


**Data de início** Quarta, 23 Novembro 2016, 17:13**Estado** Teste enviado**Data de submissão:** Quarta, 23 Novembro 2016, 19:12**Tempo gasto** 1 hora 58 minutos**Nota** 6,50 de um máximo de 20,00 (33%)

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, AvgClassification).
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 6 **SEM** utilizar predicados de obtenção de múltiplas soluções (findall, setof e bagof).

Pergunta 1

Respondida

Pontuou 1,500 de 1,500

 Destacar pergunta

Implemente o predicado *curto(+Movie)*, que sucede caso o filme *Movie* seja curto. Um filme é considerado curto quando tem uma duração inferior a 125 minutos.

Exemplo:

```
| ?- curto('Hacksaw Ridge').  
no
```

```
| ?- curto('The Girl on the Train').  
yes
```


```
curto(Movie) :-  
    film(Movie, _, Runtime, _),  
    Runtime < 125.
```

Comentário:

Pergunta 2

Respondida

Pontuou 1,200 de 1,500

 Destacar pergunta

Implemente o predicado *diff(+User1, +User2, -Difference, +Film)* que devolve em *Difference* a diferença (valor absoluto) entre os votos dos utilizadores *User1* e *User2* relativamente ao filme *Film*.

Caso pelo menos um dos utilizadores não tenha visto o filme, o predicado deve falhar.

Exemplos:

```
| ?- diff(john, jack, Diff, 'Inferno').  
Diff = 1 ? ;  
no
```

```
| ?- diff(john, peter, Diff, 'Inferno').  
no
```

```
%Always positive
(Total2 >= Total1 -> Difference is Total2 - Total1; true),
(Total2 < Total1 -> Difference is Total1 - Total2; true).
```

```
parseList([Head-Score | Tail], Film) :-
    %If current Head is Film store score for comparison
    (Head == Film -> assert(total(Score)); true),
    parseList(Tail, Film).

parseList([], _).
```

Comentário:

podia fazer `member(Film-Score1, List1), member(Film-Score2, List2), Difference is abs(Score1 - Score2)`.

código pouco declarativo... (`Head == Film`, uso excessivo de `if then else...`), e pouco eficiente no `parseList` (porque iterar até ao fim mesmo depois de encontrar o voto desejado?)

Pergunta 3

Respondida

Pontuou 0,800 de 1,500

Destacar pergunta

Implemente o predicado *niceGuy(+User)* que sucede caso o utilizador *User* tenha atribuído uma nota igual ou superior a 8 pontos a pelo menos dois filmes diferentes.

Exemplo:

```
| ?- niceGuy(richard).
yes

| ?- niceGuy(peter).
no
```

```
%Cut to only check the 1st value from retract
%which is the value asserted at the end of niceParse
%other values are from backtracking and not useful
(Final >= 2 -> true; fail).
```

```
niceParse([H-S | T], CurCount, Count) :-
    (S > 8 -> NewCount is CurCount + 1; NewCount is CurCount),
    niceParse(T, NewCount, NewCount),
    assert(total(Count)). %Assert the count, later a cut uses only the 1st assert
```

Comentário:

>=8

base de recursividade do niceParse? se nunca termina com sucesso, nunca chega ao assert...
qual a necessidade de ter dois contadores?

Pergunta 4

Respondida

Pontuou 1,500 de 1,500

🚩 Destacar pergunta

Pretende-se implementar um predicado que determine os elementos em comum entre duas listas.

Implemente o predicado *elemsComuns(+List1, -Common, +List2)* que devolve em *Common* uma lista com os elementos que existem em ambas as listas *List1* e *List2*.

Exemplo:

```
| ?- elemsComuns([a, b, d, f, g], L, [b, c, d, g, h]).  
L = [b, d, g] ? ;  
no
```

```
| ?- elemsComuns([a, c, e, g], L, [b, d, f, h]).  
L = [] ? ;  
no
```

```
recursiveList([H|T], Common, Temp, List2) :-  
    %If List1 element is found in List2 then append to final list  
    %else append the same list to keep it unchanged  
    (memberchk(H, List2) -> append(Temp, [H], NewTemp);  
     append([], Temp, NewTemp)),  
    recursiveList(T, Common, NewTemp, List2).
```

```
%Unifies Common  
recursiveList([], Common, Temp, _) :-  
    append([], Temp, Common).
```

Comentário:

código podia estar mais declarativo..

