3. Logical programming part (Prolog) [8% of your final mark]

For this part of the comprehensive assignment, you have to implement the movie recommendation algorithm following the logic programming paradigm. Refer to the general problem description section for the algorithmic steps.

We provide you with two predicates that reads the ratings and movies files. These two predicates create a database of facts based on the `user/3` and `movie/2` predicates. They work as follows:

```
?- read_users('ratings.csv').
true.

?- user(44,Liked,NotLiked).
Liked = [1639, 1552, 1517, 1476, 1429, 1405, 1393, 1210, 1120|...],
NotLiked = [1544, 1391, 891, 849, 842, 836, 833, 798, 786|...].

?- read_movies('movies.csv').
true.

?- movie(32,Title).
Title = 'Twelve Monkeys (a.k.a. 12 Monkeys) (1995)'.
```

Once the two files have been read, you have access to all the users and the list of the movies they liked and not liked and also to the list of all movies and their titles.

In order to implement the recommendation algorithm, we ask you to define the following predicates:

- `similarity(User1,User2,Sim)`: that computes the similarity between two users. You can use the standard `intersection/3` and `union/3` predicates in your definition of this predicate.

  ```
  ?- similarity(33,88,S).
  S = 0.02912621359223301.

  ?- similarity(44,55,S).
  S = 0.013888888888888888.
  ```

- `prob(User,Movie,Prob)`: that computes the probability that the user will like the movie. If the movie has not been seen by at least K users, the probability will be 0.0.

  ```
  ?- prob(44,1080,P).
  P = 0.012250559343815866.
  ?- prob(44,1050,P).
  P = 0.0.
  ```

- `prob_movies(User,Movies,Recs)`: that generates the `(MovieTitle,Prob)` pairs for all movies in the list for the user.

```
?- prob_movies(44,[1010, 1050, 1080, 2000],Rs).
Rs = [('Love Bug, The (1969)', 0.0),
('Looking for Richard (1996)', 0.0),
('Monty Python\'s Life of Brian (1979)', 0.01225055934381),
('Lethal Weapon (1987)', 0.017056653982089866)] .
```

- `liked(Movie, Users, UserWhoLiked)`: that extracts the users in the list who liked the movie.

```
?- liked(1080, [28, 30, 32, 40, 45, 48, 49, 50], U).
U = [28, 45, 50] .
```

- `seen(User, Movie)`: that determines if the user has seen the movie.

```
?- seen(32, 1080).
true .

?- seen(44, 1080).
false.
```

Once these predicates defined, the top-level predicate that will generate the movie recommendation can be simply defined as:

```
recommendations(User) :-
  setof(M,L^movie(M,L),Ms),    % generate list of all movie
  prob_movies(User,Ms,Rec),    % compute probabilities for all movies
  sort(2,@>=,Rec,Rec_Sorted), % sort by descending probabilities
  number_of_rec(N),
  display_first_n(Rec_Sorted,N). % display the result
```

```
?- recommendations(44).
Phenomenon (1996),0.06199173866147424
Ransom (1996),0.05326426015488307
River Wild, The (1994),0.046004744217356015
Nutty Professor, The (1996),0.04578611546283508
Multiplicity (1996),0.04465958653419689
Nixon (1995),0.04465943487441159
Dragonheart (1996),0.044547965249606435
Father of the Bride Part II (1995),0.040951313187447O3
Courage Under Fire (1996),0.03998115958633563
Primal Fear (1996),0.03746013261338603
Long Kiss Goodnight, The (1996),0.03737169178868447
Mighty Aphrodite (1995),0.03730010995370155
Sgt. Bilko (1996),0.036161937292640764
Secret of Roan Inish, The (1994),0.03581702968277858
Truth About Cats & Dogs, The (1996),0.03574558414355256
Leaving Las Vegas (1995),0.03458494730457354
Powder (1995),0.03429940520925141
Ghost and the Darkness, The (1996),0.033168897718840705
Frighteners, The (1996),0.0329282309403291
Star Trek: First Contact (1996),0.03159668701949167
```

Submit your project in a zip file containing the well-commented prolog predicates; all your Prolog predicates must have a short header describing what the predicate does and its parameters. Keep the definitions of predicate `test/1` included in the provided file (and make sure your predicates pass these tests). Include also a text file (`output.txt`) containing the results for :

```
?- recommendations(44).
?- recommendations(55).
?- recommendations(66).
```

3. Partie Logique (Prolog) [8% de votre note finale]

Dans cette partie du projet intégrateur, nous vous demandons de réaliser à nouveau l'algorithme de recommandation de films mais, cette fois, en utilisant le paradigme de programmation logique. Référez-vous à la description générale du problème pour les détails algorithmiques de l'approche.

