Informação

Destacar pergunta

Considere uma base de factos para guardar informação sobre utilizadores e jogos da plataforma de jogos Vapor.

O predicado *player/3* contém informação dos jogadores registados na plataforma Vapor.

```
%player(Name, UserName, Age).
player('Danny', 'Best Player Ever', 27).
player('Annie', 'Worst Player Ever', 24).
player('Harry', 'A-Star Player', 26).
player('Manny', 'The Player', 14).
player('Jonny', 'A Player', 16).
```

O predicado *game/3* contém informação sobre os jogos disponíveis na plataforma Vapor.

```
%game(Name, Categories, MinAge).
game('5 ATG', [action, adventure, open-world, multiplayer], 18).
game('Carrier Shift: Game Over', [action, fps, multiplayer, shooter], 16).
game('Duas Botas', [action, free, strategy, moba], 12).
```

O predicado *played/4* contém informação sobre os jogos jogados por cada jogador (existe no máximo um facto por cada par jogador/jogo), nomeadamente número de horas jogadas e percentagem do jogo concluída.

```
%played(Player, Game, HoursPlayed, PercentUnlocked)
played('Best Player Ever', '5 ATG', 3, 83).
played('Worst Player Ever', '5 ATG', 52, 9).
played('The Player', 'Carrier Shift: Game Over', 44, 22).
played('A Player', 'Carrier Shift: Game Over', 48, 24).
played('A-Star Player', 'Duas Botas', 37, 16).
played('Best Player Ever', 'Duas Botas', 33, 22).
```

Responda às perguntas 1 a 6 **SEM** utilizar predicados de obtenção de soluções múltiplas (findall, setof e bagof), e **SEM** usar qualquer biblioteca do SICStus.

Pergunta 1

Respondida Pontuou 1,000 de 1,000 Destacar pergunta

Implemente o predicado ***achievedALot(+Player)*** que sucede se o jogador *Player* completou pelo menos 80% em algum jogo.

Exemplos:

```
| ?- achievedALot('Best Player Ever').
yes
```

```
| ?- achievedALot('Worst Player Ever').
no
```

```
achievedALot(Player):-
    played(Player, _, _, CompletionPercentage),
    CompletionPercentage >= 80.
```

Pergunta **2**

Respondida Pontuou 1,000 de 1,000 Destacar pergunta

Implemente o predicado ***isAgeAppropriate(+Name, +Game)*** que sucede se *Game* é um jogo adequado à idade da pessoa *Name*.

Exemplos:

```
| ?- isAgeAppropriate('Danny', '5 ATG').
yes
```

```
| ?- isAgeAppropriate('Manny', '5 ATG').
no
```

```
isAgeAppropriate(Name, Game):-
    player(Name, _, PlayerAge),
    game(Game, _, MinimumAge),
    PlayerAge >= MinimumAge.
```

Pergunta 3

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***timePlayingGames(+Player, +Games, -ListOfTimes, -SumTimes)*** que determina o número total de horas que o jogador *Player* investiu a jogar cada um dos jogos indicados na lista *Games*, retornando em *ListOfTimes* os tempos de cada um dos jogos na ordem respetiva, e em *SumTimes* o número total de horas.

Exemplos:

```
| ?- timePlayingGames('Best Player Ever', ['5 ATG', 'Duas Botas'], LT, ST).
LT = [3, 33],
ST = 36 ? ;
no
```

```
| ?- timePlayingGames('Worst Player Ever', ['Duas Botas'], LT, ST).
LT = [0],
ST = 0 ? ;
no
```

Pergunta 4

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***listGamesOfCategory(+Cat)*** que imprime na consola os títulos de todos os jogos que se enquadram na categoria *Cat*, indicando ainda a idade mínima recomendada para cada jogo. Note que o predicado sucede sempre.

