The full program – the implementation of a system for managing film data and viewers' reviews

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#define STUDIO_SIZE 30
#define COUNTRY_SIZE 15
#define BUFFER_SIZE 512
#define NUMBER_OF_COUNTRIES 256 // The number of contries in the world

// Structure to represent a vot
typedef struct
{
	int value; // the grade of the reviewer
	char* p2comment; // comment
	char country[COUNTRY_SIZE]; // origin country
}vote;

// Structure to represent a movie
typedef struct
{
	int id; // the id of the movie
	char* p2name; // movie name
	char* p2genre; // movie genre
	char studio[STUDIO_SIZE]; // studio name
	int year; // year of release
	vote* p2list; // pointer to the dynamic array of the vote structre
	int Nvote; // number of votes
}movie;


// Function prototypes
int countLines(const char* file_name);
void FromFile2Movies(const char* file_name, movie* ptr_movie_array, int number_of_movies);
void FromFile2Votes(const char* file_name, movie* ptr_movie_array, int number_of_movies);
int addMovie(movie** ptr_movie_array, int* number_of_movies);
int addVote(movie* ptr_movie_array, int number_of_movies, int movie_id);
void printMenu(movie** ptr_movie_array, int* number_of_movies, const char* movies_file_name,
const char* votes_file_name);
void writeToFiles(const char* movies_file_name, const char* votes_file_name, movie*
ptr_movie_array, int number_of_movies);
int printVotes(const char* movie_name, movie* ptr_movie_array, int number_of_movies);
void printValue(int value, char country[COUNTRY_SIZE], movie* ptr_movie_array, int
number_of_movies);
void maxByCountry(movie* ptr_movie_array, int number_of_movies);
void RecommendMovie(movie* ptr_movie_array, int number_of_movies, int vote_value);
void deleteWorst(const char* genre_name, movie* ptr_movie_array, int number_of_movies);

// Main function
void main()
{	// File names for movies and votes data
	const char* movies_file_name = "moviesData.txt";
	const char* votes_file_name = "votingData.txt";

	// Count the number of movies from the file
	int number_of_movies = countLines(movies_file_name);

	// Allocate memory for movies array
	movie* movies = (movie*)malloc(number_of_movies * sizeof(movie));
	if (movies == NULL) {
		printf("Error - Memory allocation failed.\n");
		return 1;
	}

	printf("Number of movies: %d\n", number_of_movies);

	// Populate movies array from file
	FromFile2Movies(movies_file_name, movies, number_of_movies);
	// Populate votes for movies from file
	FromFile2Votes(votes_file_name, movies, number_of_movies);

	// Display menu and handle user input
	printMenu(&movies, &number_of_movies, movies_file_name, votes_file_name);

	// Free memory for the movie array and its components
	for (int i = 0; i < number_of_movies; i++) {
		free(movies[i].p2name);
		free(movies[i].p2genre);
		if (movies[i].p2list != NULL) {
			for (int j = 0; j < movies[i].Nvote; j++) {
				free(movies[i].p2list[j].p2comment);
			}
			if (movies[i].Nvote > 0) {
				free(movies[i].p2list);
			}
		}
	}
	free(movies);

	system("pause");
}


// This function counts the number of lines in a text file.
int countLines(const char* file_name)
{
	int line_counter = 0; // Initialize line counter
	char buffer[BUFFER_SIZE]; // Buffer to read lines efficiently
	FILE* file = fopen(file_name, "r"); // Open the file for reading

	if (file == NULL)
	{
		printf("Error - The file %s could not be opened.\n", file_name); // Prints error
message with explanation
		return -1; // Error code indicating failure
	}

	// Read each line from the file using fgets
	while (fgets(buffer, sizeof(buffer), file) != NULL)
	{
		line_counter++;
	}

	fclose(file); // Close the file
	return line_counter - 1;// Return the total number of lines counted and ignore the
first line.
}

// Function to populate movies array from a file
void FromFile2Movies(const char* file_name, movie* ptr_movie_array, int number_of_movies)
{
	FILE* file = fopen(file_name, "r"); // Open the file for reading

	if (file == NULL)
	{
		printf("Error - The file %s could not be opened.\n", file_name); // Prints error
message
		return; // Return without processing further
	}

	char buffer[BUFFER_SIZE];// Buffer to read lines from the file
	char buffer_name[BUFFER_SIZE]; // Buffer for movie name
	char buffer_genre[BUFFER_SIZE]; // Buffer for movie genre

	fgets(buffer, BUFFER_SIZE, file); // Read and discard first line
	// Read movie data from file
	int i = 0;
	while (i < number_of_movies && fscanf(file, "%d,%[^,],%[^,],%[^,],%d\n",
&ptr_movie_array[i].id,
		buffer_name, buffer_genre, ptr_movie_array[i].studio, &ptr_movie_array[i].year)
== 5) {	//In this case, [^,] means it will read characters until it encounters a comma
(,).

		// Allocate memory for p2name and p2genre dynamically
		ptr_movie_array[i].p2name = (char*)malloc((strlen(buffer_name) + 1) *
sizeof(char));
		ptr_movie_array[i].p2genre = (char*)malloc((strlen(buffer_genre) + 1) *
sizeof(char));

		// Check memory allocation
		if (ptr_movie_array[i].p2name == NULL || ptr_movie_array[i].p2genre == NULL) {
			printf("Error - Memory allocation failed for movie %d. Skipping...\n", i +
1);
			// Free previously allocated memory
			for (int j = 0; j < i; j++) {
				free(ptr_movie_array[j].p2name);
				free(ptr_movie_array[j].p2genre);
			}
			fclose(file);
			return;
		}

		// Copy the read strings to allocated memory
		strcpy(ptr_movie_array[i].p2name, buffer_name);
		strcpy(ptr_movie_array[i].p2genre, buffer_genre);
		ptr_movie_array[i].Nvote = 0; // Initialize vote count
		i++;

	}

	fclose(file);  // Close the file
}

// Function to read vote data from a file and update the movie array
void FromFile2Votes(const char* file_name, movie* ptr_movie_array, int number_of_movies) {
	FILE* file = fopen(file_name, "r"); // Open the file for reading

	if (file == NULL) {
		printf("Error - The file %s could not be opened.\n", file_name); // Prints error
message
		return;  // Return without processing further
	}

	char buffer[BUFFER_SIZE]; // Buffer to store each line read from the file
	char country[COUNTRY_SIZE]; // Array to store the country of the vote
	char comment[BUFFER_SIZE]; // Buffer to store the comment associated with the vote
	int movie_id, vote_value; // Variables to store movie ID and vote value
	int index; // Index variable for accessing movie array

	// Read and discard the header line
	fgets(buffer, BUFFER_SIZE, file);

	// Read vote data from the file line by line
	while (fgets(buffer, sizeof(buffer), file) != NULL) {
		// Parse the line to extract vote details
		sscanf(buffer, "%d:%d:%[^:]:%[^\n]", &movie_id, &vote_value, country, comment);

		// Find the index of the movie in ptr_movie_array
		for (index = 0; index < number_of_movies; index++) {
			if (ptr_movie_array[index].id == movie_id) {
				break;
			}
		}
```

```c
            // Check if movie ID is valid
            if (index == number_of_movies) {
                 continue; // Move to the next line
            }

            // Initialize vote count and allocate memory for vote list if necessary
            if (ptr_movie_array[index].Nvote == 0) {
                 ptr_movie_array[index].p2list = (vote*)malloc(sizeof(vote));
                 if (ptr_movie_array[index].p2list == NULL) {
                      printf("Error - Memory allocation failed for votes of movie ID
%d.\n", movie_id);
                      continue; // Move to the next line
                 }
            }
            else {
                 // Reallocate memory to expand the vote list
                 vote* temp = realloc(ptr_movie_array[index].p2list,
(ptr_movie_array[index].Nvote + 1) * sizeof(vote));
                 if (temp == NULL) {
                      printf("Error - Memory reallocation failed for votes of movie ID
%d.\n", movie_id);
                      continue; // Move to the next line
                 }
                 ptr_movie_array[index].p2list = temp;
            }

            // Allocate memory for the comment
            ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment =
(char*)malloc((strlen(comment) + 1) * sizeof(char));
            if (ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment ==
NULL) {
                 printf("Error - Memory allocation failed for votes of movie ID %d.\n",
movie_id);
                 continue; // Move to the next line
            }

            // Update the vote information
            ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].value = vote_value;
            strncpy(ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].country,
country, COUNTRY_SIZE - 1);
            ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].country[COUNTRY_SIZE
- 1] = '\0'; // Ensure null termination
            strcpy(ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment,
comment);

            // Increment the vote count for the movie
            ptr_movie_array[index].Nvote++;
       }

       fclose(file); // Close the file
}

// Function to add a new movie to the movie array
int addMovie(movie** ptr_movie_array, int* number_of_movies) {
       int year; // Variables to store year of release
       char name[BUFFER_SIZE], genre[BUFFER_SIZE], studio[STUDIO_SIZE]; // Buffers to store
movie name, genre, and studio

       // Get input from the user for movie details
       printf("Enter movie name: ");
       scanf(" %[^\n]s", name);

       // Check if the movie already exists in the array
       for (int i = 0; i < *number_of_movies; i++) {
            if (((*ptr_movie_array)[i].p2name != NULL) &&
(strcmp((*ptr_movie_array)[i].p2name, name) == 0))
            {
                 printf("Error - Movie with the name '%s' already exists.\n", name);
                 return 0; // Return 0 indicating failure
            }
       }

       printf("Enter movie genre: ");
       scanf(" %[^\n]s", genre);
       printf("Enter studio name: ");
       scanf(" %[^\n]s", studio);
       printf("Enter year of release: ");
       scanf("%d", &year);

