



Minha página principal ► Programação em Lógica ► Provas ►
Mini-Teste 1 (Recuperação) -- 2017/02/01

Data de início Quarta, 1 Fevereiro 2017, 17:00

Estado Teste enviado

Data de submissão: Quarta, 1 Fevereiro 2017, 18:53

Tempo gasto 1 hora 52 minutos

Nota 7,70 de um máximo de 20,00 (39%)

Informação

Destacar pergunta

Pretende-se implementar um sistema de recomendação para filmes, baseado na informação de filmes e utilizadores existentes, assim como na relação entre utilizadores e filmes.

Para isso, é guardada informação de cada filme no predicado *film/4* (título, lista de categorias do filme, duração (em minutos) e pontuação média).

De cada utilizador é guardado o seu nome de utilizador, ano de nascimento e país de origem com o predicado *user/3*.

É ainda guardada informação sobre a pontuação atribuída por cada utilizador a filmes, usando o predicado *vote/2*, que contém o nome do utilizador e uma lista com pares filme-pontuação (note-se que um utilizador não classifica necessariamente todos os filmes).

Apresenta-se abaixo um excerto da base de dados deste sistema.

```
%film(Title, Categories, Duration, Voting).
film('Doctor Strange', [action, adventure, fantasy], 115, 7.6).
film('Hacksaw Ridge', [biography, drama, romance], 131, 8.7).
film('Inferno', [action, adventure, crime], 121, 6.4).
film('Arrival', [drama, mystery, scifi], 116, 8.5).
film('The Accountant', [action, crime, drama], 127, 7.6).
film('The Girl on the Train', [drama, mystery, thriller], 112, 6.7).

%user(Username, YearOfBirth, Country)
user(john, 1992, 'USA').
user(jack, 1989, 'UK').
user(peter, 1983, 'Portugal').
user(harry, 1993, 'USA').
user(richard, 1982, 'USA').


%vote(Username, List_of_Film-Rating)
vote(john, ['Inferno'-7, 'Doctor Strange'-9, 'The Accountant'-6]).
vote(jack, ['Inferno'-8, 'Doctor Strange'-8, 'The Accountant'-7]).
vote(peter, ['The Accountant'-4, 'Hacksaw Ridge'-7, 'The Girl on the Train'-3]).
vote(harry, ['Inferno'-7, 'The Accountant'-6]).
vote(richard, ['Inferno'-10, 'Hacksaw Ridge'-10, 'Arrival'-9]).
```

Responda às perguntas 1 a 5 **SEM** utilizar predicados de obtenção de múltiplas soluções (findall, setof e bagof).

Pergunta 1

Respondida

Pontuou 1,200 de 1,500

 Destacar pergunta

Implemente o predicado **raro(+Movie)**, que sucede caso o filme *Movie* tenha uma duração fora do habitual. A duração habitual de um filme é entre 60 minutos e 120 minutos (inclusive).

Exemplo:

```
| ?- raro('Hacksaw Ridge').  
yes
```

```
| ?- raro('The Girl on the Train').  
no
```

```
raro(Movie):-  
film(Movie,_,Time,_),  
Time < 60.
```


```
raro(Movie):-  
film(Movie,_,Time,_),  
Time >= 120.
```

Comentário:

Pergunta 2

Respondida

Pontuou 2,000 de 2,000

 Destacar pergunta

Implemente o predicado **happierGuy(+User1, +User2, -HappierGuy)**, que recebe dois utilizadores (*User1* e *User2*) e devolve em *HappierGuy* o utilizador que atribuiu em média a maior pontuação aos filmes que viu.

Exemplo:

```
| ?- happierGuy(john, peter, HappierGuy).  
HappierGuy = john ? ;  
no
```

happierGuy(User1, User2, HappierGuy):-

```
vote(User1,Vote1),
votes(Vote1,AllVotes1),
sumlist(AllVotes1,N1),
length(AllVotes1,Total1),
AVG1 is N1 / Total1,
```

```
vote(User2,Vote2),
votes(Vote2,AllVotes2),
```

Comentário:

Pergunta 3

Respondida

Pontuou 2,000 de 2,000

Destacar pergunta

Implemente o predicado **likedBetter(+User1, +User2)** que sucede caso o utilizador *User1* tenha atribuído uma votação a um filme que viu (e que o *User2* pode não ter visto) maior que qualquer das votações atribuídas pelo *User2*.