Exemplos:

```
| ?- listGamesOfCategory(multiplayer).
5 ATG (18)
Carrier Shift: Game Over (16)
yes
```

```
| ?- listGamesOfCategory(simulation).
yes
```

```
listGamesOfCategory(Category):-
    game(Title, GameCategories, MinimumAge),
    member(Category, GameCategories),
    write(Title), write(' ('), write(MinimumAge), write(')'), nl,
    fail.
listGamesOfCategory(_).
```

Pergunta 5

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***updatePlayer(+Player, +Game, +Hours, +Percentage)*** que atualiza a base de conhecimento relativamente ao número de horas que o jogador *Player* jogou o jogo *Game*.

Exemplo:

```
| ?- played('Best Player Ever', 'Duas Botas', Hours, Percent).
Hours = 33,
Percent = 22 ? ;
no

| ?- updatePlayer('Best Player Ever', 'Duas Botas', 5, 12).
yes

| ?- played('Best Player Ever', 'Duas Botas', Hours, Percent).
Hours = 38,
Percent = 34 ? ;
no
```

```
updatePlayer(Player, Game, Hours, Percentage):-
    retract(played(Player, Game, CurrentHours, CurrentPercentage)), !,
    NewHours is CurrentHours + Hours,
    NewPercentage is CurrentPercentage + Percentage,
    assert(played(Player, Game, NewHours, NewPercentage)).

%Para quando o retract falha, significando que ainda não existia a relação para este par jogador/jogo
updatePlayer(Player, Game, Hours, Percentage):-
    assert(played(Player, Game, Hours, Percentage)).
```

Pergunta 6

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***fewHours(+Player, -Games)***, que devolve em *Games* a lista de jogos nos quais o jogador *Player* investiu menos de 10h a jogar.

Exemplos:

```
| ?- fewHours('Best Player Ever', G).
G = ['5 ATG'] ? ;
no
```

```
| ?- fewHours('Worst Player Ever', G).
G = [] ? ;
no
```

```
fewHours(Player, Games):-
    fewHoursAux(Player, [], Games).

fewHoursAux(Player, AccGames, FinalGames):-
    played(Player, GameTitle, HoursSpent, _),
    HoursSpent < 10,
    \+ (member(GameTitle, AccGames)), !,
    fewHoursAux(Player, [GameTitle | AccGames], FinalGames).

%Quando já não houverem mais jogos nessas condições
fewHoursAux(Player, FinalGames, FinalGames).
```

Pergunta 7

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***ageRange(+MinAge, +MaxAge, -Players)*** que devolve em *Players* a lista dos nomes dos jogadores com idade compreendida entre *MinAge* e *MaxAge* (limites inclusivos).

Exemplos:

```
| ?- ageRange(12, 16, P).
P = ['Manny', 'Jonny'] ? ;
no
```

```
| ?- ageRange(24, 34, P).
P = ['Danny', 'Annie', 'Harry'] ? ;
no
```

```
ageRange(MinAge, MaxAge, Players):-
    findall(PlayerName, (player(PlayerName, _, PlayerAge), PlayerAge >= MinAge, PlayerAge =< MaxAge), Players).
```

Pergunta **8**

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***averageAge(+Game, -AverageAge)*** que determina a idade média dos jogadores que jogam o jogo *Game*.

Exemplos:

```
| ?- averageAge('5 ATG', AA).
AA = 25.5 ? ;
no
```

```
| ?- averageAge('Duas Botas', AA).
AA = 26.5 ? ;
no
```

```
%Idêntico ao utilizado na pergunta 3 (de qualquer maneira, fica aqui de novo)
listSum([], Counter, Counter).
listSum([Current | Remaining], CounterAux, CounterFinal):-
    NewCounterAux is CounterAux + Current,
    listSum(Remaining, NewCounterAux, CounterFinal).

averageAge(Game, AverageAge):-
    findall(PlayerAge, (played(UserName, Game, _, _), player(_,UserName, PlayerAge)), AgeList),
    listSum(AgeList, 0, AgeSum),
    length(AgeList, NumPlayers),
    AverageAge is AgeSum/NumPlayers.
```

Pergunta 9

Respondida Pontuou 2,000 de 2,000 Destacar pergunta

Implemente o predicado ***mostEffectivePlayers(+Game, -Players)*** que determina o jogador ou jogadores que jogam o jogo *Game* com maior eficiência. O jogador mais eficiente é aquele que conseguiu alcançar uma maior percentagem de conclusão do jogo num menor número de horas.