       // Reallocate memory for the movie array to accommodate the new movie
       movie* temp = realloc(*ptr_movie_array, (*number_of_movies + 1) * sizeof(movie));
       if (temp == NULL) {
            printf("Error - Memory reallocation failed for adding a new movie.\n");
            return 0; // Return 0 indicating failure
       }
       *ptr_movie_array = temp;

       // Allocate memory for the new movie
       (*ptr_movie_array)[*number_of_movies].p2name = (char*)malloc((strlen(name) + 1) *
sizeof(char));
       (*ptr_movie_array)[*number_of_movies].p2genre = (char*)malloc((strlen(genre) + 1) *
sizeof(char));

       if ((*ptr_movie_array)[*number_of_movies].p2name == NULL ||
(*ptr_movie_array)[*number_of_movies].p2genre == NULL) {
            printf("Error - Memory allocation failed for new movie. Skipping...\n");
            // Free the memory allocated for the newly added movie structure
            free((*ptr_movie_array)[*number_of_movies].p2name);
            free((*ptr_movie_array)[*number_of_movies].p2genre);
            return 0; // Return 0 indicating failure
       }

       // Copy movie details to the new movie structure
       (*ptr_movie_array)[*number_of_movies].id = *number_of_movies + 1;
       strcpy((*ptr_movie_array)[*number_of_movies].p2name, name);
       strcpy((*ptr_movie_array)[*number_of_movies].p2genre, genre);
       strcpy((*ptr_movie_array)[*number_of_movies].studio, studio);
       (*ptr_movie_array)[*number_of_movies].year = year;
       (*ptr_movie_array)[*number_of_movies].Nvote = 0;

       (*number_of_movies)++; // Increment number_of_movies
       return 1; // Return 1 indicating success
}
// Function to add a vote for a movie
int addVote(movie* ptr_movie_array, int number_of_movies, int movie_id) {
       int index;  // Index variable for accessing movie array
       // Find the index of the movie in ptr_movie_array
       for (index = 0; index < number_of_movies; index++) {
            if (ptr_movie_array[index].id == movie_id) {
                 break;
            }
       }

       // Check if movie ID is valid
       if (index == number_of_movies) {
            printf("Error - Movie with ID %d not found.\n", movie_id);
            return 0; // Return 0 indicating failure
       }

       // Get input from the user for vote details
       int vote_value; // Variable to store the vote value
       char country[COUNTRY_SIZE]; // Array to store the country of the vote
       char comment[BUFFER_SIZE]; // Buffer to store the comment associated with the vote

       printf("Enter vote value: ");
       scanf("%d", &vote_value);
       printf("Enter country: ");
       scanf(" %[^\n]s", country);
       printf("Enter comment: ");
       // Clear input buffer before reading comment
       int c;
       // We will get input char by char from the user until the user press enter
       while ((c = getchar()) != '\n' && c != EOF);
       fgets(comment, sizeof(comment), stdin); // Read string from user

       if (strcmp(comment, "\n") == 0) { //checks if the comment is empty
            strcpy(comment, "-");      //- means an empty comment
       }
       else {
            // Remove newline character if present
            comment[strcspn(comment, "\n")] = 0;
       }

       // Check if the vote already exists in the list of votes for this movie
       for (int i = 0; i < ptr_movie_array[index].Nvote; i++) {
            if (strcmp(ptr_movie_array[index].p2list[i].country, country) == 0 &&
                 strcmp(ptr_movie_array[index].p2list[i].p2comment, comment) == 0 &&
                 ptr_movie_array[index].p2list[i].value == vote_value) {
                 printf("Error - This vote already exists for this movie.\n");
                 return 0; // Return 0 indicating failure
            }
       }

       // Allocate memory for the vote list if it's not allocated yet
       if (ptr_movie_array[index].Nvote == 0) {
            ptr_movie_array[index].p2list = malloc(sizeof(vote));
            if (ptr_movie_array[index].p2list == NULL) {
                 printf("Error - Memory allocation failed for adding a new vote.\n");
                 return 0; // Return 0 indicating failure
            }
       }
       else {
            // Reallocate memory to expand the vote list if its already allocated
            vote* temp = realloc(ptr_movie_array[index].p2list, (ptr_movie_array[index].Nvote
+ 1) * sizeof(vote));
            if (temp == NULL) {
                 printf("Error - Memory reallocation failed for adding a new vote.\n");
                 return 0; // Return 0 indicating failure
            }
            ptr_movie_array[index].p2list = temp;
       }

       // Allocate memory for the new vote comment
       ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment =
(char*)malloc((strlen(comment) + 1) * sizeof(char));
       if (ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment == NULL) {
            printf("Error - Memory allocation failed for new vote. Skipping...\n");
            return 0; // Return 0 indicating failure
       }

       // Update the vote information
       ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].value = vote_value;
       strncpy(ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].country, country,
COUNTRY_SIZE - 1);
       ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].country[COUNTRY_SIZE - 1] =
'\0'; // Ensure null termination
       strcpy(ptr_movie_array[index].p2list[ptr_movie_array[index].Nvote].p2comment, comment);

       // Increment the vote count for the movie
       ptr_movie_array[index].Nvote++;

       return 1; // Return 1 indicating success
}


// Function to write movie and vote data to files
```

```
void writeToFiles(const char* movies_file_name, const char* votes_file_name, movie*
ptr_movie_array, int number_of_movies) {
	// Write movie data to moviesData.txt
	FILE* movies_file = fopen(movies_file_name, "w");
	if (movies_file == NULL) {
		printf("Error - Cannot open moviesData.txt for writing.\n");
		return;
	}

	// Writing format to moviesData.txt
	fprintf(movies_file, "format:m_id,movie_name,Genre,Lead Studio,Year\n");
	int check_num_of_movies = 1;
	while (check_num_of_movies-1 != number_of_movies) //Arranges the list of movies in
ascending order
	{
		for (int i = 0; i < number_of_movies; i++)
		{
			if (ptr_movie_array[i].id == check_num_of_movies)
			{
				fprintf(movies_file, "%d,%s,%s,%s,%d\n", ptr_movie_array[i].id,
ptr_movie_array[i].p2name,
					ptr_movie_array[i].p2genre, ptr_movie_array[i].studio,
ptr_movie_array[i].year);
			}
		}
		check_num_of_movies++;
	}
	fclose(movies_file);// close the moviesData.txt file

	// Write vote data to votingData.txt
	FILE* votes_file = fopen(votes_file_name, "w");
	if (votes_file == NULL) {
		printf("Error - Cannot open votingData.txt for writing.\n");
		return;
	}

	// Writing format to votingData.txt
	fprintf(votes_file, "format:m_id:vote:country:comment //- means an empty comment\n");
	check_num_of_movies = 1;
	while (check_num_of_movies - 1 != number_of_movies)
	{
		for (int i = 0; i < number_of_movies; i++)
		{
			if (ptr_movie_array[i].id == check_num_of_movies)
			{
				for (int j = 0; j < ptr_movie_array[i].Nvote; j++)
				{
					fprintf(votes_file, "%d:%d:%s:%s\n", ptr_movie_array[i].id,
ptr_movie_array[i].p2list[j].value,
						ptr_movie_array[i].p2list[j].country,
ptr_movie_array[i].p2list[j].p2comment);
				}
			}
		}
		check_num_of_movies++;
	}

	fclose(votes_file); // close the votingData.txt file

	printf("Data written to files successfully.\n");
}
// Function to print all comments and countries of a movie
int printVotes(const char* movie_name, movie* ptr_movie_array, int number_of_movies) {
	int i, j;
	for (i = 0; i < number_of_movies; i++) { // goes through the array of movies
		if (strcmp(ptr_movie_array[i].p2name, movie_name) == 0) { // Check if the movie
name in the movie array is equal to the movie name that the function gets
			if (ptr_movie_array[i].Nvote == 0) {
				printf("No votes available for this movie.\n");
```

```c
                        return 0; // Return 0 indicating empty votes array
                    }
                    printf("Comments and countries for movie '%s':\n", movie_name);
                    for (j = 0; j < ptr_movie_array[i].Nvote; j++) { // goes through the votes
of the movie
                        printf("Grade: %d comment: %s (Country: %s)\n",
ptr_movie_array[i].p2list[j].value, ptr_movie_array[i].p2list[j].p2comment,
ptr_movie_array[i].p2list[j].country);
                    }
                    return 1; // Return 1 indicating success
                }
        }
        printf("Movie '%s' not found.\n", movie_name);
        return -1; // Return -1 indicating movie not found
}

// Function to count and print movies from a specific genre
void countGenre(const char* genre_name, movie* ptr_movie_array, int number_of_movies) {
        int found = 0; // Variable to track if any movie of the specified genre is found
        printf("Movies with genre '%s':\n", genre_name);
        for (int i = 0; i < number_of_movies; i++) { // goes through the array of movies
                if (strcmp(ptr_movie_array[i].p2genre, genre_name) == 0) { // Check if the genre
name in the movie array is equal to the genre name that the function gets
                        printf("- %s\n", ptr_movie_array[i].p2name);
                        found = 1;
                }
        }
        if (!found) {
                printf("No movies found with genre '%s'.\n", genre_name);
        }
}


// Function to print all movie names that got a specific value and country
void printValue(int value, char country[COUNTRY_SIZE], movie* ptr_movie_array, int
number_of_movies)
{
        int index = 0, found = 0;
        printf("This is the list of movies that get %d value from %s:\n", value, country);
        for (index = 0; index < number_of_movies; index++) // goes through the array of movies
        {
                // Checks if the vote value equals to the value that function gets and checks if
the vote of the country eqaul to the country name that function gets.
                if ((ptr_movie_array[index].p2list->value == value) &&
(strcmp(ptr_movie_array[index].p2list->country, country) == 0))
                {
                        found++; // Updating that the list is not empty
                        printf("Movie number %d - %s\n", found, ptr_movie_array[index].p2name);
                }
        }
        if (found == 0)
        {
                printf("The list is empty.");
        }
}

