WATCH OR NOT – ENTERTAINMENT RECOMMENDATION SYSTEM

C Programming Project

Submitted By:
Jashith Sharma
Sap ID: 590024037
Batch : 45
Semester: 1

Under the Guidance of:
Ms. Dolly Das

ABSTRACT

The aim of this project is to build a simple and interactive entertainment recommendation system using the C programming language. The idea behind the system is to help users decide what to watch when they are confused between endless movies and series available online. The program interacts with the user through a fictional character named Dr. Z, who acts as the librarian of the "Watch or Not" library.

The program asks the user whether they want to watch a movie or a series and then allows them to either pick a specific genre or get a random suggestion. For specific choices, the system provides separate Bollywood and Hollywood recommendations along with IMDb ratings and Netflix secret genre codes. In random mode, the program gives unique suggestions without repeating until the entire list has been shown.

The project makes use of concepts taught in class such as structures, arrays, loops, functions, string handling, and random number generation. The typing-effect output is added to make the interaction more engaging. Overall, the project demonstrates how basic C programming concepts can be combined to create a small but useful real-world application.

PROBLEM DEFINITION

With so many movies and web series available across different platforms, people often struggle to decide what to watch. The amount of content can become overwhelming, and users usually spend more time searching than actually watching anything. The confusion becomes even bigger when they have to choose between different genres, languages, and industries like Bollywood and Hollywood.

The problem is to create a simple system that can guide a user toward a movie or series based on what they feel like watching. The system should be easy to use, require minimum input, and provide quick recommendations. It should also allow both specific choices (like action or romance) and random suggestions for users who are undecided.

To solve this, we use a C-based program that interacts with the user through text, collects their preferences, and displays curated suggestions from predefined categories. This reduces decision fatigue and helps users reach a final choice faster.

OBJECTIVES

The main objectives of this project are:

1. **To create an easy-to-use entertainment recommendation system** using basic C programming concepts.
2. **To help users decide what to watch** by providing movie or series suggestions in a simple and interactive way.
3. **To support both specific and random modes,** users can either choose a genre or get surprised recommendations.
4. **To store Bollywood and Hollywood options separately**, giving the user a wider variety of suggestions.
5. **To make use of key C concepts** such as:
    a. Arrays
    b. Structures
    c. Loops

    d. Conditional statements
    e. Functions
    f. String handling
    g. Random number generation
6. **To enhance user experience** by adding a typing animation effect and humorous responses through the character "Dr. Z".
7. **To build a program that is simple, modular, and extendable**, so more genres or features can be added easily in the future.

SYSTEM REQUIREMENTS

1. Software Requirements

- **Programming Language:** C
- **Compiler:** GCC / (any standard C compiler)
- **Operating System:** macOS, Windows, or Linux
- **Editor / IDE:**
    - Visual Studio Code
    - Mac terminal
    - or any text editor that supports C

2. Libraries Used

The program uses the following C standard libraries:

- stdio.h – for input/output operations
- string.h – for string comparison, copying, and manipulation

- unistd.h – for the typing animation delay using usleep() (on macOS/Linux)
- stdlib.h – for random number functions like rand() and srand()
- time.h – for seeding the random generator using time(NULL)

SYSTEM DESIGN

The system is designed in a modular way so that each part of the program has a clear purpose. The entire flow is divided into small logical steps that together form the complete entertainment recommendation system.

1. User Input Stage

The program first asks the user what they want to watch — a **Movie** or a **Series**.
To avoid invalid choices, the user is given three chances to enter the correct option.

2. Mode Selection

After selecting the type, the user is asked whether they want:

- **Random Mode** – system gives random unique suggestions
- **Specific Mode** – user selects a genre and receives targeted suggestions

This gives the user flexibility and makes the interaction more interesting.

3. Genre Selection (Specific Mode)

If the user chooses specific mode, they can type one of the supported genres such as:
action, romance, drama, thriller, horror, comedy, sci-fi, animated, mystery, crime, fantasy, adventure

The program also normalizes input (e.g. “romantic” → “romance”, “scifi” → “sci-fi”) to handle variations in typing.

4. Data Lookup

The program stores all Bollywood and Hollywood movies/series in two structured arrays using the Item struct.
The system searches these arrays based on:

- Type (movie/series)
- Genre
- Industry (Bollywood/Hollywood)

This organized storage makes searching fast and modular.

5. Output Generation

Once the match is found, the program displays:

- Title
- Industry
- Type
- Genre
- IMDb rating
- Netflix secret codes for that genre

The output is shown with a "typing effect" to make the program feel interactive and alive, presented through the fictional character **Dr. Z**.