Exemplos:

```
| ?- mostEffectivePlayers('5 ATG', BP).
BP = ['Best Player Ever'] ? ;
no
```

```
| ?- mostEffectivePlayers('Carrier Shift: Game Over', BP).
BP = ['The Player', 'A Player'] ? ;
no
```

```
mostEffectivePlayers(Game, Players):-
    findall(Ratio-Player, (played(Player, Game, Hours, Percentage), Ratio is Percentage/Hours), RatioPlayerList),
    sort(RatioPlayerList, SortedRPL),
    reverse(SortedRPL, ReversedRPL),
    [MaxRatio-TopPlayer | RemainingPlayerRatios] = ReversedRPL,
    extractOtherTopPlayers(RemainingPlayerRatios, MaxRatio, [], OtherTopPlayerList),
    append([TopPlayer], OtherTopPlayerList, Players).

extractOtherTopPlayers([Ratio-Player | Remaining], MaxRatio, AccOtherTop, FinalOtherTop):-
    Ratio == MaxRatio, !,
    extractOtherTopPlayers(Remaining, MaxRatio, [Player | AccOtherTop], FinalOtherTop).
```

Pergunta **10**

Respondida Pontuou 1,000 de 1,500 Destacar pergunta

Indique o que faz o predicado ***whatDoesItDo(?X)***, sugerindo nomes mais apropriados para o predicado e variáveis usadas.
Indique ainda qual a funcionalidade do cut presente no código, justificando a sua resposta.

```
whatDoesItDo(X):-
        player(Y, X, Z), !,
        \+ ( played(X, G, L, M),
                game(G, N, W),
                W > Z ).
```

O predicado, para um determinado jogador identificado pelo seu username, verifica se não existe nenhum jogo jogado pelo mesmo que lhe seja inadequado. Por outras palavras, verifica que não existe um jogo jogado pelo mesmo em que a idade mínima recomendada do jogo é superior à idade do jogador.

O cut presente é verde, uma vez que apenas otimiza operações de backtracking: uma que vez o username do jogador é único, uma vez obtidas as informações pelo predicado player/3 não é necessário voltar a tentar recorrer ao mesmo, pelo que caso o programa o tente fazer (porque falhou em encontrar jogos com idade mínima superior à do jogador), deve ser impedido e falhar.

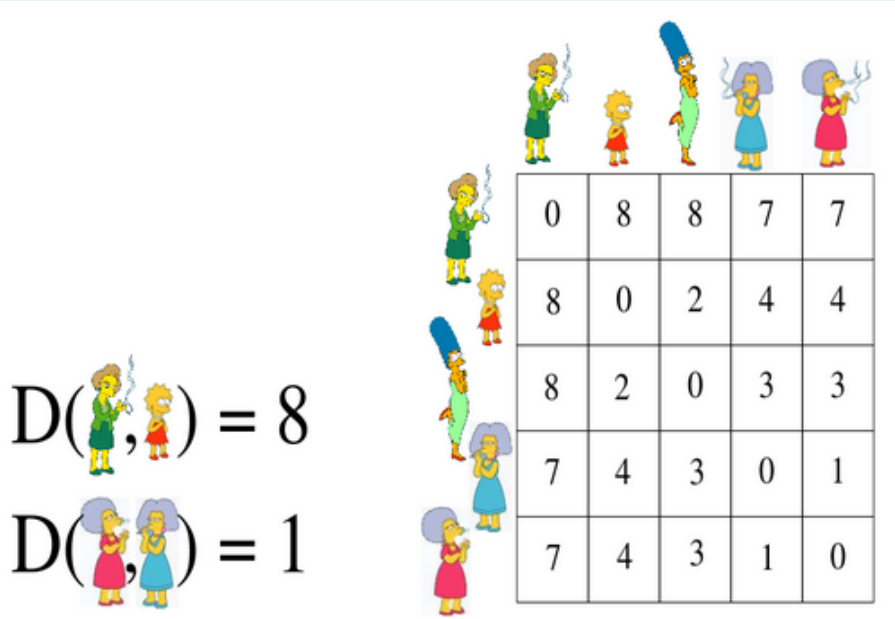
Nomes mais apropriados:
whatDoesItDo - onlyPlayedAppropriateGames
X - PlayerUserName
Y - PlayerName