// Function to count the number of different countries that voted for movies from a given year
void countCountry(int year, movie* ptr_movie_array, int number_of_movies) {
        int countries_count = 0; // Variable to store the count of different countries
        char countries[NUMBER_OF_COUNTRIES][COUNTRY_SIZE]; // Array to store the unique
countries
        int index;

        // Initialize the countries array
        for (index = 0; index < number_of_movies; index++) {
                countries[index][0] = '\0';
        }

        // Iterate through movies and their votes to count unique countries
```

```
	for (index = 0; index < number_of_movies; index++) { // Goes through the array of 
movies
		if (ptr_movie_array[index].year == year) { // Checks if the movie year is equals 
to the year that function gets.
			for (int j = 0; j < ptr_movie_array[index].Nvote; j++) { // goes through 
the movie votes list
				int found = 0;
				// Check if the country is already counted
				for (int k = 0; k < countries_count; k++) {// goes through the 
countries array
					if (strcmp(ptr_movie_array[index].p2list[j].country, 
countries[k]) == 0) { // Checks if the vote country equals to the country name in the 
countries array.
						found = 1;
						break;
					}
				}
				// If the country is not counted, add it to the list of unique 
countries
				if (!found) {
					strcpy(countries[countries_count], 
ptr_movie_array[index].p2list[j].country);
					countries_count++; // Updating the count of the countries.
				}
			}
		}
	}

	// Print the count of unique countries
	printf("Number of different countries that voted for movies from %d: %d\n", year, 
countries_count);
}

// Function to count the number of comments for each country
void maxByCountry(movie* ptr_movie_array, int number_of_movies) {
	int max_comment_count = 0; // Variable to store the maximum comment count

	// Struct to store country comment count
	typedef struct {
		char country[COUNTRY_SIZE]; // country name.
		int comment_count; // number of comments for the country.
	} CountryCommentCount;

	// this array contains the name of contries and number of comments.
	CountryCommentCount country_counts[NUMBER_OF_COUNTRIES];
	int acutal_number_of_conutries = 0; // The actual number of countries of comments we 
found.

	// Initialize comment counts
	for (int i = 0; i < NUMBER_OF_COUNTRIES; i++) {
		country_counts[i].comment_count = 0;
	}

	// Iterate through each movie and its votes to count comments by country
	for (int i = 0; i < number_of_movies; i++) { // goes through the array of movies
		for (int j = 0; j < ptr_movie_array[i].Nvote; j++) { // goes through the movie 
votes list
			if (ptr_movie_array[i].p2list[j].p2comment[0] != '-') { // Check if 
comment is not empty
				int found = 0;
				// Search if country already exists in country_counts
				for (int k = 0; k < acutal_number_of_conutries; k++) { //// goes 
through the country_counts array.
					if (strcmp(ptr_movie_array[i].p2list[j].country, 
country_counts[k].country) == 0) { // Checks if the vote country equals to the country name in 
the count country array.
						country_counts[k].comment_count++; // updating the 
number of comments for this country
						found = 1; // Updating that the country was found
```

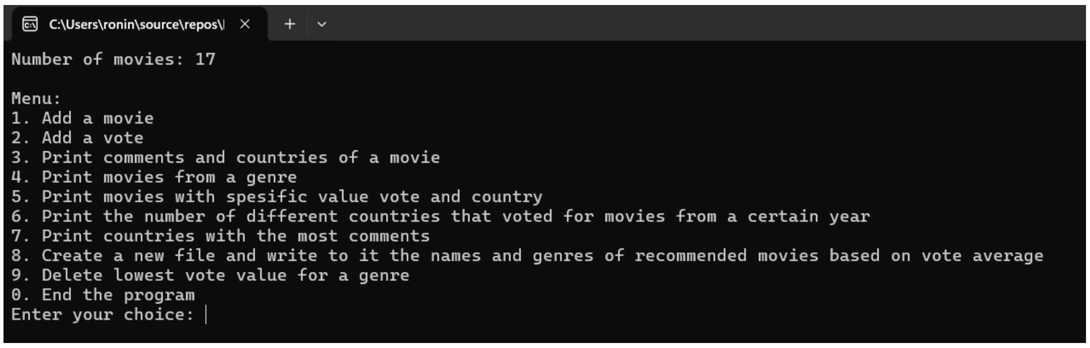
```
                                        break;
                                }
                        }
                        if (!found) { // Country not found, add to country_counts
                                strcpy(country_counts[acutal_number_of_conutries].country, 
ptr_movie_array[i].p2list[j].country); // Update the country name
                                country_counts[acutal_number_of_conutries].comment_count = 1; 
// The comments count is 1 because its the first time we found comment for this country.
                                acutal_number_of_conutries++; // Updating actual number of 
countries
                        }
                }
        }
    }

    // Find the maximum comment count
    for (int i = 0; i < acutal_number_of_conutries; i++) { // Goes through the country 
count array.
        if (country_counts[i].comment_count > max_comment_count) { // Checks if the count 
comment for this country is bigger than max_comment_count value.
            max_comment_count = country_counts[i].comment_count; // Update the 
max_comment_count value.
        }
    }

    // Print all countries with the maximum comment count
    printf("Countries with the most comments:\n");
    for (int i = 0; i < acutal_number_of_conutries; i++) { // Goes thourgh the country 
comments array
        if (country_counts[i].comment_count == max_comment_count) { // Checks if the 
comments count for this country is equal to the max_comment_count value.
            printf("- %s: %d comments\n", country_counts[i].country, 
country_counts[i].comment_count);
        }
    }
}

// Function to recommend movies based on vote average
void RecommendMovie(movie* ptr_movie_array, int number_of_movies, int vote_value) {
    //Create Recommendation.txt file
    FILE* recommendation_file = fopen("Recommendation.txt", "w");
    if (recommendation_file == NULL) { // Checks if the open file succeed
        printf("Error - Cannot open Recommendation.txt for writing.\n");
        return;
    }

    fprintf(recommendation_file, "Movies with vote average greater than or equal to %d:\n", 
vote_value);

    int found = 0; // Flag to check if any movies are found
    for (int i = 0; i < number_of_movies; i++) { // Goes through the array of movies
        int total_votes = 0;
        int total_value = 0;
        for (int j = 0; j < ptr_movie_array[i].Nvote; j++) { // Goes through the movie 
votes
            total_votes++; // Updates total votes
            total_value += ptr_movie_array[i].p2list[j].value; // Updates total value
        }
        float average = (float)total_value / total_votes; // Calculates the average
        if (average >= vote_value) { // Checks if the average vote value for specific 
movie is bigger than vote value that function gets
            fprintf(recommendation_file, "%s, %s\n", ptr_movie_array[i].p2name, 
ptr_movie_array[i].p2genre);
            found = 1; // Updates that we found movie
        }
    }

    if (!found) {
        fprintf(recommendation_file, "No movies found with vote average greater than or
equal to %d.\n", vote_value);
    }

    fclose(recommendation_file);
    printf("Recommendation.txt created successfully.\n");
}

// Function to delete the lowest vote value for a given genre
void deleteWorst(const char* genre_name, movie* ptr_movie_array, int number_of_movies) {
    int found = 0; // Variable to track if the genre is found
    int min_vote = 10; // Initialize minimum vote value to an arbitrary high value

    // Find the minimum vote value for the given genre
    for (int i = 0; i < number_of_movies; i++) { // Goes through the array of movies
        if (strcmp(ptr_movie_array[i].p2genre, genre_name) == 0) { // Checks if the movie
genre is equals to the genre that function gets.
            found = 1; // Genre found
            for (int j = 0; j < ptr_movie_array[i].Nvote; j++) { // Goes through the
movie votes
                if (ptr_movie_array[i].p2list[j].value < min_vote) { // Checks if
the vote value is smaller than the min_vote value
                    min_vote = ptr_movie_array[i].p2list[j].value; // Updates the
min_vote
                }
            }
        }
    }

    // Delete all votes with the minimum vote value for the genre
    if (found) {
        for (int i = 0; i < number_of_movies; i++) { /// Goes through the array of movies
            if (strcmp(ptr_movie_array[i].p2genre, genre_name) == 0) {  // Checks if
the movie genre is equals to the genre that function gets.
                for (int j = 0; j < ptr_movie_array[i].Nvote; j++) { // Goes through
the movie votes
                    if (ptr_movie_array[i].p2list[j].value == min_vote) { //
Checks if the vote value is equal to the min_vote value
                        // Free memory allocated for comment
                        free(ptr_movie_array[i].p2list[j].p2comment);
                        // Shift votes to remove the deleted one
                        for (int k = j; k < ptr_movie_array[i].Nvote - 1; k++)
{ // Goes through the vote array from the index that we found the min vote.
                            ptr_movie_array[i].p2list[k] =
ptr_movie_array[i].p2list[k + 1]; // The current index which we are on is updated to its next
index.
                        }
                        ptr_movie_array[i].Nvote--; // Decrease vote count
                        // Reallocate memory for the smaller array of votes
                        ptr_movie_array[i].p2list =
realloc(ptr_movie_array[i].p2list, ptr_movie_array[i].Nvote * sizeof(vote));
                        if (ptr_movie_array[i].Nvote == 0) {
                            // If no votes left, free memory for the array
of votes and set pointer to NULL

                            free(ptr_movie_array[i].p2list);
                            ptr_movie_array[i].p2list = NULL;
                        }
                        j--; // Adjust index after deletion
                    }
                }
            }
        }
        printf("The deletion of the lowest ranking movie of the given genre was
succesfull.\n", genre_name);
    }
    else {
        printf("Genre '%s' not found.\n", genre_name);
    }
}
```