Exemplo:

```
| ?- likedBetter(peter, harry).
no
```

```
| ?- likedBetter(richard, harry).
yes
```

```
votes([],[]).
votes([Film-Vote|T],[H|T1]):-
H = Vote,
votes(T,T1).
```

```
likedBetter(User1,User2):-
vote(User1,Vote1),
votes(Vote1,AllVotes1),
vote(User2,Vote2),
```

Comentário:

Pergunta 4

Respondida

Pontuou 2,500 de 2,500

Destacar pergunta

Um sistema de recomendação baseia as suas recomendações na semelhança entre itens. Neste caso, seria usada a similaridade entre utilizadores.

Implemente o predicado **recommends(+User, -Movie)** que devolve em *Movie* um filme visto por

um utilizador que viu todos os filmes do *User* e mais alguns (utilizador "semelhante"). *Movie* é o primeiro da lista do utilizador semelhante que *User* não viu.

Exemplo:

```
| ?- recommends(harry, L).  
L = 'Doctor Strange' ? ;  
no
```

```
recommends(User,Movie):-  
vote(User,List1),  
vote(User2,List2),  
areSimiliar(List1,List2),!,  
firstDiff(List1,List2,Movie-X).
```

```
areSimiliar([],_).  
areSimiliar([H-X|T],List):-  
member(H-X1,List),
```

Comentário:

Pergunta 5

Não respondida

Pontuação 2,500



Destacar pergunta

Implemente o predicado ***invert(+PredicateSymbol, +Arity)*** que inverte a ordem das clausulas de *PredicateSymbol*/*Arity* guardadas na base de dados interna do Prolog.

Exemplo:

```
antes:  
vote(john, ['Inferno'-7, 'Doctor Strange'-9, 'The Accountant'-6]).  
vote(jack, ['Inferno'-8, 'Doctor Strange'-8, 'The Accountant'-7]).  
vote(peter, ['The Accountant'-4, 'Hacksaw Ridge'-7, 'The Girl on the Train'-3]).  
vote(harry, ['Inferno'-7, 'The Accountant'-6]).  
vote(richard, ['Inferno'-10, 'Hacksaw Ridge'-10, 'Arrival'-9]).
```

```
| ?- invert(vote, 2).  
yes
```

```
depois:  
vote(richard, ['Inferno'-10, 'Hacksaw Ridge'-10, 'Arrival'-9]).  
vote(harry, ['Inferno'-7, 'The Accountant'-6]).  
vote(peter, ['The Accountant'-4, 'Hacksaw Ridge'-7, 'The Girl on the Train'-3]).  
vote(jack, ['Inferno'-8, 'Doctor Strange'-8, 'The Accountant'-7]).  
vote(john, ['Inferno'-7, 'Doctor Strange'-9, 'The Accountant'-6]).
```

Informação

🚩 Destacar pergunta

Nas perguntas seguintes pode fazer uso de predicados de obtenção de múltiplas soluções (findall, setof e bagof).

Pergunta 6

Respondida

Pontuou 0,000 de 2,000

🚩 Destacar pergunta

Implemente o predicado ***onlyOne(+User1, +User2, -OnlyOneList)*** que devolve em *OnlyOneList* a lista de filmes que o *User1* viu mas o *User2* não viu mais os filmes que o *User2* viu mas o *User1* não viu.

Exemplo:

```
| ?- onlyOne(john, jack, List).
```

```
List = [] ? ;
```

```
no
```

```
| ?- onlyOne(john, peter, List).
```

```
List = ['Inferno','Doctor Strange','Hacksaw Ridge','The Girl on the Train'] ? ;
```

```
no
```

```
onlyOne(User1,User2,OnlyOneList):-
```

```
vote(User1,List1),
```

```
vote(User2,List2),
```

```
setFilms(List1,List1R),
```

```
setFilms(List2,List2R),
```

```
dontApp(List1R,List2R,ListD),
```

```
dontApp(List2R,List1R,ListD1),
```

```
append(ListD,ListD1,OnlyOneList).
```

Comentário:

Pergunta 7

Não respondida

Pontuação 2,000

Destacar pergunta

Implemente o predicado ***filmVoting/0*** que coloca na base de dados interna, e para cada filme, a seguinte informação: *filmUsersVotes(Film,[User1-Vote1, User3-Vote3,])*.

Pergunta 8

Não respondida

Pontuação 2,000

Destacar pergunta

Implemente o predicado ***dumpDataBase(+FileName)*** que guarda no ficheiro *FileName* todas as cláusulas dos predicados *user/3*, *film/4* e *vote/2*.

Seguinte ►

NAVEGAÇÃO NO TESTE



Joao Pedro Antunes Pereira Gomes

Mostrar todas as perguntas numa página

Terminar revisão