6. Random Suggestion Flow

In random mode, the system:

- Picks a random movie or series
- Ensures no suggestion is repeated
- Asks if the user wants another suggestion
- Stops when the list is exhausted

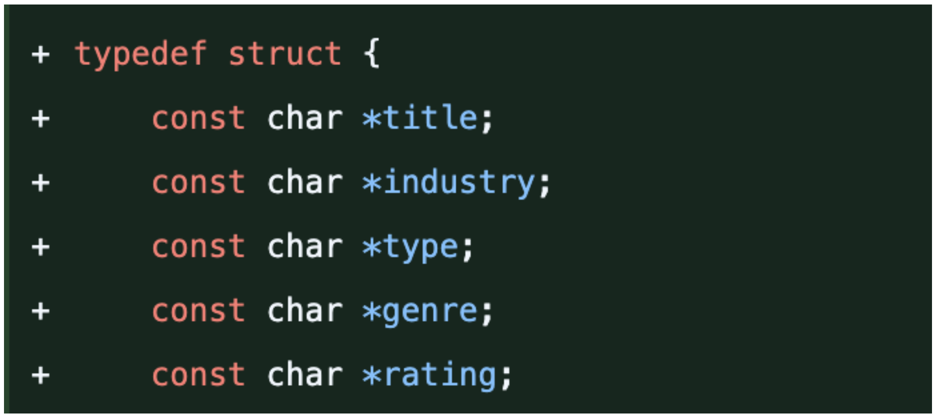
7. Exit & Final Message

At the end, the program politely ends with:
“Visit the library again soon.”

IMPLEMENTATION DETAILS

The entire program is implemented in the C programming language using a modular approach. The main logic is divided into multiple functions so that the code remains clean, easy to understand, and simple to maintain. The following concepts were used in the implementation:

1. Structures (struct Item)



```
+ typedef struct {
+     const char *title;
+     const char *industry;
+     const char *type;
+     const char *genre;
+     const char *rating;
```

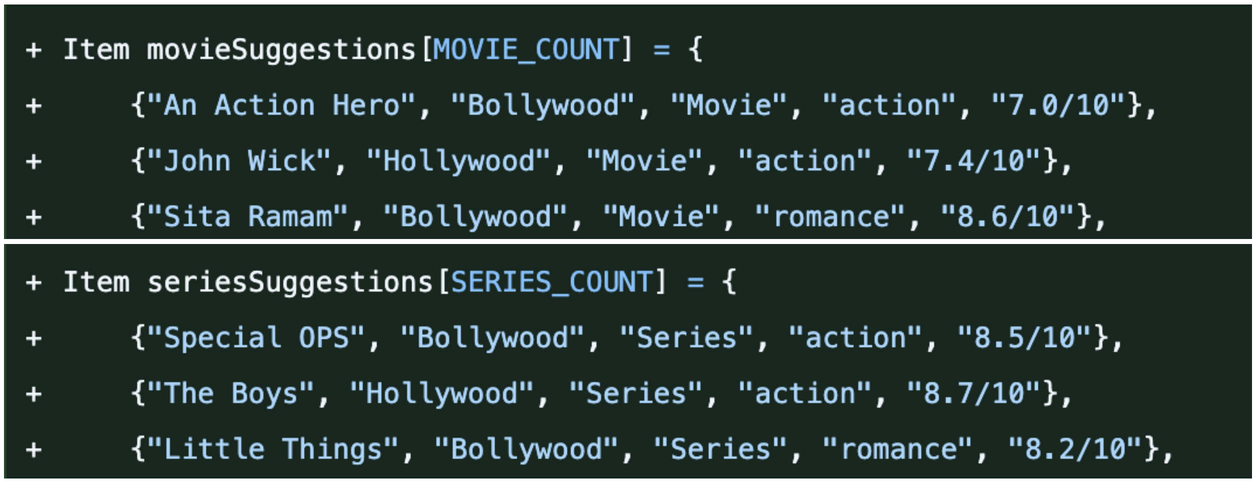
Fig 1: Custom structure used to store movies and series with multiple attributes.

A custom structure named Item is used to store information about each movie or series.
Each entry contains:

- Title
- Industry (Bollywood/Hollywood)
- Type (Movie/Series)
- Genre
- IMDb Rating

Using structures helps group related information together instead of using separate arrays for every field.