```c
// Function to display the menu and handle user input
void printMenu(movie** ptr_movie_array, int* number_of_movies, const char* movies_file_name,
const char* votes_file_name) {
	int choice; // Variable to store the user's choice

	do {
		printf("\nMenu:\n");
		printf("1. Add a movie\n");
		printf("2. Add a vote\n");
		printf("3. Print comments and countries of a movie\n");
		printf("4. Print movies from a genre\n");
		printf("5. Print movies with spesific value vote and country\n");
		printf("6. Print the number of different countries that voted for movies from a
certain year\n");
		printf("7. Print countries with the most comments\n");
		printf("8. Create a new file and write to it the names and genres of recommended
movies based on vote average\n");
		printf("9. Delete lowest vote value for a genre\n");
		printf("0. End the program\n");
		printf("Enter your choice: ");
		scanf("%d", &choice);

		switch (choice) {
		case 1:
			if (addMovie(ptr_movie_array, number_of_movies)) {
				printf("Movie is added\n");
			}
			break;
		case 2: {
			int movie_id; // Variable to store the movie ID for adding a vote
			printf("Enter the ID of the movie you want to add a vote to: ");
			scanf("%d", &movie_id);
			if (addVote(*ptr_movie_array, *number_of_movies, movie_id)) {
				printf("Vote is added successfully.\n");
			}
			else {
				printf("Failed to add vote.\n");
			}
			break;
		}
		case 3: {
			char movie_name[BUFFER_SIZE]; // Variable to store the movie name to print
its comments and countries.
			printf("Enter movie name to print comments and countries: ");
			scanf(" %[^\n]s", movie_name);
			int result = printVotes(movie_name, *ptr_movie_array, *number_of_movies);
			if (result == 0) {
				printf("Votes array is empty.\n");
			}
			else if (result == -1) {
				printf("Movie '%s' not found.\n", movie_name);
			}
			break;
		}

		case 4: {
			char genre_name[BUFFER_SIZE]; // Variable to store the genre name to print
its related movies.
			printf("Enter genre name to count and print movies: ");
			scanf(" %[^\n]s", genre_name);
			countGenre(genre_name, *ptr_movie_array, *number_of_movies);

			break;
		}
		case 5:
		{
			int vote_value = 0; // Variable to store the value of the vote
			char Country[COUNTRY_SIZE]; // Variable to store the country name
```

```
                        printf("Enter value: ");
                        scanf("%d", &vote_value);
                        printf("Enter country: ");
                        scanf(" %[^\n]s", Country);
                        printValue(vote_value, Country, *ptr_movie_array, *number_of_movies);

                        break;
                }
                case 6: {
                        int year; // Variable to store the year
                        printf("Enter the year: ");
                        scanf("%d", &year);
                        countCountry(year, *ptr_movie_array, *number_of_movies);
                        break;
                }
                case 7:
                        maxByCountry(*ptr_movie_array, *number_of_movies);
                        break;
                case 8: {
                        int vote_value;  // Variable to store the value of the vote
                        printf("Enter the vote value to recommend movies: ");
                        scanf("%d", &vote_value);
                        RecommendMovie(*ptr_movie_array, *number_of_movies, vote_value);
                        break;
                }
                case 9: {
                        char genre_name[BUFFER_SIZE];  // Variable to store the genre name
                        printf("Enter genre name to delete lowest vote value: ");
                        scanf(" %[^\n]s", genre_name);
                        deleteWorst(genre_name, *ptr_movie_array, *number_of_movies);
                        break;
                }
                case 0:
                        printf("Ending the program.\n");
                        // Write data to files before exiting
                        writeToFiles(movies_file_name, votes_file_name, *ptr_movie_array,
*number_of_movies);
                        break;
                default:
                        printf("Invalid choice. Please choose again.\n");
                }
        } while (choice != 0);
}
```

Screenshots:

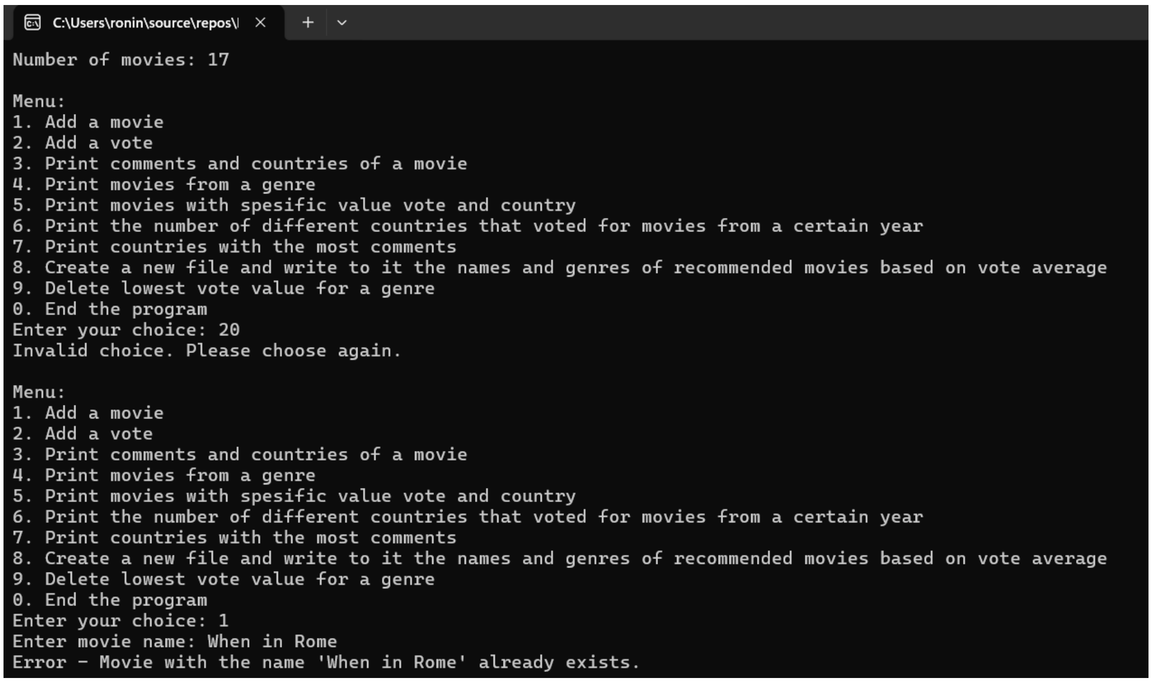
Main Menu:



```
C:\Users\ronin\source\repos\

Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice:
```

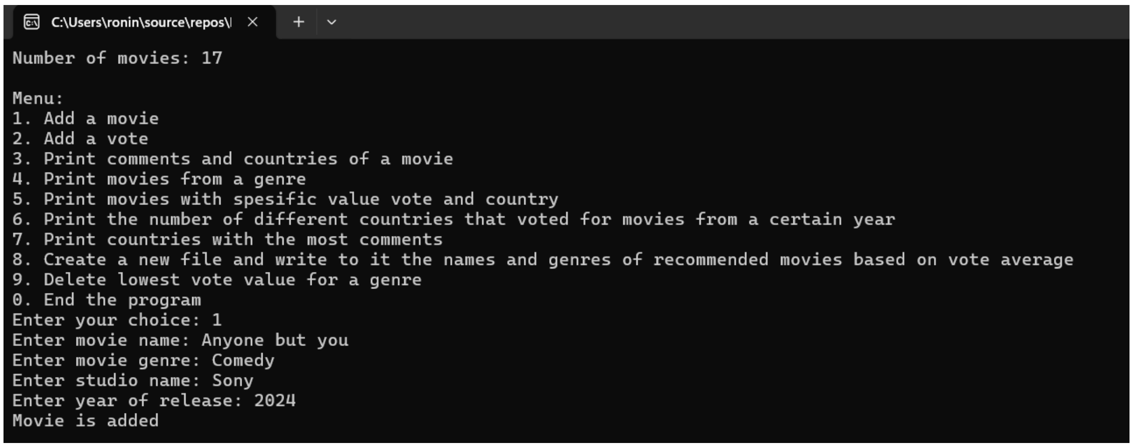
Choice "1" - addMovie:

In the case of a movie that already exists or choosing an option that does not exist in the menu, we will get:



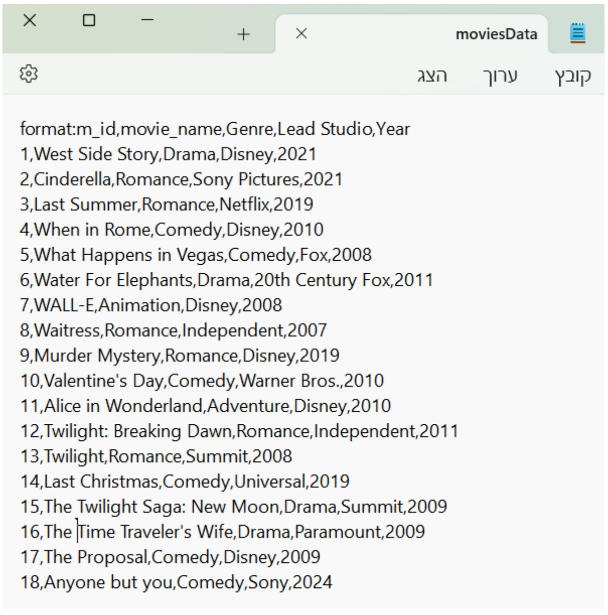
```
C:\Users\ronin\source\repos\

Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 20
Invalid choice. Please choose again.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 1
Enter movie name: When in Rome
Error - Movie with the name 'When in Rome' already exists.
```

If the input is correct, we will get the following output:



```
C:\Users\ronin\source\repos\

Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 1
Enter movie name: Anyone but you
Enter movie genre: Comedy
Enter studio name: Sony
Enter year of release: 2024
Movie is added
```

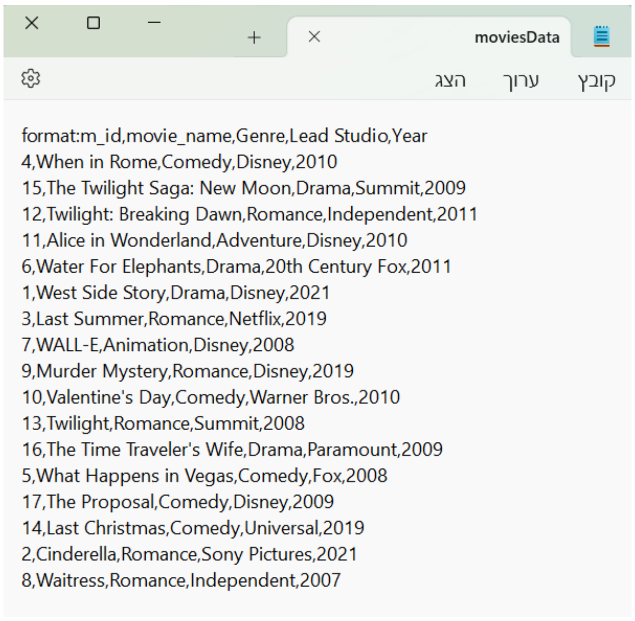
The movies' files before and after the addition:



moviesData

קובץ ערוך הצג

```
format:m_id,movie_name,Genre,Lead Studio,Year
1,West Side Story,Drama,Disney,2021
2,Cinderella,Romance,Sony Pictures,2021
3,Last Summer,Romance,Netflix,2019
4,When in Rome,Comedy,Disney,2010
5,What Happens in Vegas,Comedy,Fox,2008
6,Water For Elephants,Drama,20th Century Fox,2011
7,WALL-E,Animation,Disney,2008
8,Waitress,Romance,Independent,2007
9,Murder Mystery,Romance,Disney,2019
10,Valentine's Day,Comedy,Warner Bros.,2010
11,Alice in Wonderland,Adventure,Disney,2010
12,Twilight: Breaking Dawn,Romance,Independent,2011
13,Twilight,Romance,Summit,2008
14,Last Christmas,Comedy,Universal,2019
15,The Twilight Saga: New Moon,Drama,Summit,2009
16,The Time Traveler's Wife,Drama,Paramount,2009
17,The Proposal,Comedy,Disney,2009
18,Anyone but you,Comedy,Sony,2024
```



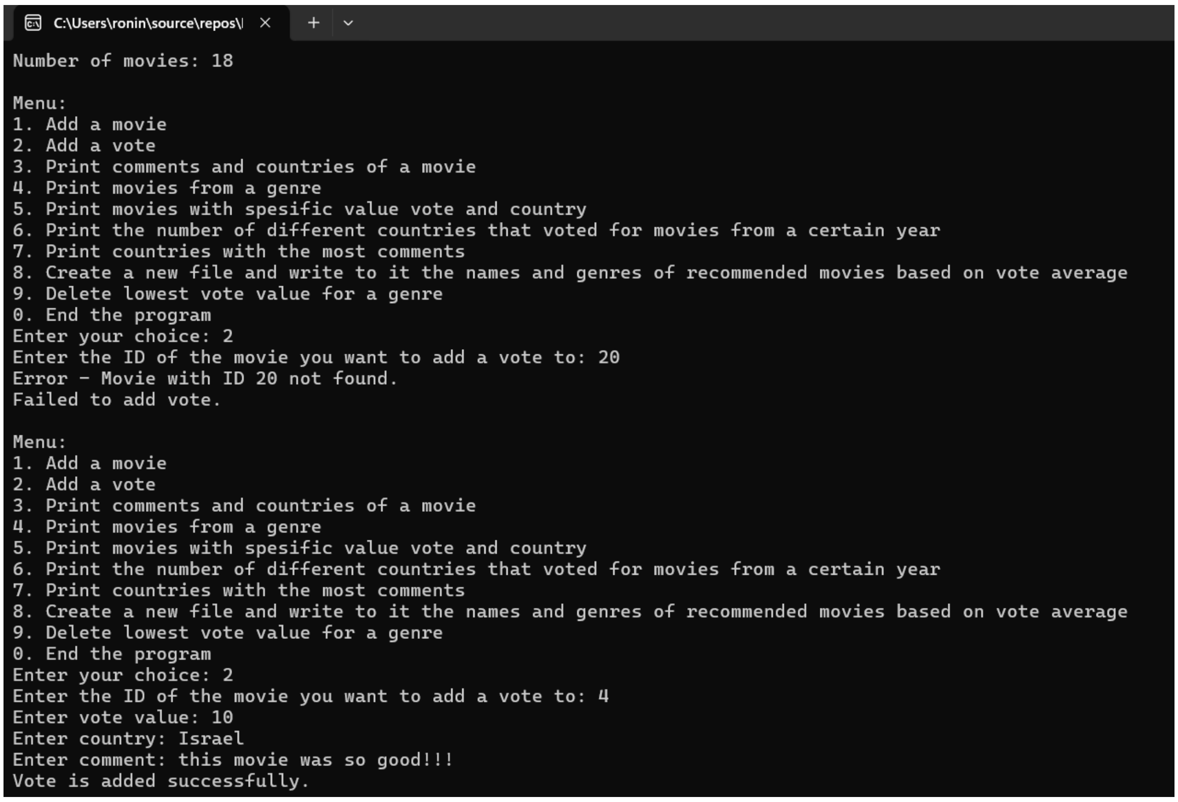
moviesData

קובץ ערוך הצג

```
format:m_id,movie_name,Genre,Lead Studio,Year
4,When in Rome,Comedy,Disney,2010
15,The Twilight Saga: New Moon,Drama,Summit,2009
12,Twilight: Breaking Dawn,Romance,Independent,2011
11,Alice in Wonderland,Adventure,Disney,2010
6,Water For Elephants,Drama,20th Century Fox,2011
1,West Side Story,Drama,Disney,2021
3,Last Summer,Romance,Netflix,2019
7,WALL-E,Animation,Disney,2008
9,Murder Mystery,Romance,Disney,2019
10,Valentine's Day,Comedy,Warner Bros.,2010
13,Twilight,Romance,Summit,2008
16,The Time Traveler's Wife,Drama,Paramount,2009
5,What Happens in Vegas,Comedy,Fox,2008
17,The Proposal,Comedy,Disney,2009
14,Last Christmas,Comedy,Universal,2019
2,Cinderella,Romance,Sony Pictures,2021
8,Waitress,Romance,Independent,2007
```

Choice "2" - addVote:

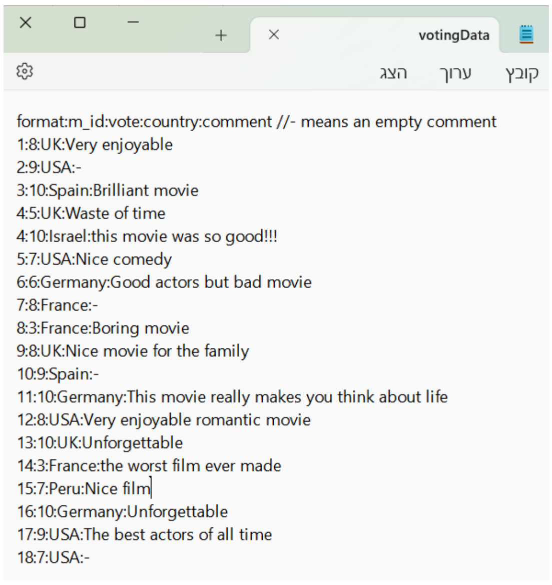
In the top case - an input of a movie that does not exist in the system and an appropriate message, in the bottom case - an existing movie input + an appropriate message:



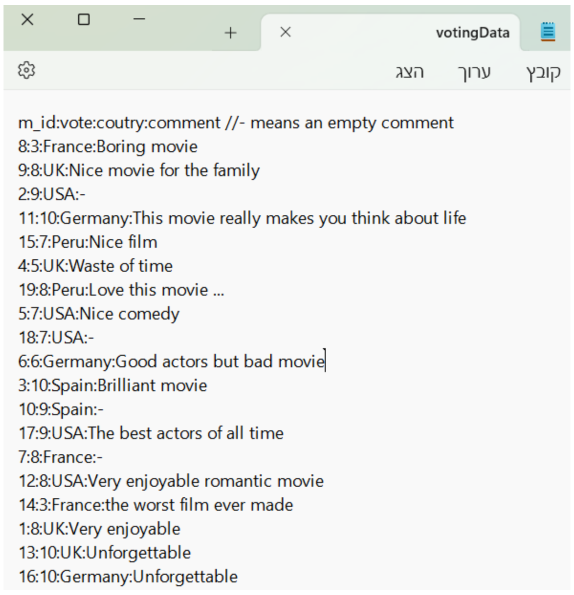
C:\Users\ronin\source\repos\

```
Number of movies: 18

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 2
Enter the ID of the movie you want to add a vote to: 20
Error - Movie with ID 20 not found.
Failed to add vote.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 2
Enter the ID of the movie you want to add a vote to: 4
Enter vote value: 10
Enter country: Israel
Enter comment: this movie was so good!!!
Vote is added successfully.
```

List of reviews before and after the addition:



```
format:m_id:vote:country:comment //- means an empty comment
1:8:UK:Very enjoyable
2:9:USA:-
3:10:Spain:Brilliant movie
4:5:UK:Waste of time
4:10:Israel:this movie was so good!!!
5:7:USA:Nice comedy
6:6:Germany:Good actors but bad movie
7:8:France:-
8:3:France:Boring movie
9:8:UK:Nice movie for the family
10:9:Spain:-
11:10:Germany:This movie really makes you think about life
12:8:USA:Very enjoyable romantic movie
13:10:UK:Unforgettable
14:3:France:the worst film ever made
15:7:Peru:Nice film
16:10:Germany:Unforgettable
17:9:USA:The best actors of all time
18:7:USA:-
```

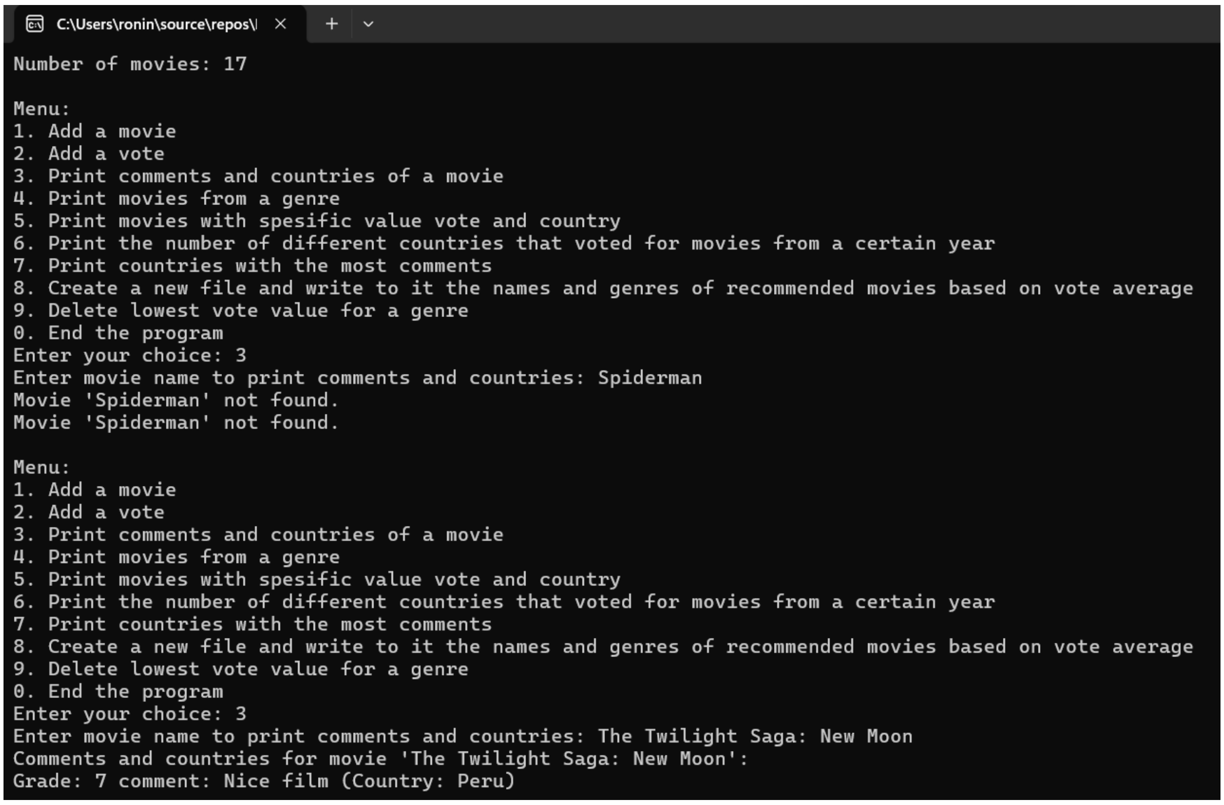


```
m_id:vote:coutry:comment //- means an empty comment
8:3:France:Boring movie
9:8:UK:Nice movie for the family
2:9:USA:-
11:10:Germany:This movie really makes you think about life
15:7:Peru:Nice film
4:5:UK:Waste of time
19:8:Peru:Love this movie ...
5:7:USA:Nice comedy
18:7:USA:-
6:6:Germany:Good actors but bad movie
3:10:Spain:Brilliant movie
10:9:Spain:-
17:9:USA:The best actors of all time
7:8:France:-
12:8:USA:Very enjoyable romantic movie
14:3:France:the worst film ever made
1:8:UK:Very enjoyable
13:10:UK:Unforgettable
16:10:Germany:Unforgettable
```

****Note: In the original review file there is a review of movie number 19 that does not exist in the system, therefore, in the new file, the review will not appear because it is meaningless - as there is no such film in the initial collection of films.**

Choice "3" - printVotes:

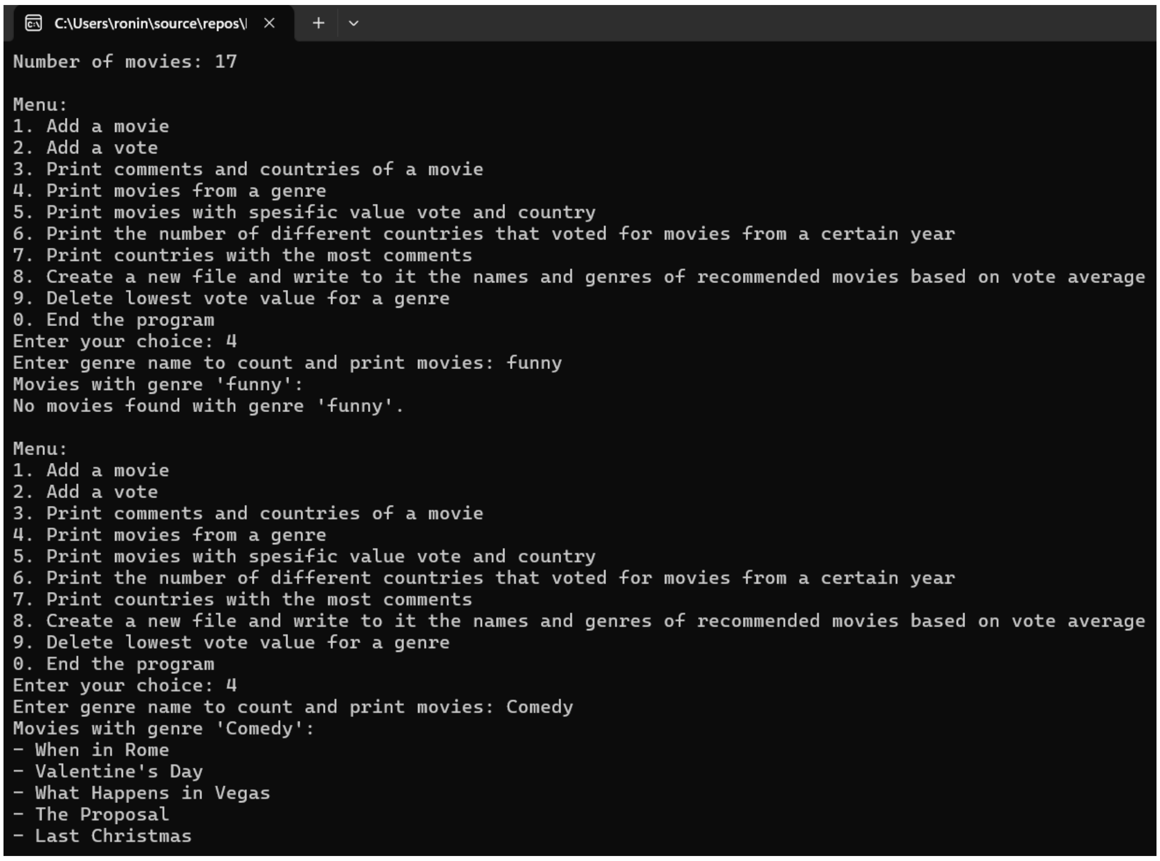
Top Case - incorrect input feed, Bottom Case - correct input feed:



```
Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 3
Enter movie name to print comments and countries: Spiderman
Movie 'Spiderman' not found.
Movie 'Spiderman' not found.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 3
Enter movie name to print comments and countries: The Twilight Saga: New Moon
Comments and countries for movie 'The Twilight Saga: New Moon':
Grade: 7 comment: Nice film (Country: Peru)
```

Choice "4" - countGenre:

Top Case - incorrect input feed, Bottom Case - correct input feed:

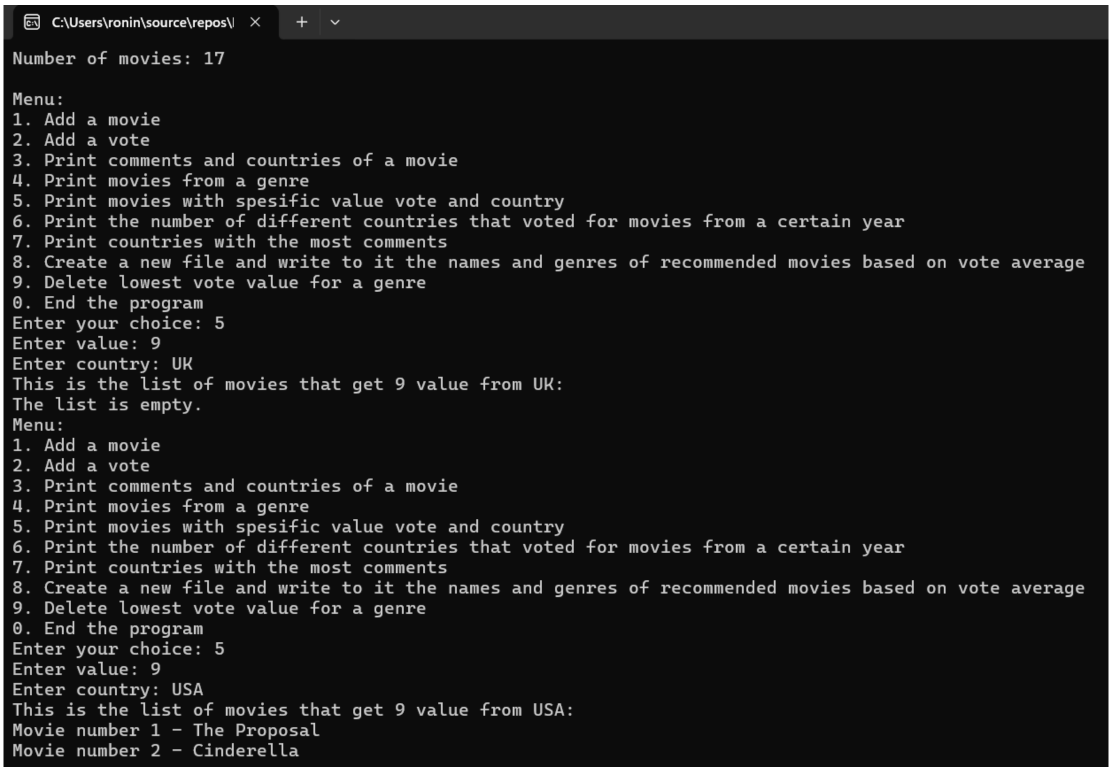


