Creating a Knowledge-Based Recommendation System in Rust and Prolog

Objective:

In this project, you will create a knowledge-based recommendation system that will be able to suggest movies to users based on a set of rules or criteria. Using provided information, the system should accept a set of user input preferences and return a list of recommendations that matches those criteria. The project should base these recommendations on the movies name, director, genre, price, and rating. The project should demonstrate using each of these attributes to create recommendations for the user. This project will emphasize the differences between programming in an imperative (Rust) and declarative (Prolog) programming language. Once completed, you will give a **5-minute presentation IN CLASS** explaining your code and discussing how you implemented the various concepts in Rust and Prolog.

Code Requirements:

You will need to create a knowledge-based recommendation system in Rust and Prolog that meets the following constraints and provides the following functionality.

User Interface:

The user interface should be a programming interface that allows the user to enter a desired movie director, genre, price, and rating. **If the user does not wish to designate a constraint for any of the four, the programs should be able to ignore the ones they do not specify.** Furthermore, the interface should allow the user to provide a sorting criteria that allows the programs to sort the results based on any one of the five movie attributes.

Once the user has entered this information, the programs should process this information and display all options that meet the user's entered criteria. When displaying the options, the programs should display the results in the ranked order and with all relevant attributes to the movies. Once the operation has finished, the programs should repeat until the programs is closed. **In Rust, use loops to repeat the process. In Prolog, use recursion to call the process again.**

Constraints:

In this project, the tools you use should depend on the language you are using. For Rust, use imperative programming practices to complete the project. For Prolog, use declarative programming practices to complete the project. The code in both languages should be organized, well documented, and all parts of the code should have a purpose.

Required Functionality:

The programs you create will need to have an initial dataset of movies and store the following information about each movie:

1. Name
2. Director
3. Genre
4. Price
5. Rating

The programs should use this data to provide recommendations to the user based on the criteria they entered while at the interface. To provide recommendations, the programs will need to complete the following tasks:

1. **Filter the movies based on the user criteria**
    a. While at the interface, the user should be able to provide criteria that they wish to use when filtering the recommended movies. The user should be able to specify a specific director, genre, max price, and min rating.
    b. If the user does not wish to restrict the recommendations on any of the five attributes, they should be allowed to enter default value, such as None, that prevents the programs from using them during the filtering process.
2. **Rank the items based on a priority order defined by the user**
    a. Once the filtering process has completed, the programs should proceed to rank the results based on a user defined sorting criteria.
    b. This sorting criteria, should prioritize one of the five movie criteria and results should be sorted based on this.

Ensure that your programs handle edge cases gracefully, such as when no recommendations match the user's criteria or when invalid input is provided.

Research and Discussion:

After completing your code, research and discuss the following topics for your presentation:

- **Imperative Paradigm**: Discuss the imperative programming paradigm.
    - How does Rust's imperative programming features, such as explicit control over memory, mutable state, and control flow impact the programs you can write?
    - In what situations do you believe an imperative programming language like Rust would be an ideal choice?
- **Declarative Paradigm**: Discuss the declarative programming paradigm.
    - How does Prolog's declarative programming features, where logic and rules determine the flow of execution impact the programs you can write?
    - In what situations do you believe a declarative programming language like Prolog would be an ideal choice?
- **Strengths and Weaknesses**: What are the strengths and weaknesses of **Rust** and **Prolog** (e.g., ease of use, performance, efficiency, community, ecosystem, portability, interoperability, scalability, maintainability, reliability, security)?

After finishing, you will be required to give a **5-minute presentation IN CLASS** that includes:

1. A walkthrough of the **Rust and Prolog source codes** you wrote.
2. A demonstration of the programs you built that shows off all the functionality.
3. A discussion on how Rust's imperative features are seen in your program.
4. A discussion on how Prolog's declarative features are seen in your program.
5. Your opinion on the **strengths and weaknesses** of Rust and Prolog, based on your experience writing both source codes.

Resources:

To learn Rust programming,
https://www.rust-lang.org/learn
Or follow videos on Moodle
To learn Prolog programming,
https://lpn.swi-prolog.org/lpnpage.php?pageid=online
Or follow videos on Moodle
To quickly write Rust and Prolog source code,
https://www.onlinegdb.com/
Note 1: Once completed click the download option to download source code, account not needed

Submission Requirements:

- Add your source codes for both your Rust and Prolog programs to a project folder named **lastname_project5**
- Compress/Zip the folder and submit the assignment to the Moodle assignment.
- Come to class during presentation week prepared to give a **5-minute presentation** with **1-minute for follow up questions.**

Grading Criteria:

- **Correctness of the Rust Program**: Your Rust program should be functional, without errors, use imperative features, and implement the correct functionality it is intended to.
- **Correctness of the Prolog Program**: Your Prolog program should be functional, without errors, use declarative features, and implement the correct functionality it is intended to.
- **Clear demonstration of your programs**: Your presentation should showcase you providing a detailed demonstration of your programs, including how to enter criteria, enter sorting criteria, receiving recommendations, and all other functionality it provides.
- **Discussion on imperative, declarative, and strengths/weaknesses**: Your presentation should provide explanation on both language's paradigms and their strengths/weaknesses.
- **Presentation Quality**: Your presentation should include a visual aids (*i.e. code, slides, etc.*), be practiced, well thought out, and effectively use the time provided.
- **Presentation Length**: Must be **4-to-5 minutes** in length but **no longer than 5 minutes**.
- **Question & Answer**: You should be familiar enough to answer questions directed to you during the Q&A session. (*Do not make up answers if you don't know!)*