Junxi

After studying the source code, I noticed that a lot of code is duplicated for processing the two input CSV files. However, one of these duplicated code blocks is asymmetrical. Specifically, lines 72 and 77 are mismatched; line 72 is a return statement, while line 77 is not. What this means is that if the first input file has more lines than the second, the program will not complete the comparison correctly, restarting the loop in the main function. However, if the first input file has fewer lines than the second, the comparison works properly. A simple fix is to remove “return” from line 72, while a better fix is to make all the duplicated code into functions where possible, so that there is only one set of code to fix instead of two nearly identical ones.

There is also another issue, but I am not sure if it is considered a bug. The file paths to be accessed are partially hardcoded, thereby requiring a directory named “file” to exist. This is used for fetching the input files, and where the output file will be created. However, in the code, the “file” directory is hardcoded into the file path (see line 42 for example). The result is that the program will only run properly on Windows, and not on any POSIX-compliant OS due to the use of backslashes instead of forward slashes for directories. This is not necessarily a bug if the program is only intended to be used on Windows. However, I think a better way to implement this would be to allow the user to pass in the full file path themselves, instead of forcing them to use the “file” directory.

Swastik

Aside from the fact that this program is not generalised to work with any CSV files, there are a couple of bugs. The first one is an uncaught index out of bounds exception, which occurs when one of the input CSV files has a row with less than 5 entries. This is due to not checking the length of the “words_split” array (created in lines 51 and 95) before attempting to access the individual elements. A fix would be to add an if block after line 51 and 95 to check the size of the “words_split” array before proceeding to access any individual element.

The second bug is a race condition. I noticed that even though I intentionally failed to follow the format that is checked in the series of if blocks (lines 52 to 76 and lines 96 to 120) and that the correct error messages are printed out to me, the program proceeds to run anyway instead of terminating. I suspect this is due to the use of “if” instead of “else if” for the blocks in lines 57 to 73 and lines 101 to 117. It seems like doing so made the if blocks run in parallel. Unfortunately, this also includes the final if block which checks the boolean variable “incorrect” to see if it should terminate. As this is a relatively simple check as compared to the other if blocks, it always completes first, and since the “incorrect” variable is originally set to false, the program does not return. Then, when the other if blocks are completed and the “incorrect” variable is set to true, the final if block has already completed and will not be run again, so the program does not terminate. As such, there is a race condition where the “incorrect” variable is being read from before it is being written to. There are two simple fixes for this. The first is to use “else if” for all the if blocks except the first and last ones, which makes Java process the first 5 if blocks sequentially instead of in parallel, ensuring that the final if block will only run after the first 5 are completed. The second is to remove the “incorrect” variable, and have the if blocks return after printing the error message, eliminating the race condition entirely.

Both

One final note is that neither program checks for whether the commas are surrounded by quotation marks. This could result in issues for CSV files that use commas within the data entries themselves, such as CSV files that include book titles or object descriptions. These commas will be picked up as delimiters in these programs, instead of simply being part of a string of text. This is not necessarily a bug, since CSV files can follow different standards. However, an end user may not be aware of the standards followed by these CSV comparison programs, resulting in the programs behaving in unexpected ways to the end user. For example, an end user may create a spreadsheet about books, with fields like book title, synopsis, author(s) etc. Keying in commas as part of a text string in a spreadsheet will not cause any visible issues. Then, when the spreadsheet is exported as a CSV and run through these CSV comparison programs, they may be met with errors or incorrect results. As such, an end user may think the programs are bugged, when the programs are merely not compliant with certain standards.