2. Arrays



```
+ Item movieSuggestions[MOVIE_COUNT] = {
+     {"An Action Hero", "Bollywood", "Movie", "action", "7.0/10"},
+     {"John Wick", "Hollywood", "Movie", "action", "7.4/10"},
+     {"Sita Ramam", "Bollywood", "Movie", "romance", "8.6/10"},
```

```
+ Item seriesSuggestions[SERIES_COUNT] = {
+     {"Special OPS", "Bollywood", "Series", "action", "8.5/10"},
+     {"The Boys", "Hollywood", "Series", "action", "8.7/10"},
+     {"Little Things", "Bollywood", "Series", "romance", "8.2/10"},
```

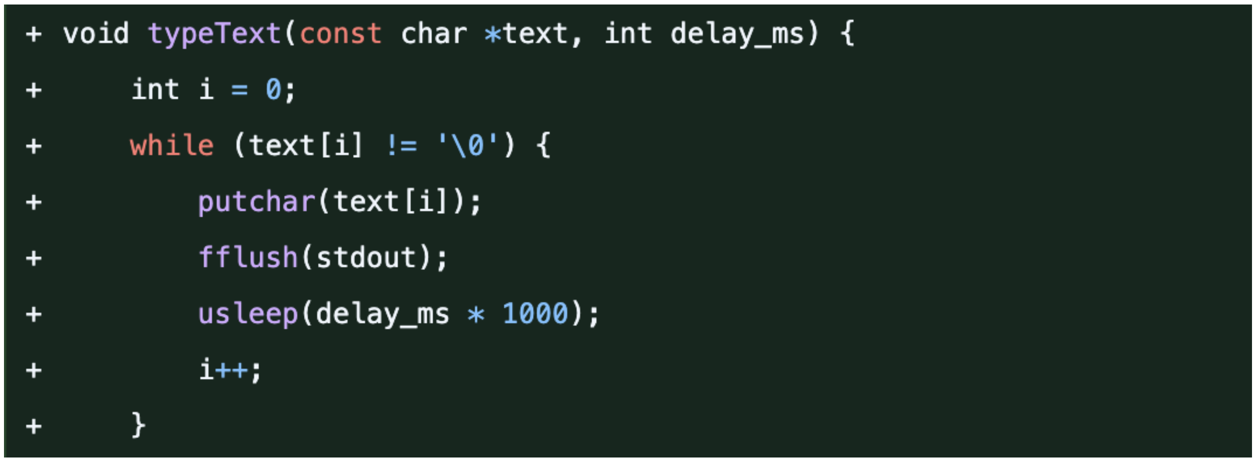
Fig 2: Arrays of structures storing Bollywood and Hollywood data for different genres.

Two arrays of structures are used:

- movieSuggestions[ ]
- seriesSuggestions[ ]

Each array stores multiple Item entries.
This allows easy searching, looping, and random selection.

3. Functions



```
+ void typeText(const char *text, int delay_ms) {
+     int i = 0;
+     while (text[i] != '\0') {
+         putchar(text[i]);
+         fflush(stdout);
+         usleep(delay_ms * 1000);
+         i++;
+     }
```

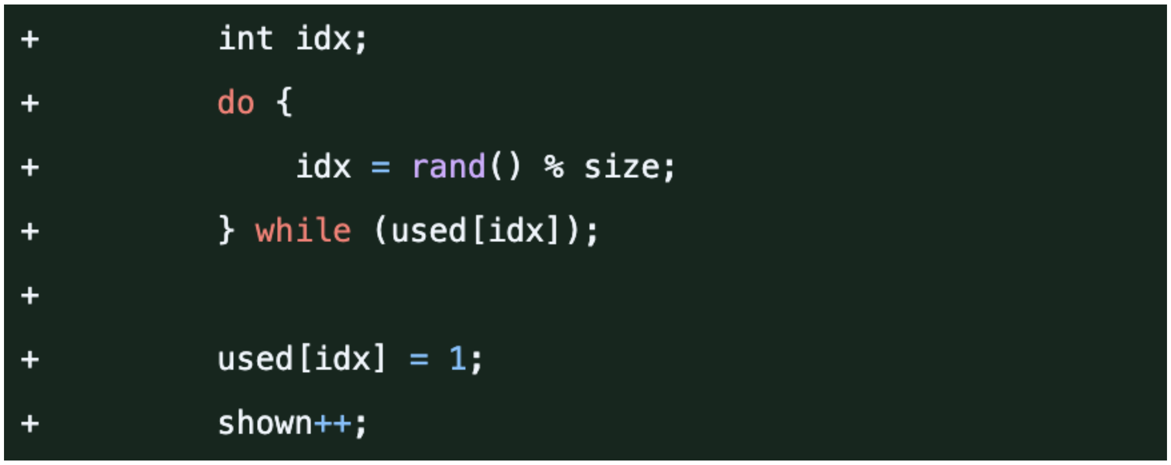
Fig 3: Typing animation function to make output look interactive.