Comentário:
explicação sobre o cut errada

Informação

Destacar pergunta

Uma matriz de distâncias contém em cada célula (i,j) o valor da distância entre o objeto i e objeto j.



0	8	8	7	7
8	0	2	4	4
8	2	0	3	3
7	4	3	0	1
7	4	3	1	0

Figura 1 - Exemplo de uma matriz de distâncias entre 5 objetos.

Pergunta 11

Respondida Pontuou 0,400 de 0,500 Destacar pergunta

Tendo em consideração a simetria existente nestas matrizes, escolha uma representação em Prolog que considere ser eficiente em termos de utilização de espaço para armazenar uma matriz e represente a informação contida na Figura 1.

Pode ser tudo armazenado numa lista de inteiros, com as distâncias guardadas sequencialmente, linha a linha, representando um dos triângulos da matriz (para o caso, considerarei o triângulo inferior). Assim, no caso, a lista seria [0, 8, 0, 8, 2, 0, 7, 4, 3, 0, 7, 4, 3, 1, 0]. Para aceder a um elemento, através de Linha/Coluna, faz-se o seguinte:
- Se Coluna > Linha, dar swap nos dois valores
- Caso contrário, aceder ao elemento, cujo índice (começando em 1) é Func(Linha-1) + Coluna, em que Func(X) devolve a soma dos números naturais até X (facilmente obtém por X(X+1)/2, sendo que para X = 0 o resultado é 0).

Pergunta 12

Respondida Pontuou 1,000 de 2,000 Destacar pergunta

Codifique em Prolog o predicado ***areClose/3*** que recebe como primeiro argumento uma distância máxima, como segundo argumento uma matriz de distâncias e instancia o terceiro argumento com a lista de pares de objetos que estão a uma distância igual ou inferior à distância dada no primeiro argumento. A lista resultado não deve ter pares simétricos. Identifique os objetos pelo índice da sua linha (ou coluna) na Figura 1.

Exemplo (*MatDist* é a sua codificação da matriz da Figura 1):

```
| ?- areClose(3, MatDist, Pares).
Pares = [3/2, 4/3, 5/3, 5/4] ;
no
```

```
countZeros([], FinalCounter, FinalCounter).
countZeros([Current | Rem], AuxCounter, FinalCounter):-
    Current == 0, !,
    NewAuxCounter is AuxCounter + 1,
    countZeros(Rem, NewAuxCounter, FinalCounter).
countZeros([Current | Rem], AuxCounter, FinalCounter):-
    countZeros(Rem, AuxCounter, FinalCounter).

sumNaturals(0, 0).
sumNaturals(N, Result):-
    Result is integer(N*(N+1)/2)
```

Comentário:

Informação

Destacar pergunta

Considere o dendograma da Figura 2. Considere que um dendograma é uma árvore binária em que os objetos estão guardados nas folhas e cada nó tem um identificador único. Os objetos têm apenas um nome que é um átomo e os nós internos têm um número inteiro que é único (não há dois nós internos com o mesmo número).

