Usage:

1. The program reads from stdin. And output to stdout.
2. Command line:
    a. For sorting:
        i. sorter -c sorting_key
    b. For calculating correlation between two fields (For Extra Credit 1):
        i. sorter -r field1 field2
3. Example:
    a. For sorting:
        i. cat movie_metadata.csv | ./sorter -c color > output_01_color.csv
        ii. cat movie_metadata.csv | ./sorter -c director_name > output_02_director_name.csv
    b. For calculating correlation between two fields:
        i. cat movie_metadata.csv | ./sorter -r num_critic_for_reviews gross > correlation.txt
        ii. cat movie_metadata.csv | ./sorter -r duration gross >> correlation.txt

Design:

1. String comparison is not case sensitive.
2. Tried to use strtok to tokenize fields in a record. But quoted fields (with "," inside) make it very complicated. End up writing own tokenizer.
3. Store sorting key info in the Record struct, where the key is two char pointers, + long and float for possible data types:

```
typedef struct Record {
        // pointer to sorting key
        char* pSKey;

        /* pointer to terminating char of sorting key. It will be set to '\0' while
processing. And reset back before output record to file */
        char* pSKeyTerm;

        // Hold the char at *pSKeyTerm. Will be set back before output to file
        char chHold;

        // integer number if sorted by number
        long lKey;
        // floating point number is sorted by float number
        float fKey;

        /* String for one record. Define as array of 1. Will add more at allocation to fix
the string. */
        char recordData[1];
} Record;
```

- It is simpler to code and requires less resources (Memory and CPU).
- Originally, it parses the key and made a copy of it. It required more memory and malloc calls.

4. Originally, we try to use fgets to read a line. But its requirement that it should be able to read any length made memory management complicated.
    a. After research, we used getline instead, which automatically manages memory allocation.
5. Divide into small files like Object Orientated Programming.
    a. mergesort.c, mergesort.h:
        i. Merge sorting algorithm.
    b. record.c, record.h:
        i. Record is one line in file.
    c. recordarray.c, recordarray.h:
        i. Manage memory allocation.
    d. sortingkey.c, sortingkey.h:
        i. Finding sorting key.
    e. tokenizer.c, tokenizer.h:
        i. Tokenizer.
    f. helper.c, helper.h:
        i. Buffer.
    g. sorter.c, sorter.h:
        i. Main.
    h. global.h:
        i. Some defines
6. Automatically detect datatype of a field to sort:
    a. Integer/Long: All data are full number or empty field.
    b. Float: All data are float number, decimal or empty field. At least one float number.
    c. String: All others.
7. Test:
    a. Test sorting with all fields.
    b. Compare file sizes of output with input: they should be identical.

c. Make sure NULL are sorted correctly.
d. Make sure String, integer, and float data type sorted correctly.
e. We built a shell script to test all fields.

Assumptions:

1. Data follows csv conventions.
2. Machine have enough resources as all records are loaded into memory before sorting.

Extra Credit 1: Find something interesting about the data set

- We try to find out which factors are most related to the success of a movie. Using gross as the proxy of success, we calculated the following factors' correlation to gross:

Field	Correlation (with gross)
num_critic_for_reviews	0.480601
duration	0.250298
director_facebook_likes	0.144945
actor_3_facebook_likes	0.308026
actor_1_facebook_likes	0.154468
gross	1.000000
num_voted_users	0.637271
cast_total_facebook_likes	0.247400
facenumber_in_poster	-0.027755
num_user_for_reviews	0.559958
budget	0.102179
title_year	0.030886
actor_2_facebook_likes	0.262768
imdb_score	0.198021
aspect_ratio	0.069346
movie_facebook_likes	0.378082

- Here are some findings:
  1. num_voted_users are the most relevant to predict success.
  2. facenumber_in_poster has negative correlation to success. So, use less faces (no face or 1 face).
  3. budget is not a good indicator or success. So, a small company can have a chance.
  4. Social media counts have positive correlations. So, promoting movies in social media is a good idea.
  5. num_critic_for_reviews are a good indicator. So, ask many critics to review the movies.

Extra Credit 2: Find a way to generalize your sorter given *any* CSV file

- The program is coded to handle any csv files with the following considerations:
    1. Make no assumptions on the lengths of a record and a token.
    2. The program does not have any metadata about the file "movie_metadata.csv" hard-coded.
    3. Automatically detect data type of a field (integer/long number, float, or string) and perform sorting accordingly.
    4. If a record contains less fields than the header line, the missing field will be treated as NULL when try to sort.
    5. No additional metadata is needed.