CS214 Project 0: Simple CSV sorter
Section Charmian Goh, Nikita Kolotov

Design:

- **simpleCSVsorter.h-** This file contains a typedef for a struct movie_data that is used to store each tuple from the csv file. It also contains a define initial_size that is 1 to be used during reallocation of memory when we need to expand the memory for more tuples. It also contains method declarations to helper methods to be used in simpleCSVsorter.c

- **mergesort.c -** Mergesort.c takes in an array of structs and contains 2 functions that will modify them. Mergesort will take the array and find the midpoint, then it will recursively split them until they are singular. Then it will throw those array elements into mergeArr and it will then merge the elements in ascending order into the original array. This function uses temp arrays that save all the data until no longer needed. At the end of mergesort, a sorted array of structs is released back into the main program.

- **simpleCSVsorter.c -** Aside from the main function. It consists of 3 helper functions. 2 functions are used to convert numeric values to a string value for printing and the remaining function is used to print all the variables inside a single movie_data struct. The main function intialises a buffer and Movie*. It then reads from stdin to the buffer once to remove the first tuple in the csv. The for loop reads each individual tuple into their respective variables in the struct, reallocating memory as the array grows. After that we check for the argument passed in the command line to check which column the csv file will be sorted upon. A while loop then prints out the sorted file.

Assumptions:

- CSV file inputted should contain the same number of columns as the csv provided for testing.
- The columns should be in the same order and have the same name as the CSV provided for testing.
- Each row should contain under 1000 characters.

Difficulties:

- In simpleCSVsorter.c , instead of using ‘strtok’ to tokenize the input, ‘strsep’ was used. Initially, using strtok, we quickly discovered it skips over null values in the csv. Using strsep, we had to use char* tok and end to correctly point to the input being tokenized.
- In mergesort.c, we had to hard code the category names because we couldn’t figure out why our program wouldn’t accept the category names as a variable. When using the

variable as a parameter in the mergesort function, the code would often break. Thus, we needed to hardcode the fields to ensure it would work for every case.

Testing:

We tested the program with various datasets of different sizes to check if memory reallocates properly. Different initial_sizes were also used. All programs were tested with smaller datasets to check if the mergesort was sorting correctly.