Several functions were written to divide the program logically:

- typeText() → creates a typing animation effect
- toLowerStr() → converts user input to lowercase
- normalizeGenre() → corrects input variations (e.g., “scifi”, “romantic”)
- getGenreCodes() → returns Netflix secret genre codes
- printItemWithCodes() → prints details of a suggestion
- randomSuggestions() → handles random mode
- showSpecificSuggestions() → handles specific genre selection

This modular approach improves code readability and reusability.

4. Random Number Generation



```
+           int idx;
+           do {
+               idx = rand() % size;
+           } while (used[idx]);
+
+           used[idx] = 1;
+           shown++;
```

Fig 4: Logic for generating unique random suggestions without repetition.

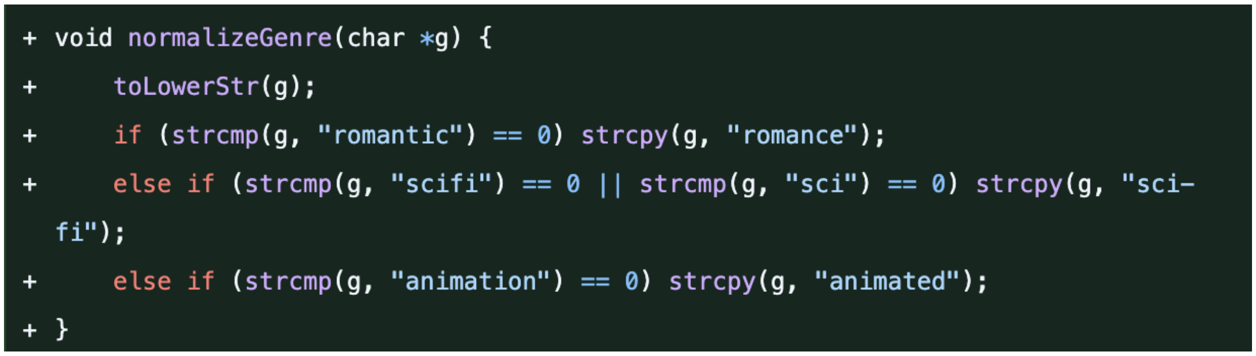
Random suggestions are generated using:

- rand() – generates random numbers

- srand(time(NULL)) – seeds the random generator so output changes each run

A separate array is used to track already-shown items so no movie/series is repeated in random mode.

5. String Handling



```
+ void normalizeGenre(char *g) {
+     toLowerStr(g);
+     if (strcmp(g, "romantic") == 0) strcpy(g, "romance");
+     else if (strcmp(g, "scifi") == 0 || strcmp(g, "sci") == 0) strcpy(g, "sci-
  fi");
+     else if (strcmp(g, "animation") == 0) strcpy(g, "animated");
+ }
```

Fig 5: Genre normalization function to correct spelling variations like “scifi”, “romantic”, etc.

The program uses functions from string.h:

- strcmp() for comparing user input
- strcpy() for normalizing genres
- Arrays of characters (char[]) for storing inputs

String handling is essential for reading user choices like “movie”, “series”, and genres.

6. Conditional Statements

Nested if-else conditions help verify user input, mode selection, and genre selection.
This ensures users enter valid data and receive correct suggestions.

7. Loops

Loops are used for:

- Searching arrays
- Generating multiple random suggestions
- Managing repeated input attempts

8. Typing Animation

usleep() from unistd.h is used to slow down the output slightly, giving the feel of Dr. Z “typing” messages.
This is purely for user experience and makes the program more engaging.

TESTING AND RESULTS

The program was tested with multiple input combinations to ensure that all features work correctly. The tests included valid inputs, invalid inputs, boundary cases, random mode behavior, and genre-based output. Below is a summary of the main test cases and their results.

Test Case 1: Valid Input (Movie + Specific Mode + Romance)