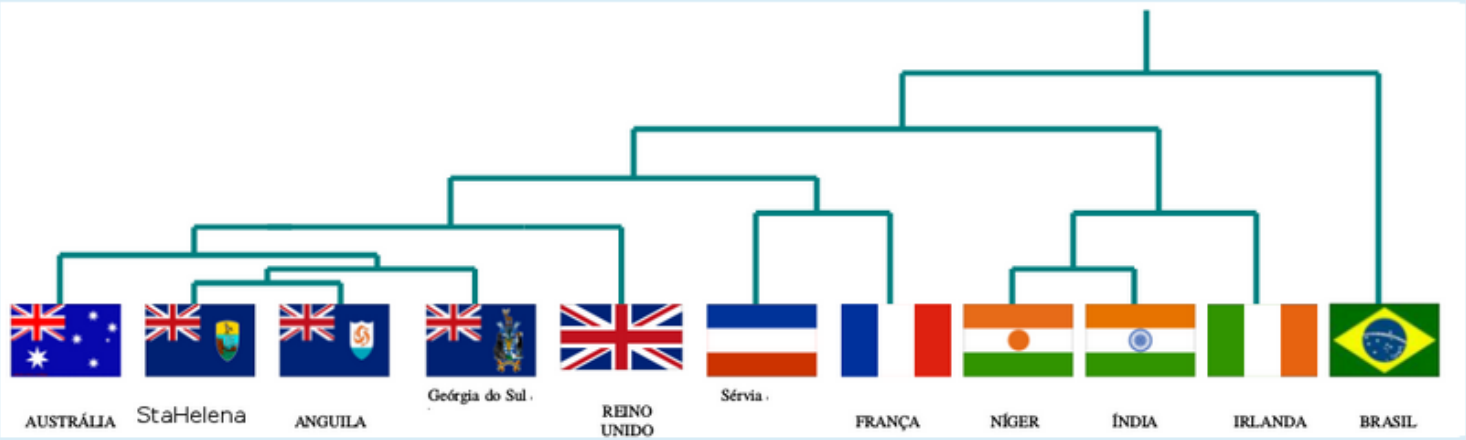


Figura 2 Dendograma de bandeiras de países.

Pergunta 13

Respondida Pontuou 1,000 de 1,000 Destacar pergunta

Proponha uma estrutura para codificar dendogramas e represente nessa estrutura o dendograma da Figura 2.

Um nó do dendograma é representado por [ID, subÁrvoreEsquerda, subÁrvoreDireita], sendo que uma sub-árvore é representada da mesma maneira. No caso da folha, apenas é utilizado o seu átomo.

O dendograma da figura seria representado da seguinte forma:

[1,[2,[5,[7,[8,australia,[9,[10, stahelena, anguila], georgiadosul]],reinounido], [6,servia,franca]],[3,[4,niger,india],irlanda]], brasil]

Pergunta **14**

Respondida Pontuou 0,000 de 2,000 Destacar pergunta

A distância entre dois quaisquer objetos é definida como a altura do nó mais baixo comum aos dois objetos. A altura a que estão as folhas é zero. A altura de um nó interno é 1 mais o máximo entre a altura da sub-árvore esquerda e sub-árvore direita.

Por exemplo:
- a altura da raiz da árvore na Figura 2 é 7
- a distância (na Figura 2) entre a bandeira do Brasil e da Irlanda é 4.

Implemente em Prolog o predicado ***distance/4*** que recebe nos dois primeiros argumentos o nome de dois objetos, recebe um dendograma no terceiro argumento e devolve no quarto argumento a distância entre os dois objetos dados. Assuma que os objetos existem sempre no dendograma.

Exemplo:

```
| ?- distance(brasil, niger, DendogramaFigura2, Distancia).
Distancia = 4 ? ;
no
```