Pergunta 5

Respondida

Pontuou 1,500 de 1,500

🚩 Destacar pergunta

Implemente o predicado *printCategory(+Category)* que imprime na consola informação sobre todos os filmes classificados com uma determinada categoria, incluindo o nome, duração e classificação média. Note que o predicado sucede sempre.

Exemplo:

```
| ?- printCategory(action).  
Doctor Strange (115min, 7.6/10)  
Inferno (121min, 6.4/10)  
The Accountant (127min, 7.6/10)  
yes  
  
| ?- printCategory(none).  
yes
```

```
printCategory(Category) :-  
    film(Name, List, Duration, Score),  
    (memberchk(Category, List) -> printCat(Name, List, Duration, Score); true),  
    fail.  
  
printCategory(_).  
  
printCat(Name, List, Duration, Score) :-  
    write(Name),  
    write(' '),
```

Comentário:
código podia estar mais declarativo...

Pergunta 6

Não respondida

Pontuação 1,500

Destacar pergunta

Um sistema de recomendação baseado em conteúdo (*Content-Based Recommender System*) baseia as suas recomendações na semelhança entre itens. Neste caso, seria usada a similaridade entre filmes.

Implemente o predicado *similarity(+Film1, +Film2, -Similarity)* que devolve em *Similarity* uma medida de similaridade entre os filmes Film1 e Film2.

A similaridade entre dois filmes pode ser medida usando a seguinte fórmula:

$$Similarity = PercentCommonCat - 3 * DurDiff - 5 * ScoreDiff$$

em que *PercentCommonCat* é a percentagem de categorias em comum entre os dois filmes (sobre o total de categorias distintas nos dois filmes); *DurDiff* a diferença absoluta entre as durações dos dois filmes; e *ScoreDiff* a diferença absoluta entre as pontuações médias dos dois filmes.


Exemplos:

```
| ?- similarity('Inferno', 'The Accountant', Score).  
Score = 26.0 ? ;  
no
```

(tanto um filme como outro possuem 3 categorias, 2 das quais comuns (quatro distintas no total), sendo $\text{PercentCommonCat} = 2/4 * 100 = 50$; a diferença de duração é de 6 minutos; a diferença de classificação é de 1.2; $50 - 3*6 - 5*1.2 = 50 - 18 - 6 = 26$)

```
| ?- similarity('Doctor Strange', 'Inferno', Score).  
Score = 26.0 ? ;  
no
```

Informação


 Destacar pergunta

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

Pergunta 7

Não respondida

Pontuação 1,500

 Destacar pergunta

Implemente o predicado *mostSimilar(+Film, -Similarity, -Films)* que devolve em *Similarity* o valor máximo de similaridade com *Film* e em *Films* a lista dos filmes (um ou mais, em caso de empate) com essa similaridade relativamente a *Film*.

Devem ser incluídos na lista de resultados apenas filmes com uma similaridade superior a 10 (medida de similaridade usada em alínea anterior). Caso não existam filmes com similaridade superior a 10, *Similarity* deve ser 0 e *Films* deve ser uma lista vazia.

Exemplos:

```
| ?- mostSimilar('Doctor Strange', Sim, Films).
Sim = 26.0,
Films = ['Inferno'] ? ;
no


| ?- mostSimilar('Inferno', Sim, Films).
Sim = 26.0,
Films = ['The Accountant', 'Doctor Strange'] ? ;
no

| ?- mostSimilar('Other Movie', Sim, Films).
Sim = 0,
Films = [] ? ;
no
```

Pergunta 8

Não respondida

Pontuação 1,500

 Destacar pergunta

Um sistema de recomendação de filtros colaborativos (*Collaborative Filtering Recommender System*) baseia as suas recomendações na semelhança entre utilizadores.

Implemente o predicado *distancia(+User1, -Distancia, +User2)* que devolve em *Distancia* uma medida de distância entre os dois utilizadores, *User1* e *User2*.

A distância entre dois utilizadores pode ser dada pela seguinte fórmula:

$$\text{Dist} = \text{AvgDiff} + \text{AgeDiff}/3 + \text{CountryDiff}$$

AvgDiff é a média da diferença de votos nos filmes em que ambos os utilizadores votaram.

AgeDiff é a diferença entre os anos de nascimento dos utilizadores (valor absoluto).

CountryDiff deve ser igual a 2 caso os países de origem dos utilizadores sejam diferentes, ou 0 caso sejam iguais.