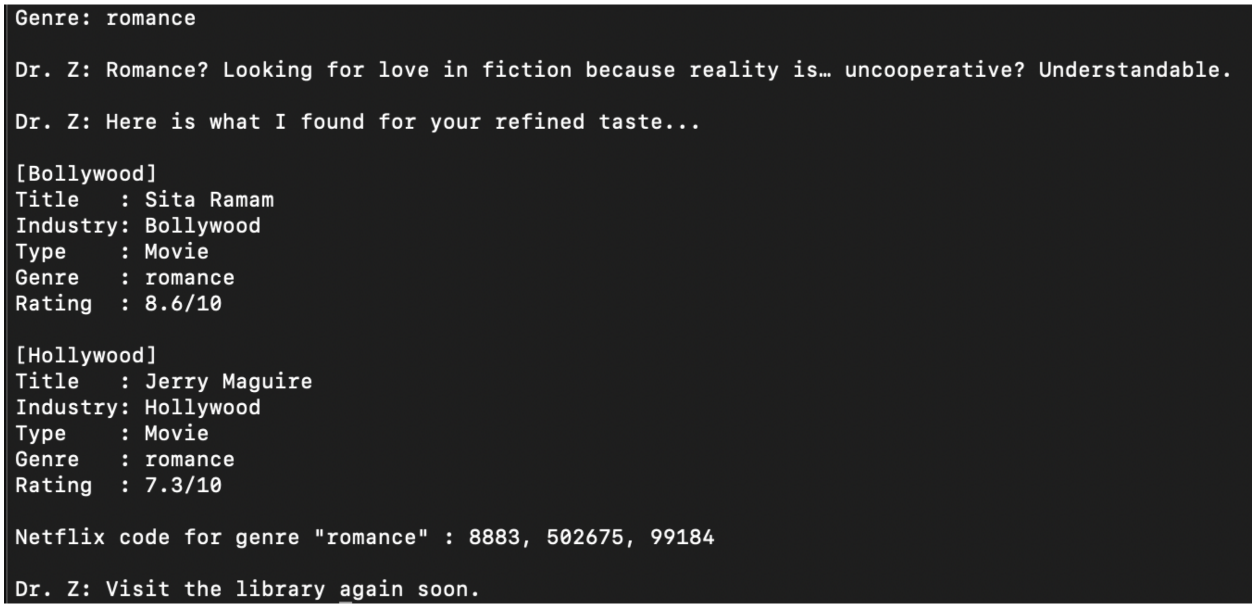
Input:

- Type: movie
- Mode: specific
- Genre: romantic

Expected Output:

- Bollywood Suggestion
- Hollywood Suggestion
- Correct IMDb ratings displayed
- Correct Netflix genre codes displayed

Actual Result:



```
Genre: romance

Dr. Z: Romance? Looking for love in fiction because reality is… uncooperative? Understandable.

Dr. Z: Here is what I found for your refined taste...

[Bollywood]
Title   : Sita Ramam
Industry: Bollywood
Type    : Movie
Genre   : romance
Rating  : 8.6/10

[Hollywood]
Title   : Jerry Maguire
Industry: Hollywood
Type    : Movie
Genre   : romance
Rating  : 7.3/10

Netflix code for genre "romance" : 8883, 502675, 99184

Dr. Z: Visit the library again soon.
```

Status: Passed

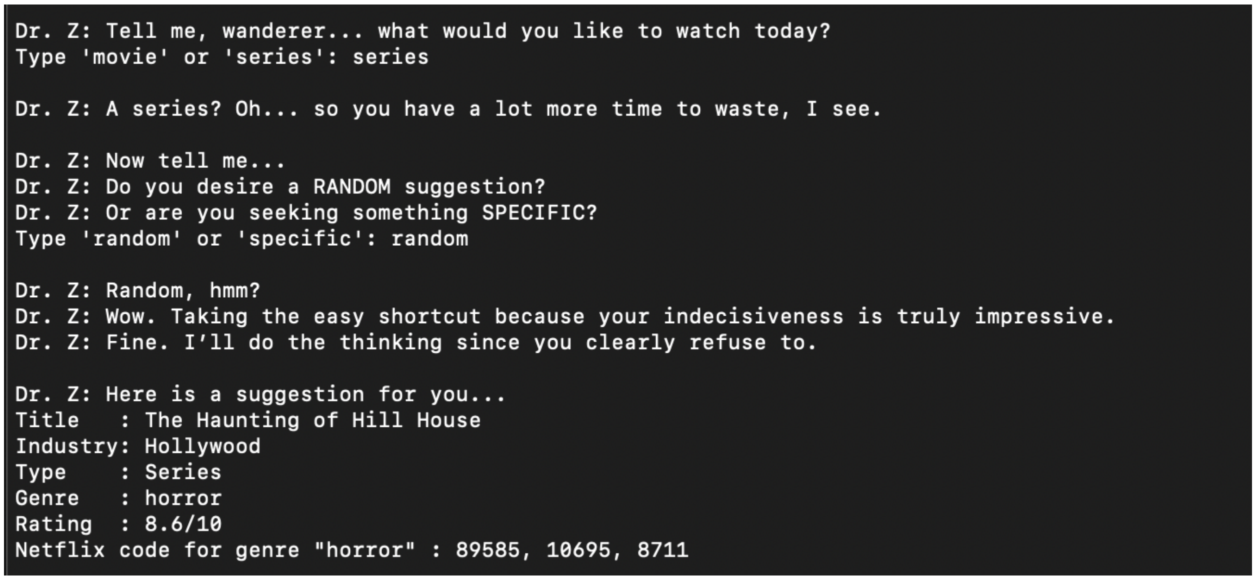
Test Case 2: Valid Input (Series + Random Mode)

