Andrew Boothe
CSE 465
3/29/2023

In order to produce the best possible results for this homework, I used a simple implementation that kept me on the same initial path from start to finish. Firstly, I began the program by opening up both of the files that our code takes as Command-Line arguments. For example, if the files are a.tsv and a.tmp, they are now both an I/O filestream due to the simple open() function and are assigned to the variables f and g, respectively. Next, I assign each line of the tmp file (the template for our assignment) into an array with a simple for loop. This will allow me to easily update the array before writing it into our result files. Next, I have a for loop that contains the bulk of the code and loops through each line of our tab separated value file. In each line, we initially split it into an array of strings using the split() function (by tabs since it is a TSV file). Then, through a basic conditional I assign the *index* value to the line of the file where the titles of all the data are declared. Then I create an *elements* array that contains each title, placed in the array at their corresponding index. For the remainder of the loop, I create the new file with names from all the ID's, open those files for writing, and after pasting in all of the template from the .tmp files, I replace all the relevant tags (<<"val">>) with their corresponding values based on each line, and the index of the *elements* array.

This code in essence uses a variety of nested loops to accomplish what is necessary to pass all of the test cases, as well as some choice functions, such as replace(), split(), and .index(). While this code may not be the most efficient in terms of memory usage, it was the easiest for me to understand, and negated the need to toy with unnecessary dictionaries, data manipulation, etc. The use of the replace() function was the most useful in that it gave me full autonomy to change out the tags with the needed answers without modifying the file every time. The use of the split() function allowed me to fill the TSV file into a more manipulatable format. The use of the .index() function gave me the ability to determine which data element was which in the TSV file based on what index it was (index 1 is id, index 0 is name, etc.). The use of all these features allowed me to generate a simple program that solved all the test cases in less than 40 lines.