```
Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 4
Enter genre name to count and print movies: funny
Movies with genre 'funny':
No movies found with genre 'funny'.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 4
Enter genre name to count and print movies: Comedy
Movies with genre 'Comedy':
- When in Rome
- Valentine's Day
- What Happens in Vegas
- The Proposal
- Last Christmas
```

Choice "5" - printValue:

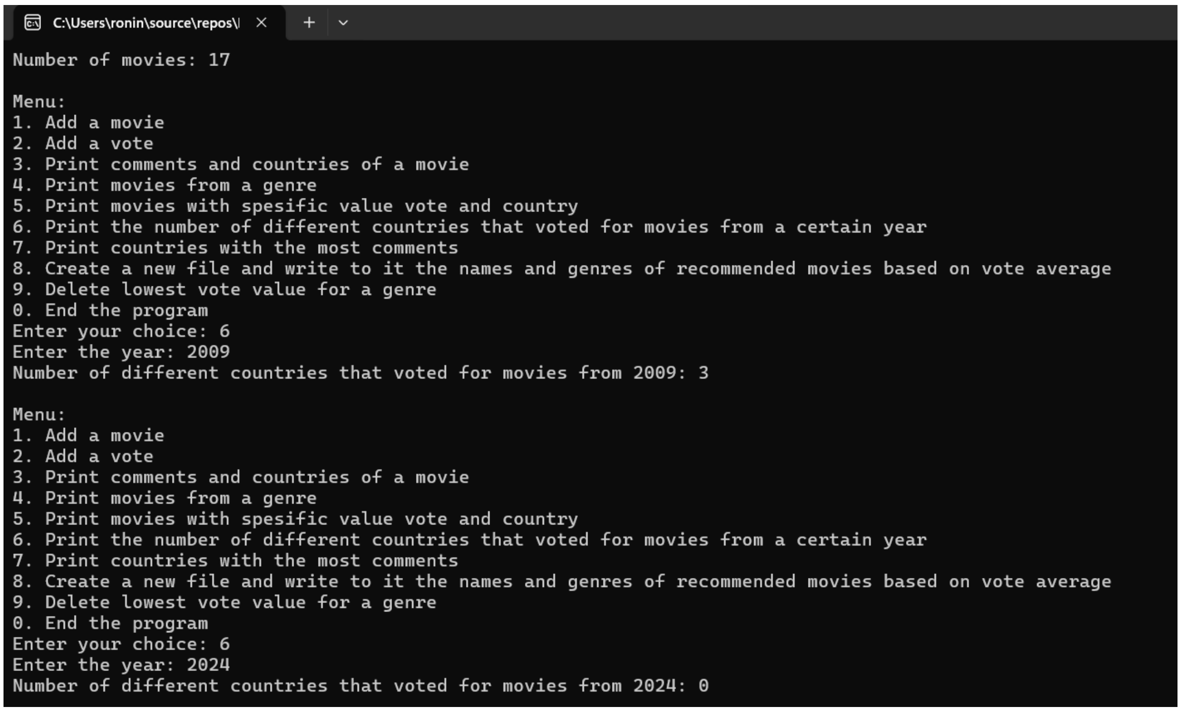
Top case - input feed that doesn't give appropriate values + appropriate output message,

Bottom Case - proper input feed:



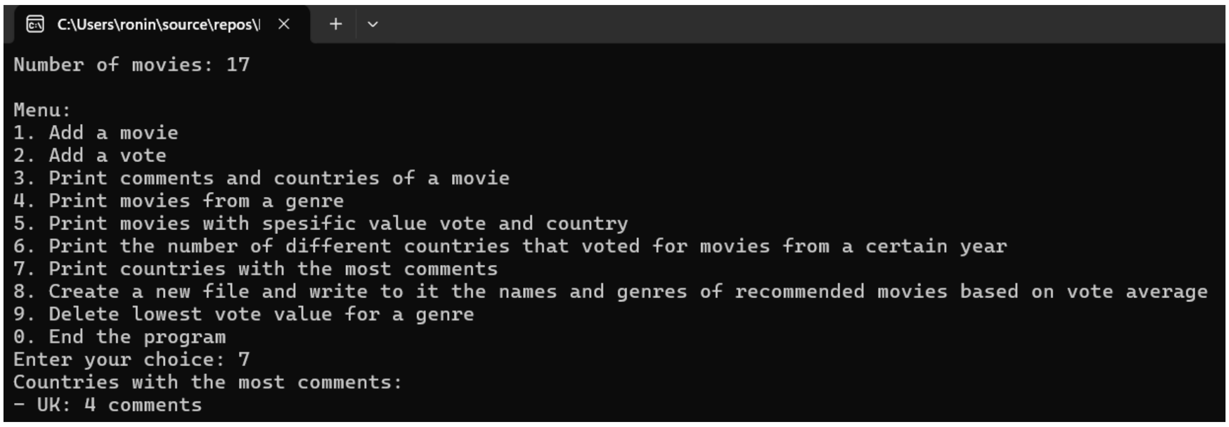
```
Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 5
Enter value: 9
Enter country: UK
This is the list of movies that get 9 value from UK:
The list is empty.
Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 5
Enter value: 9
Enter country: USA
This is the list of movies that get 9 value from USA:
Movie number 1 - The Proposal
Movie number 2 - Cinderella
```

Choice "6" - countCountry:



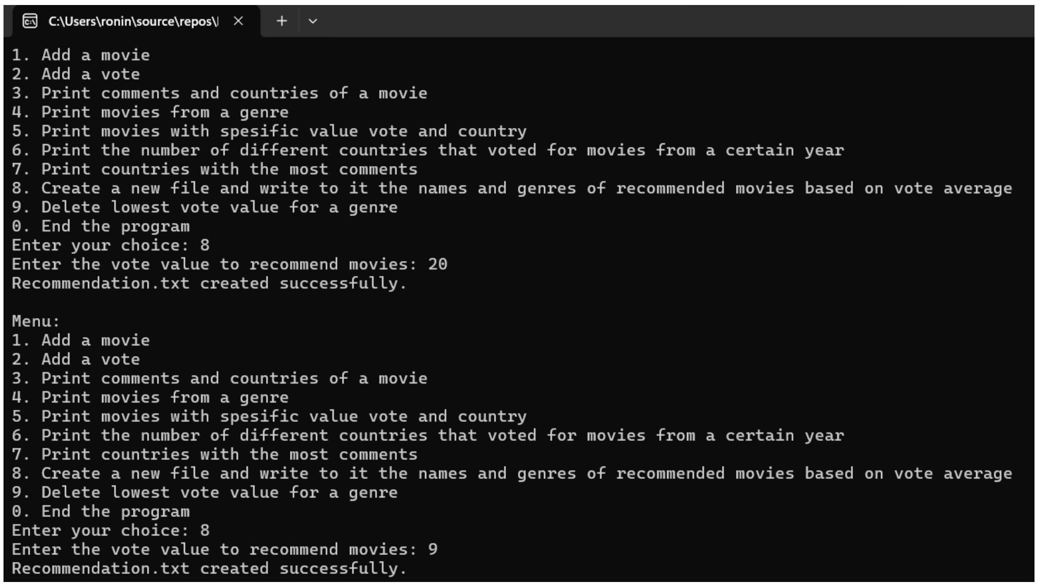
```
C:\Users\ronin\source\repos\

Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 6
Enter the year: 2009
Number of different countries that voted for movies from 2009: 3

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 6
Enter the year: 2024
Number of different countries that voted for movies from 2024: 0
```

Choice "7" - maxByCountry:

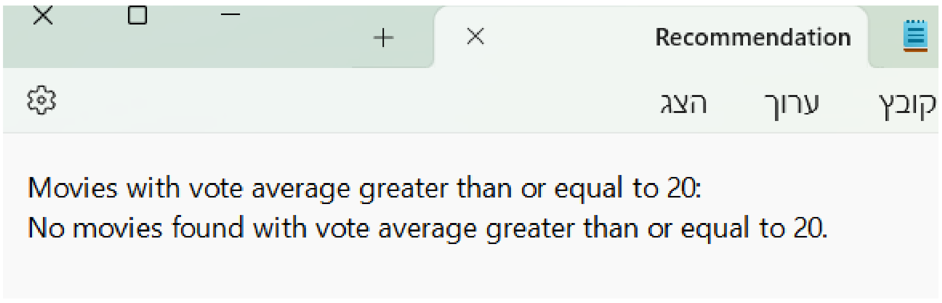


```
C:\Users\ronin\source\repos\

Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 7
Countries with the most comments:
- UK: 4 comments
```

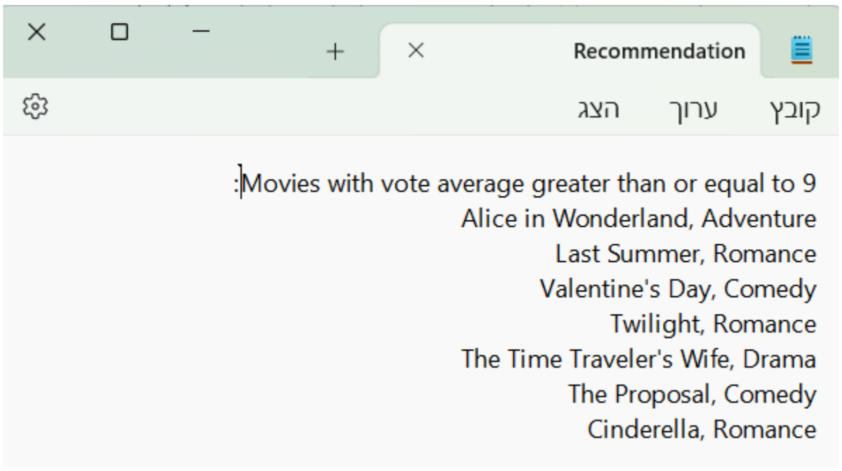
Choice "8" - RecommendMovie:



