Assignment 1

Due: Monday, February 5, 2024 (11:59 pm)

Submission via Git only

Contents

Programming environment

For this assignment you must ensure your code executes correctly on the reference platform you configured as part of Assignment #0 and Lab 01. This same environment will also be used by the teaching team when evaluating your submitted work.

All test files and sample code for this assignment are available on Brightspace. Git (and the assignment submission method) will be discussed in detail during the week of January 29th.

An important learning outcome of SENG 265 is to learn the tools on the Unix/Linux platform including shell scripting languages (e.g., Bash), keyboard editors (e.g., vim), dependency analyzers (e.g., make & makefiles), compilers (e.g., gcc), interpreters (e.g., Python 3), file system exploration and manipulation tools (e.g., grep, locate, ls, pwd, cd, tree, cp, rm). Therefore, you will have to make sure that your programs compile, link, and execute perfectly on the reference platform.

Individual work

This assignment is to be completed by each individual student (i.e., no group work). You are encouraged to discuss aspects of the problem with your fellow students. However, **sharing of code fragments is strictly forbidden**. Note **SENG 265 uses highly effective plagiarism detection tools to discover copied code in your submitted work**. Both, using code from others and providing code to others, are considered cheating.

Objectives of this assignment

- Understand a problem description, along with the role of the provided sample input and output.
- Use the programming language C to implement a Unix filter—a song data processor named *song_analyzer.c*—without resorting to dynamic memory allocation.
- Use *git* to manage changes in your source code and annotate the continuous evolution of your solution with "messages" given to commits.
- Test your code against the provided test cases.

This assignment: *song_analyzer.c*

a) In A1, song_analyzer is a small program that uses relevant C features to process songs data to produce descriptive analytics.

b) Based on provided arguments and data files (i.e., datasets), song_analyzer will help us answer the following questions:

> **Q1**: What are the songs released before the 2020s where the only artist is 'Rae Spoon'?
>
> **Q2**: What are the songs released during the 2020s where the only artist is 'Tate McRae'.
>
> **Q3**: What are the songs released before the 2020s where the only artist is 'The Weeknd' and were written in the major scale?
>
> **Q4**: What are the songs released during the 2020s that are in more than 5000 Spotify playlists and were written in the D or A keys?
>
> **Q5**: What are the songs released in the period of 2021-2022 where 'Drake' is included in the list of artists?

c) After each execution, your program **must produce a file named output.csv** that represents the answer to the question asked to program (i.e., arguments passed).

d) The most reliable way (and the only one encouraged) to test your program is to use the provided **tester** file which will validate the output produced by your program, given a particular question.

Testing your solution

- The **tester** file will execute your program automatically, but you need to compile it first. You'll only need to pass the number of the question as an argument (e.g., `./tester 1`). If no arguments are passed to the tester file (i.e., `./tester`), it will run the tests for all the questions. Using a specialized library that compares the differences between .csv files, the

tester file will describe the differences between the expected output and the one provided by your program.

- **Refer to the example commands in the file "TESTS.md" for appropriate command line input** (i.e., your program should be executed according to the "Command automated by tester:" for each test in "TESTS.md"). Your solution must accommodate all specified command line inputs.
- Make sure to use the test files provided as part of this assignment.
- Use the test files and listed test cases to guide your implementation effort. Develop your program incrementally. Save stages of your incremental development in your Git repository.
- For this assignment you can assume all test inputs are well-formed (i.e., exception handling is not required).
- **DO NOT rely on visual inspection**. You can use the provided **tester** file so you can verify the validity of your outputs in a simpler manner.

Exercises for this assignment

To facilitate the development of song_analyzer, try to decompose the problem into small and more manageable parts. To promote this, we suggest dividing the assignment into the following parts.

Part 1: Argument processing

- Obtain the arguments from the command line using the char *argv[] parameter of your main function.
- To get the actual value of the argument, use the function strtok().

Part 2: Read file content

- You will need to read line by line the file passed as an argument to *song_analyzer*. The functions fopen() and fgets() together with for or while loops can readily accomplish this task.

Part 3: Process data lines

- Once your program can read all the lines from the input *.csv* file, you will need to process each line to obtain relevant data to produce the required output.
- Use the function strtok() again to obtain data.
- Another option is to create some representation of the songs in the file using a struct (optional but recommended).

Part 4: Write output

- Use the fputs() function (and format specifiers) to generate the required output in the output.csv file.

Requirements and recommendations

1. **You MUST use the -std=c99 flag when compiling your program as this will be used during assignment evaluation (i.e., the flag ensures the 1999 C standard is used during compilation).**
2. **DO NOT use malloc(), calloc() or any of the dynamic memory functions.** For this assignment you can define a constant for the longest input line (i.e., maximum number of characters).

3. Keep all of your code in one file for this assignment (that is, *song_analyzer.c*). In later assignments we will use the separable compilation facility available in C.
4. Use the test files to guide your implementation effort. Start with the simple example in test 01 and move onto 02, 03, etc. in order. **Refrain from writing the program all at once, and budget time to anticipate when things go wrong!**
5. For this assignment you can assume all test inputs will be well-formed (i.e., the teaching team will not evaluate your submission for handling of input or for arguments containing errors). Later assignments might specify error-handling as part of their requirements.
6. Use git when working on your assignment. Remember, that the ONLY acceptable method of submission is through Git.

What you must submit

A single C source file named **song_analyzer.c**, submitted to the a1 folder **in your Git repository**. Git is the **only** acceptable way of submission.

Evaluation

Refer to *seng265-spring2024-a1-evaluation.pdf* for the complete description of the assignment's evaluation.

Input specification

- All input test files are in the CSV format and are based on the "Most Streamed Spotify Songs 2023" dataset in Kaggle.com.
- The metadata (i.e., description of each column) for the input datasets can be found online in the 'Key Features' section of the dataset description.

Program Arguments

Argument	Description	Example (Test 02)
--question	The question to be answered by the program (e.g., 1, 2, 3, 4, 5)	`./song_analyzer --question=2 --data=during_2020s.csv`
--data	The input .csv file to be used (e.g., before_2020s.csv, during_2020s.csv)	

Output specification

- After execution, your program must produce\create the following file: **output.csv**. This file will contain two-dimensional data (i.e., a table with two columns) that represents the answer to the question passed as an argument to the program (e.g., **Q1**, **Q2**, **Q3**, **Q4**, **Q5**).
    - The first column refers to the name of the song's artist(s) and must be named 'Artist(s)'.
    - The second column refers to the name of the song and must be named 'Song'.