```
// Jessica Elkins
// CS332 Lab 6
// 2/27/20
// This program converts listings.csv into a structure and sorts the structure by host
// name and outputs that to listingsHostName.csv and then sorts the structure by price
// and outputs that structure to listingsPrice.csv.

// TO COMPILE: $gcc lab6.c -o lab6
// TO RUN: ./lab6

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//given structure
struct listing {
        int id, host_id, minimum_nights, number_of_reviews,
            calculated_host_listings_count, availability_365;
        char *host_name, *neighbourhood_group, *neighbourhood, *room_type;
        float latitude, longitude, price;

};

//function that tokenizes csv into each corresponding structure variable
struct listing getfields(char* line) {
        struct listing item;
        item.id = atoi(strtok(line, ","));
        item.host_id = atoi(strtok(NULL, ","));
        item.host_name = strdup(strtok(NULL, ","));
        item.neighbourhood_group = strdup(strtok(NULL, ","));
        item.neighbourhood = strdup(strtok(NULL, ","));
        item.latitude = atof(strtok(NULL, ","));
        item.longitude = atof(strtok(NULL, ","));
        item.room_type = strdup(strtok(NULL, ","));
        item.price = atof(strtok(NULL, ","));
        item.minimum_nights = atoi(strtok(NULL, ","));
        item.number_of_reviews = atoi(strtok(NULL, ","));
        item.calculated_host_listings_count = atoi(strtok(NULL, ","));
        item.availability_365 = atoi(strtok(NULL, ","));

        return item;
}

//compare function for qsort that sorts by host name
// by returning string compare value
static int cmp_host(const void *p1, const void *p2) {
        struct listing *pointer1 = (struct listing *)p1;
        struct listing *pointer2 = (struct listing *)p2;
        return strcmp(pointer1->host_name, pointer2->host_name);
}

//compare function for qsort that sorts by price
// by returning
static int cmp_price(const void *p1, const void *p2) {
        struct listing *pointer1 = (struct listing *)p1;
        struct listing *pointer2 = (struct listing *)p2;
        // price is listed as a float for some reason so I
        // multiplied by 100 to try to get it to an integer for comparison
        return (int)(100*pointer1->price - 100*pointer2->price);
}

void sortByHostName(void) {
        struct listing list_items[22555];
```

```
        char line[BUFSIZ];
        int i, j, count;

        //opening the input file
        FILE *fptr1 = fopen("listings.csv", "r");

        //error message if file pointer is null
        if(fptr1 == NULL) {
                printf("Error reading input file listings.csv \n");
                exit(-1);
        }

        count = 0;
        while(fgets(line, BUFSIZ, fptr1) != NULL) {
                // reading the file line by line and sending
                // the line to get tokenized
                list_items[count++] = getfields(line);
        }

        // creating the output file
        FILE *fptr2 = fopen("listingsHostName.csv", "w");
        if(fptr2 == NULL) {
                printf("Error creating output file listingsHostName.csv. \n");
                exit(-1);
        }

        //sorting the array of structs
        qsort(list_items,count, sizeof(struct listing), cmp_host);

        //using fprintf to format the structures to csv format to the output file
        for(i = 0; i < count; i++) {
                fprintf(fptr2, "%d,%d,%s,%s,%s,%f,%f,%s,%f,%d,%d,%d,%d\n", list_items[i
].id, list_items[i].host_id, list_items[i].host_name,
                                                                        list_items[i].n
eighbourhood_group, list_items[i].neighbourhood,
                                                                        list_items[i].l
atitude, list_items[i].longitude, list_items[i].room_type,
                                                                        list_items[i].p
rice, list_items[i].minimum_nights,
                                                                        list_items[i].n
umber_of_reviews, list_items[i].calculated_host_listings_count,
                                                                        list_items[i].a
vailability_365);
        }

        // closing files
        fclose(fptr1);
        fclose(fptr2);

}

void sortByPrice(void) {
        struct listing list_items[22555];
        char line[BUFSIZ];
        int i, count;

        // opening input file
        FILE *fptr1 = fopen("listings.csv", "r");

        // error message if file pointer is equal to null
        if(fptr1 == NULL) {
                printf("Error reading input file listings.csv \n");
                exit(-1);
```

```
        }

        count = 0;
        while(fgets(line, BUFSIZ, fptr1) != NULL) {
                // reading input file line by line and sending
                // each line to get tokenized into a structure
                list_items[count++] = getfields(line);
         }

        // creating the output file
        FILE *fptr2 = fopen("listingsPrice.csv", "w");

        //error message if file point is equal to null
        if(fptr2 == NULL) {
                printf("Error creating output file listingsPrice.csv. \n");
                exit(-1);
        }

        // sorting array of stucts
        qsort(list_items,count, sizeof(struct listing), cmp_price);

        // using fprintf to format structs into csv format and put them in the output f
ile
        for(i = 0; i < count; i++) {
                fprintf(fptr2, "%d,%d,%s,%s,%s,%f,%f,%s,%f,%d,%d,%d,%d\n", list_items[i
].id, list_items[i].host_id, list_items[i].host_name,
                                                                        list_items[i
].neighbourhood_group, list_items[i].neighbourhood,

                                                                        list_items[i
].latitude, list_items[i].longitude, list_items[i].room_type,
                                                                        list_items[i
].price, list_items[i].minimum_nights,

                                                                        list_items[i
].number_of_reviews, list_items[i].calculated_host_listings_count,
                                                                        list_items[i
].availability_365);
        }

        //closing the files
        fclose(fptr1);
        fclose(fptr2);
}



int main(int argc, char **args) {

        // calling sortByHostName to sort listings.csv by host name
        sortByHostName();

        //calling sortByPrice to sort listings.csv by price
        sortByPrice();

        return 0;
}
```