Vous disposez de deux prédicats qui lisent les fichiers d'évaluations et des films. Ces deux prédicats créent une base de données de faits reposant sur les prédicats `user/3` et `movie/2`. Leur fonctionnement est comme suit :

```
?- read_users('ratings.csv').
true.

?- user(44,Liked,NotLiked).
Liked = [1639, 1552, 1517, 1476, 1429, 1405, 1393, 1210, 1120|...],
NotLiked = [1544, 1391, 891, 849, 842, 836, 833, 798, 786|...].

?- read_movies('movies.csv').
true.

?- movie(32,Title).
Title = 'Twelve Monkeys (a.k.a. 12 Monkeys) (1995)'.
```

Une fois les deux fichiers lus, vous avez donc accès à tous les utilisateurs ainsi qu'à la liste des films qu'ils ont aimés ou non aimés, ainsi qu'à la liste de tous les films et de leurs titres.

Afin d'implémenter l'algorithme de recommandation, nous vous demandons de définir les prédicats suivants :

- `similarity(User1,User2,Sim)` : calculant la similarité entre deux utilisateurs. Vous pouvez utiliser les prédicats standards `intersection/3` and `union/3` dans la définition de ce prédicat.

  ```
  ?- similarity(33,88,S).
  S = 0.02912621359223301.

  ?- similarity(44,55,S).
  S = 0.013888888888888888.
  ```

- `prob(User,Movie,Prob)` : calculant la probabilité que l'utilisateur aime ce film. Si le film n'a pas été vu par au moins K utilisateurs, alors le prédicat retourne 0.0.

  ```
  ?- prob(44,1080,P).
  P = 0.012250559343815866.
  ?- prob(44,1050,P).
  P = 0.0.
  ```

- `prob_movies(User,Movies,Recs)`: générant la paire `(MovieTitle,Prob)` pour tous les films dans la liste pour cet utilisateur.

```
?- prob_movies(44,[1010, 1050, 1080, 2000],Rs).
Rs = [('Love Bug, The (1969)', 0.0),
('Looking for Richard (1996)', 0.0),
('Monty Python\'s Life of Brian (1979)', 0.01225055934381),
('Lethal Weapon (1987)', 0.017056653982089866)] .
```

- `liked(Movie, Users, UserWhoLiked)`: qui extrait les utilisateurs dans la liste qui ont aimé ce film.

```
?- liked(1080, [28, 30, 32, 40, 45, 48, 49, 50], U).
U = [28, 45, 50] .
```

- `seen(User, Movie)`: qui determine si l'utilisateur a vu ce film.

```
?- seen(32, 1080).
true .

?- seen(44, 1080).
false.
```

Une fois tous ces prédicats définis, le prédicat principal générant les recommandations de films pour un utilisateur peut être défini comme suit:

```
recommendations(User) :-
  setof(M,L^movie(M,L),Ms),    % generate list of all movie
  prob_movies(User,Ms,Rec),    % compute probabilities for all movies
  sort(2,@>=,Rec,Rec_Sorted), % sort by descending probabilities
  number_of_rec(N),
  display_first_n(Rec_Sorted,N). % display the result
```

```
?- recommendations(44).
Phenomenon (1996),0.06199173866147424
Ransom (1996),0.05326426015488307
River Wild, The (1994),0.046004744217356015
Nutty Professor, The (1996),0.04578611546283508
Multiplicity (1996),0.04465958653419689
Nixon (1995),0.04465943487441159
Dragonheart (1996),0.044547965249606435
Father of the Bride Part II (1995),0.040951313187447 03
Courage Under Fire (1996),0.03998115958633563
Primal Fear (1996),0.03746013261338603
Long Kiss Goodnight, The (1996),0.03737169178868447
Mighty Aphrodite (1995),0.03730010995370155
Sgt. Bilko (1996),0.036161937292640764
Secret of Roan Inish, The (1994),0.03581702968277858
Truth About Cats & Dogs, The (1996),0.03574558414355256
Leaving Las Vegas (1995),0.03458494730457354
Powder (1995),0.03429940520925141
Ghost and the Darkness, The (1996),0.033168897718840705
Frighteners, The (1996),0.0329282309403291
Star Trek: First Contact (1996),0.03159668701949167
```

Vous devez soumettre votre projet dans un fichier zip contenant les prédicats Prolog bien commentés ; tous vos prédicats Prolog doivent avoir un court en-tête décrivant ce que fait le prédicat et ses paramètres. Conserver les définitions du prédicat `test/1` contenues dans le fichier fourni (et assurez-vous que vos prédicats passent ces tests). Inclure un fichier texte (`output.txt`) contenant les résultats pour :

```
?- recommendations(44).
?- recommendations(55).
?- recommendations(66).
```