Input:

- Type: series
- Mode: random
- User requests multiple suggestions

Expected Output:

- Random unique series suggestions
- No repetition
- Program stops when list is exhausted



```
Dr. Z: Tell me, wanderer... what would you like to watch today?
Type 'movie' or 'series': series

Dr. Z: A series? Oh... so you have a lot more time to waste, I see.

Dr. Z: Now tell me...
Dr. Z: Do you desire a RANDOM suggestion?
Dr. Z: Or are you seeking something SPECIFIC?
Type 'random' or 'specific': random

Dr. Z: Random, hmm?
Dr. Z: Wow. Taking the easy shortcut because your indecisiveness is truly impressive.
Dr. Z: Fine. I'll do the thinking since you clearly refuse to.

Dr. Z: Here is a suggestion for you...
Title   : The Haunting of Hill House
Industry: Hollywood
Type    : Series
Genre   : horror
Rating  : 8.6/10
Netflix code for genre "horror" : 89585, 10695, 8711
```

Actual Result:

- Random suggestions displayed without duplicates
- User able to request multiple suggestions

Status: Passed

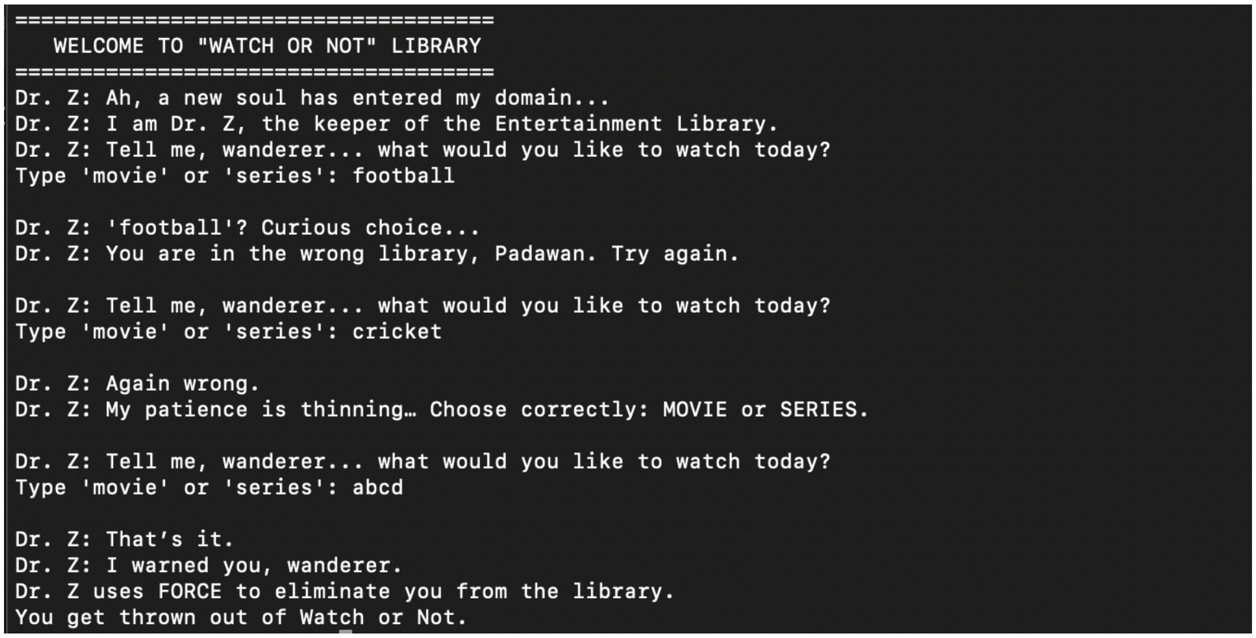
Test Case 3: Invalid Type Input (More than 3 Times)

Input:

- Type: “football”, “cricket”, “abcd”

Expected Output:

- Dr. Z warns the user
- After 3 wrong attempts, program should exit



```
=====================================
   WELCOME TO "WATCH OR NOT" LIBRARY
=====================================
Dr. Z: Ah, a new soul has entered my domain...
Dr. Z: I am Dr. Z, the keeper of the Entertainment Library.
Dr. Z: Tell me, wanderer... what would you like to watch today?
Type 'movie' or 'series': football

Dr. Z: 'football'? Curious choice...
Dr. Z: You are in the wrong library, Padawan. Try again.

Dr. Z: Tell me, wanderer... what would you like to watch today?
Type 'movie' or 'series': cricket

Dr. Z: Again wrong.
Dr. Z: My patience is thinning… Choose correctly: MOVIE or SERIES.

Dr. Z: Tell me, wanderer... what would you like to watch today?
Type 'movie' or 'series': abcd

Dr. Z: That’s it.
Dr. Z: I warned you, wanderer.
Dr. Z uses FORCE to eliminate you from the library.
You get thrown out of Watch or Not.
```

Actual Result:

- Warning messages displayed correctly
- Program exited after third invalid input

Status: Passed

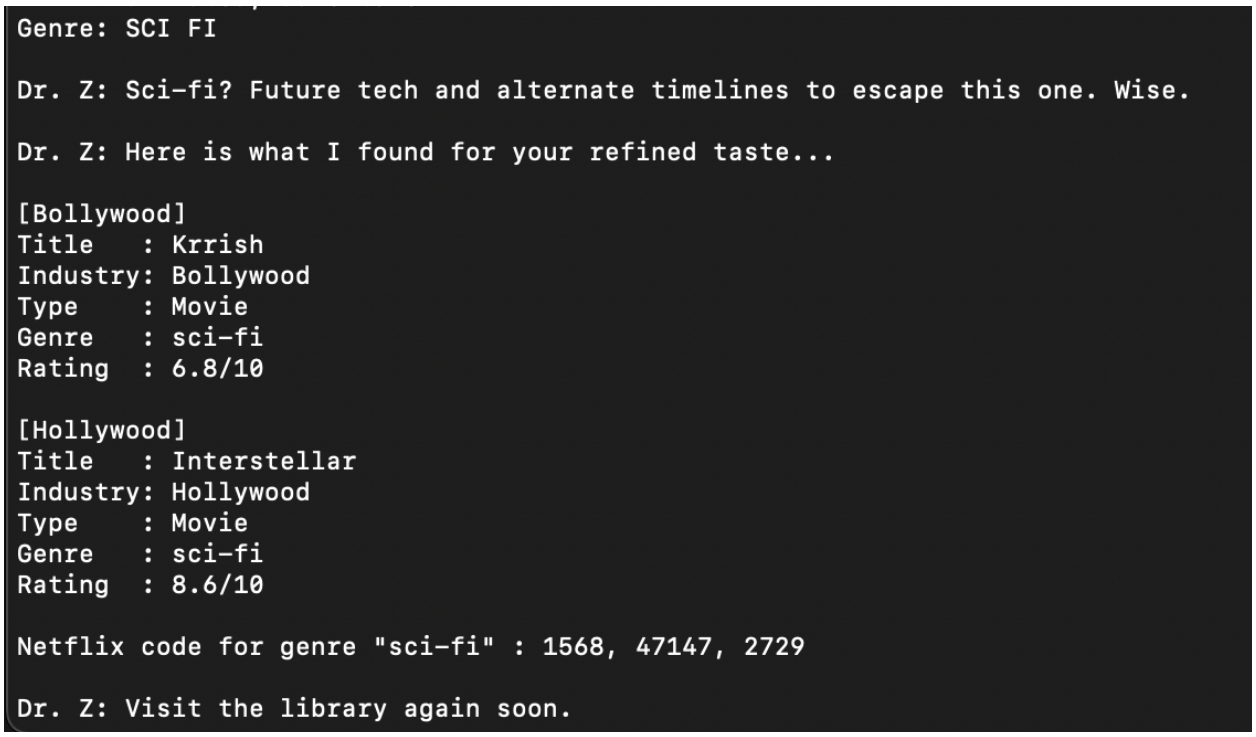
Test Case 4: Genre Normalization

Input Examples:

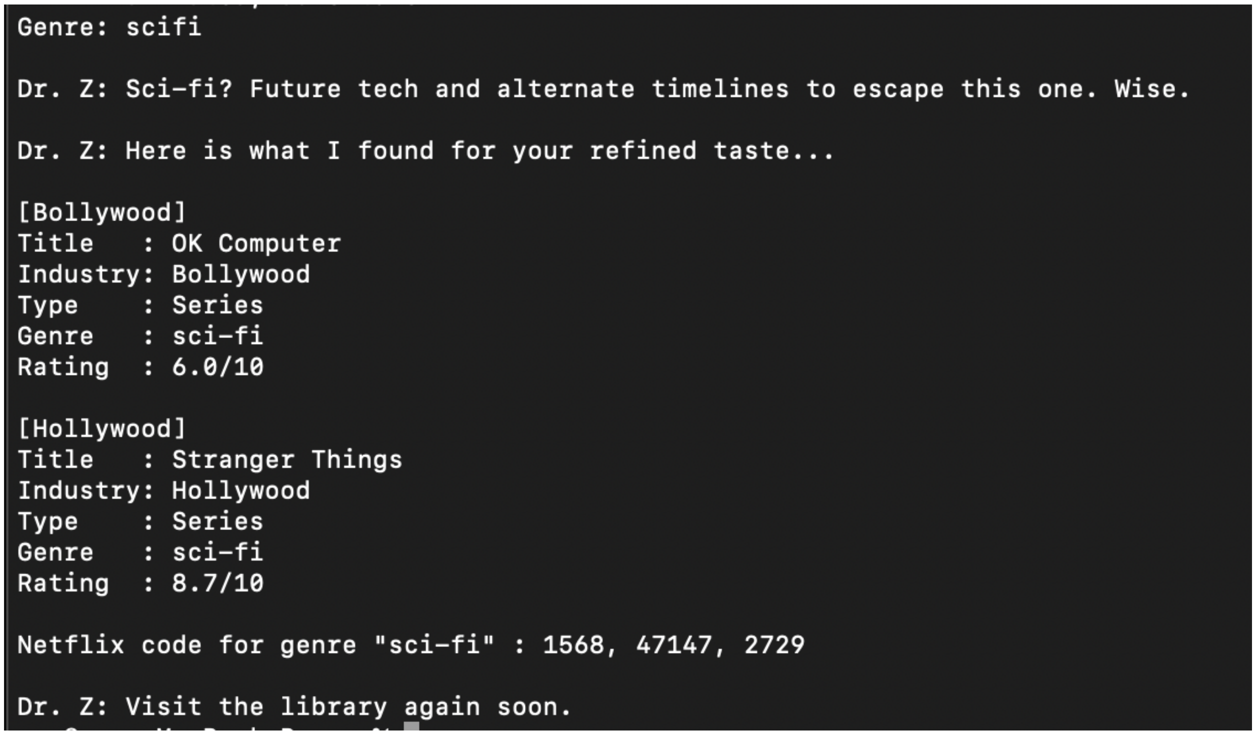
- scifi
- Sci-Fi
- SCI FI
- romantic
- thrillerr

Expected Output:

- All variations should map to valid genres
- Correct suggestions displayed



```
Genre: SCI FI

Dr. Z: Sci-fi? Future tech and alternate timelines to escape this one. Wise.

Dr. Z: Here is what I found for your refined taste...

[Bollywood]
Title   : Krrish
Industry: Bollywood
Type    : Movie
Genre   : sci-fi
Rating  : 6.8/10

[Hollywood]
Title   : Interstellar
Industry: Hollywood
Type    : Movie
Genre   : sci-fi
Rating  : 8.6/10

Netflix code for genre "sci-fi" : 1568, 47147, 2729

Dr. Z: Visit the library again soon.
```



```
Genre: scifi

Dr. Z: Sci-fi? Future tech and alternate timelines to escape this one. Wise.

Dr. Z: Here is what I found for your refined taste...

[Bollywood]
Title   : OK Computer
Industry: Bollywood
Type    : Series
Genre   : sci-fi
Rating  : 6.0/10

[Hollywood]
Title   : Stranger Things
Industry: Hollywood
Type    : Series
Genre   : sci-fi
Rating  : 8.7/10

Netflix code for genre "sci-fi" : 1568, 47147, 2729

Dr. Z: Visit the library again soon.
```

Actual Result:

- All cases normalized successfully
- Suggestions matched correctly

Status: Passed

Testing Conclusion

All major test scenarios were executed, and the system behaved as expected.
The program successfully handled:

- Valid input
- Wrong input
- Repeated attempts
- Random mode
- Specific mode

- Genre normalization
- Exhausted suggestion list

This proves that the implementation is correct, stable, and ready for use.

CONCLUSION

This project gave me a practical understanding of how different concepts of the C programming language can be combined to build a complete, working application. The "Watch or Not" entertainment recommendation system successfully provides users with movie and series suggestions based on their preferences. The features like random mode, genre-based recommendations, Bollywood and Hollywood separation, and Netflix secret codes helped make the program more interesting and useful.

By using structures, arrays, loops, string functions, random number generation, and modular functions, I was able to design a system that is simple but still feels interactive and engaging. The character of Dr. Z and the typing-style output also added a more user-friendly experience.

Overall, this project helped me strengthen my logical thinking, problem-solving skills, and understanding of core C programming concepts. It also showed me how even basic language features can be used creatively to make something fun and functional.