Exemplos:

```
| ?- distancia(john, D, jack).  
D = 4.0 ? ;  
no
```

john e jack votaram em três filmes em comum, sempre com uma diferença de 1 ponto; em média, a diferença entre os dois (AvgDiff) será de 1 ponto por filme. AgeDiff/3 será igual a 1 (um terço da diferença de 3 anos). CountryDiff será igual a 2 (países diferentes).

```
| ?- distancia(john, D, harry).  
D = 0.333333333 ? ;  
no
```

```
| ?- distancia(john, D, peter).  
D = 7.0 ? ;  
no
```



Pergunta 9

Não respondida

Pontuação 1,500



Destacar pergunta

Implemente o predicado *update(+Film)* que atualiza na base de dados a pontuação média do filme *Film* de acordo com os votos registados.


Exemplo:


```
| ?- film('Inferno', _C, _D, Score).  
Score = 6.4 ? ;  
no  
| ?- update('Inferno').  
yes  
| ?- film('Inferno', _C, _D, Score).  
Score = 8.0 ? ;  
no
```

Pergunta 10

Não respondida

Pontuação 1,500

 Destacar pergunta

Considere o seguinte código:

```
what(U, A):-  
    vote(U, VL),!,  
    findall(V, member(_F-V, VL), Vs),  
    length(Vs, L),  
    sumlist(Vs, S),  
    A is S/L.
```

Explique o funcionamento deste predicado, sugerindo nomes mais apropriados para o predicado e para as variáveis.

O cut presente no código é vermelho ou verde? Justifique a sua resposta.

Seguinte ►

NAVEGAÇÃO NO TESTE



Francisco Teixeira Lopes

- i
- 1
- 2
- 3
- 4
- 5
- 6
- i
- 7
- 8
- 9
- 10
- i
- 11
- 12
- 13

Mostrar todas as perguntas numa página

Terminar revisão



Informação

🚩 Destacar pergunta

Considere que num tabuleiro de Xadrez (8x8) só existe um cavalo.

As linhas e colunas são contadas a partir de 1. A célula 1/1 é a célula em baixo à esquerda.

Um cavalo pode efetuar um dos seguintes deslocamentos:

- 2 casas para a esquerda e depois uma casa para cima OU para baixo
- 2 casas para a direita e depois uma casa para cima OU para baixo
- 2 casas para baixo e depois uma casa para a esquerda OU para a direita
- 2 casas para cima e depois uma casa para a esquerda OU para a direita

O cavalo não pode deslocar-se para fora do tabuleiro.

Pergunta 11

Não respondida

Pontuação 1,500

🚩 Destacar pergunta

Implemente em Prolog o procedimento *move/2* que recebe a posição inicial do cavalo no formato *Linha/Coluna* e devolve a lista das posições para onde ele se pode deslocar efetuando um movimento.

Exemplo:

```
| ?- move(1/1, Celulas).  
Celulas = [2/3, 3/2] ? ;  
no
```

Pergunta 12

Não respondida

Pontuação 1,500

🚩 Destacar pergunta

Implemente o procedimento Prolog *podeMoverEmN/3* que recebe a posição inicial de um cavalo L/C e um número inteiro N e devolve a lista (SEM posições repetidas) com todas as posições para onde o cavalo se pode deslocar até N jogadas.


Exemplo:

```
| ?- podeMoverEmN(1/1, 2, Celulas).  
Celulas = [1/1, 1/3, 1/5, 2/3, 2/4, 3/1, 3/2 3/5, 4/2, 4/4, 5/1, 5/3] ? ;  
no
```

Pergunta 13

Não respondida

Pontuação 2,000

 Destacar pergunta

Implemente o procedimento Prolog `minJogadas/3` que recebe uma posição inicial de um cavalo (Li/Ci) e uma posição final (Lf/Cf) e unifica o terceiro argumento com o número mínimo de movimentos que levam o cavalo de Li/Ci a Lf/Cf.

Exemplo:

```
| ?- minJogadas(1/1, 4/4, N).  
N = 2 ? ;  
no
```

Terminar revisão

NAVEGAÇÃO NO TESTE



Francisco Teixeira Lopes

- i
- 1
- 2
- 3
- 4
- 5
- 6
- i
- 7
- 8
- 9
- 10
- i
- 11
- 12
- 13

Mostrar todas as perguntas numa página

Terminar revisão

© 2017 UPdigital - Tecnologias Educativas

Nome de utilizador: Francisco Teixeira Lopes (Sair)

Gestão e manutenção da plataforma Moodle U.PORTO da responsabilidade da unidade de Tecnologias Educativas da UPdigital. Mais informações:

apoio.elearning@uporto.pt | +351 22 040 81 91 | <http://elearning.up.pt>



Based on an original theme created by Shaun Daubney | moodle.org