```
C:\Users\ronin\source\repos\

1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 8
Enter the vote value to recommend movies: 20
Recommendation.txt created successfully.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 8
Enter the vote value to recommend movies: 9
Recommendation.txt created successfully.
```



```
Movies with vote average greater than or equal to 20:
No movies found with vote average greater than or equal to 20.
```

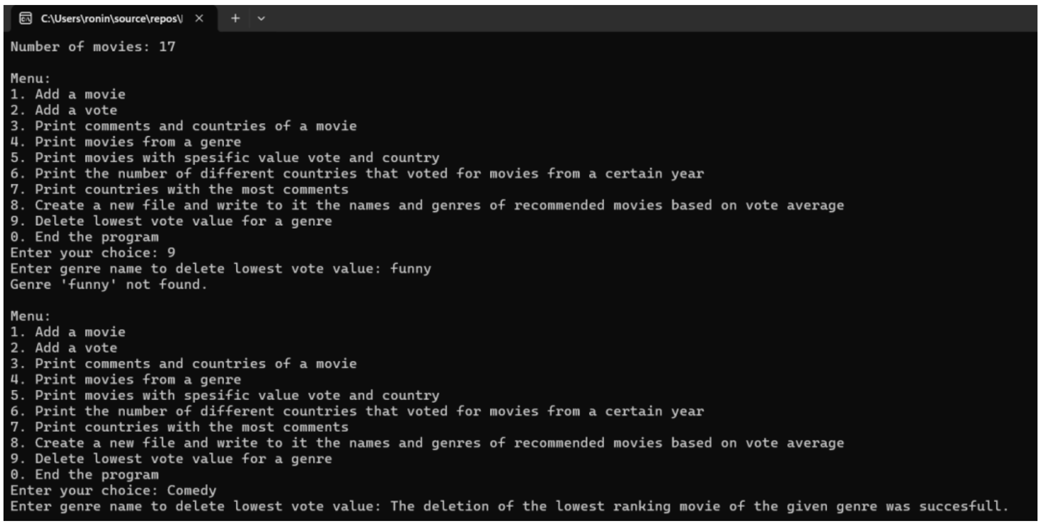


```
:Movies with vote average greater than or equal to 9
Alice in Wonderland, Adventure
Last Summer, Romance
Valentine's Day, Comedy
Twilight, Romance
The Time Traveler's Wife, Drama
The Proposal, Comedy
Cinderella, Romance
```

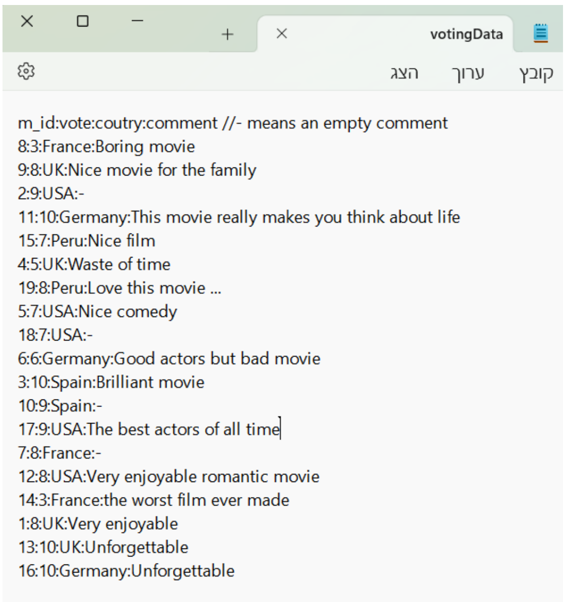
Choice "9" - deleteWorst:

Top case - input feed that doesn't give appropriate values + appropriate output message,

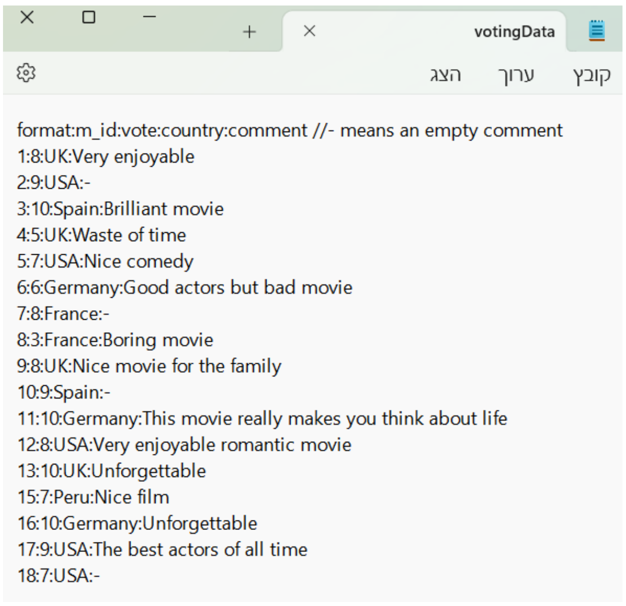
Bottom Case - proper input feed:



```
Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 9
Enter genre name to delete lowest vote value: funny
Genre 'funny' not found.

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: Comedy
Enter genre name to delete lowest vote value: The deletion of the lowest ranking movie of the given genre was succesfull.
```

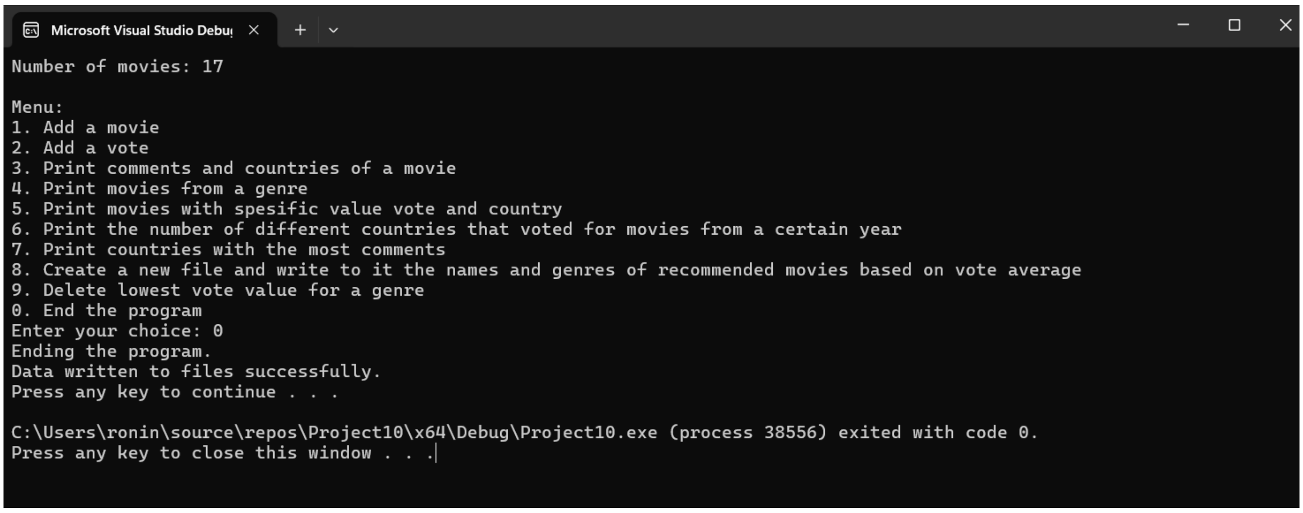


```
m_id:vote:coutry:comment //- means an empty comment
8:3:France:Boring movie
9:8:UK:Nice movie for the family
2:9:USA:-
11:10:Germany:This movie really makes you think about life
15:7:Peru:Nice film
4:5:UK:Waste of time
19:8:Peru:Love this movie ...
5:7:USA:Nice comedy
18:7:USA:-
6:6:Germany:Good actors but bad movie
3:10:Spain:Brilliant movie
10:9:Spain:-
17:9:USA:The best actors of all time
7:8:France:-
12:8:USA:Very enjoyable romantic movie
14:3:France:the worst film ever made
1:8:UK:Very enjoyable
13:10:UK:Unforgettable
16:10:Germany:Unforgettable
```



```
format:m_id:vote:country:comment //- means an empty comment
1:8:UK:Very enjoyable
2:9:USA:-
3:10:Spain:Brilliant movie
4:5:UK:Waste of time
5:7:USA:Nice comedy
6:6:Germany:Good actors but bad movie
7:8:France:-
8:3:France:Boring movie
9:8:UK:Nice movie for the family
10:9:Spain:-
11:10:Germany:This movie really makes you think about life
12:8:USA:Very enjoyable romantic movie
13:10:UK:Unforgettable
15:7:Peru:Nice film
16:10:Germany:Unforgettable
17:9:USA:The best actors of all time
18:7:USA:-
```

Choice "0" - Exit the menu:



```
Number of movies: 17

Menu:
1. Add a movie
2. Add a vote
3. Print comments and countries of a movie
4. Print movies from a genre
5. Print movies with spesific value vote and country
6. Print the number of different countries that voted for movies from a certain year
7. Print countries with the most comments
8. Create a new file and write to it the names and genres of recommended movies based on vote average
9. Delete lowest vote value for a genre
0. End the program
Enter your choice: 0
Ending the program.
Data written to files successfully.
Press any key to continue . . .

C:\Users\ronin\source\repos\Project10\x64\Debug\Project10.exe (process 38556) exited with code 0.
Press any key to close this window . . .
```