CS214 Project 0: Simple CSV Sorter

Abstract
This is the first in a series of projects that will involve sorting a large amount of data. In this first phase, you will write a simple C program to sort a list of records of movies from imdb alphabetically by the data in a given column. All projects will be done in groups. Group signups will go out later this week. One submission per group.

Introduction
The input file will be a CSV or comma-separated value file, which is a simple way to represent a database table. In a CSV file each line is one record (or row). Commas are separators that denote different types of values per record (columns). Very often the first record (row) of a CSV is a list of value types (column headings):

```
food,calories,fat
soup,200,12
pie,500,25
celery,-10,0
```

Your code will be given a CSV as well as a value type (column heading) to sort on. Your code will then need to read in the CSV, sort it on the given value type and output the sorted version of the CSV.

Sorting the CSV above on the 'food' value type:

```
food,calories,fat
celery,-10,0
pie,500,25
soup,200,12
```

Sorting the CSV above on the 'calories' value type:

```
food,calories,fat
celery,-10,0
soup,200,12
pie,500,25
```

Methodology

a. Parameters

Your code will read the records through standard input (STDIN). Remember, the first record (line) is the column headings and should not be sorted as data. Your code must take in a command-line parameter to determine which value type (column) to sort on. If that parameter is not present be sure to print out an informative error message on standard error (STDERR) and return without creating an output file. The first argument to your program will be '-c' to indicate sorting by column and the second will be the column name:

./simpleCSVsorter -c food

Be sure to check the arguments are there and that they correspond to a listed value type (column heading) in the CSV.

b. Operation

Your code will be reading the CSV to be sorted from STDIN. In order to run your code to test it, you will need to open the CSV and read it in to the STDIN for your code:

cat input.file | ./simpleCSVsorter -c movie_title

The line above, if entered on the terminal, will open the file “input.file” and read it in to some executing code named “simpleCSVsorter”, which was invoked with the parameters “-c” and “movie_title”.

Your code's output will be a new CSV file outputted to STDOUT. You should output each record line by line using printf.

For testing purposes you can redirect STDOUT to a file:

cat input.file | ./simpleCSVsorter -c movie_title > sortedmovies.csv

c. Structure

Your code should use Mergesort to do the actual sorting of records. It is a powerful algorithm with an excellent average case. You should write your own Mergesort code.

It is strongly suggested that you use structs internally to represent each record in the CSV coming in as input.

In order to help you do both of these, please use a user-defined header file named sorter.h. We have attached one to the assignment description with some simple comments in it.

If you write any other definitions, typedefs, structs, unions or large helper functions (>~25 lines), be sure to put them in your header file and document them.

Results
Submit your “simpleCSVsorter.c”, “simpleCSVsorter.h” and “mergesort.c” as well as any other source files your header file references.

Document your design, assumptions, difficulties you had and testing procedure. Include any test CSV files you used in your documentation. Be sure to also include a short description of how to use your code. Look at the man pages for suggestions on format and content. Do not neglect your header file(s). Be sure to describe the contents of it and why you needed them. Submit this as “Asst0Doc.txt” or “Asst0Doc.pdf”. Do not submit any proprietary or Microsoft documents.

Extra
Here are some extra credit options for you. You can choose to do either, both, or neither.

Extra Credit (10 points):
Find something interesting about the data set. Analyze/mine the data for interesting statistics. Include a discussion of the algorithm and your findings.

Extra Credit (10 points):
Find a way to generalize your sorter given *any* CSV file. Include a discussion of the algorithm and any additional metadata you might require.

Additional
Q> When sorting by the item which is type of string, the sort rule is A-Z or lexicographic order?
A> The default should be lexicographical order (i.e. how strcmp evaluates strings)

Q> For the column of color, there is a blank before “Black and White”, should it be considered during sorting?
A> For all fields, one should trim leading and trailing whitespace

Q> What about the items with null value? should them be considered during sorting?
A> It is sufficient to move all null items to the beginning of the sorted list, in the order they appear in the original list

Q> Will the grader use a different CSV?
A> The grader will use a different CSV, but it will adhere to the format given in the assignment

Q> Should the sorting be ascending or descending?
A> Ascending (you can always parse a command line parameter to do the opposite)

Q> What are the types of the fields?
A> See listing below.

color
string

director_name
string

num_critic_for_reviews
numeric

duration
dateTime

director_facebook_likes
numeric

actor_3_facebook_likes
numeric

actor_2_name
string

actor_1_facebook_likes
numeric

gross
numeric

genres
string

actor_1_name
string

movie_title
string

num_voted_users
numeric

cast_total_facebook_likes
numeric

actor_3_name
string

facenumber_in_poster
numeric

plot_keywords
string

movie_imdb_link
string

num_user_for_reviews
numeric

language
string

country
string

content_rating
string

budget
numeric

title_year
numeric

actor_2_facebook_likes
numeric

imdb_score
numeric

aspect_ratio
numeric

movie_facebook_likes
numeric