CSE102 – Computer Programming (Spring 2021)

Homework #11

Handed out: 7:00pm May 24, 2021.

Due: 11:55pm June 5, 2021.

Hand-in Policy: Via Moodle. No late submissions will be accepted. A student with the number 20180000001 should hand in 20180000001.c for this homework.
Collaboration Policy: No collaboration is permitted.
Grading: This homework will be graded on a scale of 100.

Write a complete program achieving all the tasks below.

- The program takes a single argument as the name of the file to be read. A sample file is attached (Movies.txt). The program reads this file which has labeled columns separated by commas. Every row of this text file includes information about a movie. You are asked to read every row as a string and decompose the given information (separated by commas).
- The following shows a few entries of this file in table format.

budget	genre	name	score	year
8000000	Adventure	Stand by Me	8.1	1986
6000000	Comedy	Ferris Bueller's Day Off	7.8	1986
15000000	Action	Top Gun	6.9	1986
18500000	Action	Aliens	8.4	1986
9000000	Adventure	Flight of the Navigator	6.9	1986
6000000	Drama	Platoon	8.1	1986
25000000	Adventure	Labyrinth	7.4	1986
6000000	Drama	Blue Velvet	7.8	1986
9000000	Comedy	Pretty in Pink	6.8	1986
15000000	Drama	The Fly	7.5	1986
8800000	Adventure	Crocodile Dundee	6.5	1986
16000000	Action	Highlander	7.2	1986
6000000	Comedy	Lucas	6.8	1986
25000000	Action	Big Trouble in Little China	7.3	1986

Figure.1 Movie List Example

- The program should keep the movies in two distinct linked lists whose nodes are named as Movie_Budget and Movie_Name. In Movie_Budget, there should be three components; budget and year are kept as integers, and the name is stored as strings, in Movie_Name there should be three fields; genre and name as string and score as double. In the Movie_Budget list, elements are inserted and kept in descending order by year. If years are the same, it should keep elements according to their budget in descending order. If these two values are the same, you can choose an arbitrary order among them. There should be exactly the same number of elements in these lists as the number of lines in the file.
    - Note that the file may have multiple lines for the same movie. You should check this with the movie name. If the name is already read and stored in the list, the data should be updated with the new data.

- The program should keep the movie names and genres (can be of arbitrary length) in dynamically allocated strings.
- The number of movies in the Movies.txt file is unknown.
- Some of the movies have no value for the 'budget' column. In this case, the string ‘unknown’ is used.

Part 1: The program should support 8 different operations. Therefore, there should be a menu like the one below. The program should be terminated if and only if the user enters "8" as input. If an entry is invalid or the previously requested action has been taken, the menu should appear repeatedly. If the menu appears due to an invalid entry, there should also be an error message.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation:
```

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 9

TRY AGAIN.


1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation:
```

Part 2: If the user enters '1', the program should return the specified number of movie records, sorted by the selected area. As stated in the first section, if the user enters a wrong value in these selection steps, you should be directed to a new one in order to enter the correct one. If the user selects a single selection, you must return a single record. If the user selects more than one selection, you must fetch all records in the specified range. For only in multiple selections, for year or budget or score, the program should take exact year or budget or score values from the user as ranges to get related records. All rankings must be made in ascending order. Don’t forget! In the first selection, you will choose according to which field it will be sorted.

Part 3: 2nd operation lists the movie genres/types as strings, if the user enters "2" as input, the program should list the genres/types as strings.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 2

adventure
action
drama
comedy
crime
horror
biography
musical
animation
```

Part 4: 3rd operation lists the movies based on the information received from the user. The program asks for a "year" first, after that program asks "until that year or since that year?". Finally, the program lists the names of all movies that have been aired until / since that year. Note that the program must handle an invalid year or period values. So, if the oldest movie was released in 2000 and the user wants to print the movie that was released until 2001, the program should display an error message and request new values as there were no movies before 2001.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 3

Enter a year:
2001
Until (0) or Since (1)  2001:0

Juego de espÿas
Baby Boy
Kate & Leopold
Heartbreakers
Ghosts of Mars
Don't Say a Word
```

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 3

Enter a year:
2000
Until (0) or Since (1)  2000:0

TRY AGAIN

Enter a year:
```

Part 5: 4th operation lists the movies according to information received from the user. Firstly, the program asks for a 'score', after that the program asks 'less or greater than that score?'. Finally, the program lists the names of all movies whose score is less / greater than that score. Don't forget that the program should handle the invalid score or comparison values. So, if there is no movie with a score below 4.7 and the user wants to print movies with a score below 4.7, the program should display an error message and ask for new values because there are no movies scored below 4.7.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 4

Enter a score:
5.2
Less (0) or Greater (1)  5.2:0

One Crazy Summer
Trick or Treat
Gung Ho
The Morning After
Police Academy 3: Back in Training
She's Gotta Have It
Ruthless People
Band of the Hand
```

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 4

Enter a score:
4.7
Less (0) or Greater (1)  4.7:0

TRY AGAIN

Enter a score:
```

Part 6: 5th operation asks for a movie name from the user. If the input matches with a name from the list, then the program prints all information of that movie. If the input doesn't match with any element of the list, then it should give an error message and ask for new input. If the 'budget' information of that movie was 'unknown' when the program read it, then the users should see it as 'Unknown'.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 5

Please Enter the name of the movie:
critters
Budget: 2000000
Genre:  Action
Name:   Critters
Score:  6.0
Year:   1986
```

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 5

Please Enter the name of the movie:
matador
Budget: Unknown
Genre:  Drama
Name:   Matador
Score:  5.1
Year:   1986
```

Part 7: 6th operation basically calculates and prints the average of the IMDB scores of all movies. This value is for example purposes only. The actual value of the file given to you will be different.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 6

Average: 5.423974
```

Part 8: 7th operation printing the frequency of every genre. I.e. print the number of movies for every genre. These values are for example purposes only. The numbers on the file given to you will be different.

```
1. List of the Sorted Data
2. List of the Genres
3. List of the Movie Through the Years
4. List of the Movie Through the Scores
5. All Informations of a Single Movie
6. Average of the IMDB Scores
7. Frequence of the Genres
8. Exit

Please Select an operation: 7

adventure       127
action          346
drama           468
comedy          704
crime           139
horror          269
biography       78
musical         154
animation       651
```

❖ You are not allowed to use any library other than;

- ➢ <stdio.h>
- ➢ <string.h>
- ➢ <stdlib.h>

❖ You can use the ‘strtod’ function from <stdlib.h> library to parse a string to double.
❖ You cannot use **realloc()** for dynamic allocation operations. You can only use **malloc()** and **calloc().**
❖ If there is more than one match while performing any search, select the first one.
❖ You can write your own functions to make things easier.
❖ Don’t forget that the program shouldn’t terminate when an operation is done.

General Rules:

1. Obey and do not break the function prototypes that are shown on each part, otherwise, you will get zero from the related part.
2. The program must be developed on Linux-based OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.
3. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it’s working in some way.
4. You can ask any question about the homework by using the forum on the Teams page of the course.