.tsv`.

Command-line arguments: The inputs to the program will be supplied via the following command-line arguments:

- The first command-line argument will always be a valid path to the TSV file to be processed.
- The next command-line argument is an optional `--filter` flag. If specified, a substring to filter based on the values in the first user-specified column is specified as the next command-line argument. E.g., "`--filter Drama`"
- The next command-line argument can one (but not both) of the following:
  - **`--colnums`** *num1 num2* …: The `--colnums` command-line argument indicates that the user is specifying the column numbers to be printed. Subsequent command-line arguments are the zero-based column index positions to be printed. For example, `--colnums 3 0 1` (print columns at index positions 3, 0, and 1)
  - **`--cols`** *name1 name2* …: The --cols argument indicates that the user is specifying column names (instead of zero-based index positions) to be printed. Subsequent command-line arguments are the column names (as they appear in the TSV). For example, `--cols Genre Year` (prints columns with title `Genre` and `Year`)

Output format: The output contains the columns specified by the user. Each column is separated by a tab (`'\t'`) character. The first line of output is the column titles as read from the TSV. All output lines are terminated by a newline character (`'\n'`).

Sample inputs and outputs

You can test and debug your program from VS-Code, by setting command-line argument as demonstrated in the `OSC OnDemand & VS-Code demonstrations` page on Canvas. See video titled `Using command-line arguments in VS-Code`.

Base case #1 (single column number) – must work to earn any points:

```
./tsv movies.tsv --colnums 2
Year
2006
2006
2012
2006
2006
2006
2005
2015
2006
```

Base case #2 (single column number) – must work to earn any points:

```
./tsv movies.tsv --colnums 4
ImdbID
450345
468094
2388725
405676
462519
434139
475169
5342766
475944
```

Additional feature #1 (multiple column numbers) – must work for graduate students only to earn any points

```
$ ./main movies.tsv --colnums 4 2 1
ImdbID  Year    Title
450345  2006    Wicker Man, The
468094  2006    Road to Guantanamo, The
2388725 2012    Paperman
405676  2006    All the King's Men
462519  2006    School for Scoundrels
434139  2006    Last Kiss, The
475169  2005    13 Tzameti
5342766 2015    Jon Stewart Has Left the Building
475944  2006    Covenant, The
```

Additional feature #2 (multiple column numbers & filter)

```
$ ./main movies.tsv --filter Drama --colnums 3 1
Genres  Title
Drama|War       Road to Guantanamo, The
Drama   All the King's Men
Comedy|Drama    Last Kiss, The
```

Additional feature #3 (multiple column names & filter)

```
$ ./main movies.tsv --filter Drama --cols Genres Title Year
Genres      Title Year
Drama|War   Road to Guantanamo, The 2006
Drama All the King's Men      2006
Comedy|Drama    Last Kiss, The 2006
```

Data files and Tips

1. Start with the basic template from `/fs/ess/PMIU0184/cse443/templates/basic` on `ptizer.osc.edu`.
2. You may copy the data files for this project from the shared project directory `/fs/ess/PMIU0184/cse443/homeworks/homework1`
3. It would be useful for you to write a simple `split` method that breaks a line into columns. Use the standard `std::quoted` modifier as shown below, to read values. **Note**: always use `std::quoted` it will handle cases with and without quotes correctly.

```
// Here, ifs could be std::cin, std::ifstream, or std::istringstream
for (std::string value; ifs >> std::quoted(value);) {
   // More code goes here…
}
```

4. Handling command-line arguments is pretty straightforward. So, think through it and processing it will be easy.
5. Prefer to work with column index positions. When working with column names, write a method that will convert column names to column index positions.
6. Recollect that filtering is based on a substring occurring in the value in the first column specified by the user.

Turn-in:

This homework assignment must be turned-in electronically via Canvas CODE plug-in. Ensure you have tested operations of your program with different test files and command-line arguments. Once you have tested your implementation, upload just your source code to Canvas.