

[Painel do utilizador](#)

As minhas unidades curriculares

[Programação Funcional e em Lógica](#)[Avaliação](#)[MT2 \(27/1/2022\)](#)**Início** segunda, 16 de janeiro de 2023 às 23:19**Estado** Prova submetida**Data de
submissão:** segunda, 16 de janeiro de 2023 às 23:23**Tempo gasto** 4 minutos 25 segundos

Informação

Considere a seguinte base de conhecimento relativa a um espetáculo de dança da companhia *Perfect Footwork and Leaps* (PFL).

- O espetáculo consiste em N rondas.
- Cada ronda tem um número único (*RoundNumber*), entre 1 e N , que indica a ordem das N rondas.
- Em cada ronda dança-se um estilo de dança único, podendo a duração de cada ronda ser diferente.
- As danças podem ser rápidas (*fast*) ou lentas (*slow*).
- Todas as danças são feitas em pares.
- Uma pessoa só participa numa ronda num único par.
- Nem todos os dançarinos participam em todas as rondas.
- Os dançarinos podem trocar de pares entre rondas.

```
:- dynamic round/4.
```

```
% round(RoundNumber, DanceStyle, Minutes, [Dancer1-Dancer2 | DancerPairs])
% round/4 indica, para cada ronda, o estilo de dança, a sua duração, e os pares de dançarinos participantes.
round(1, waltz, 8, [eugene-fernanda]).
round(2, quickstep, 4, [asdrubal-bruna,cathy-dennis,eugene-fernanda]).
round(3, foxtrot, 6, [bruna-dennis,eugene-fernanda]).
round(4, samba, 4, [cathy-asdrubal,bruna-dennis,eugene-fernanda]).
round(5, rhumba, 5, [bruna-asdrubal,eugene-fernanda]).

% tempo(DanceStyle, Speed).
% tempo/2 indica a velocidade de cada estilo de dança.
tempo(waltz, slow).
tempo(quickstep, fast).
tempo(foxtrot, slow).
tempo(samba, fast).
tempo(rhumba, slow).
```

Responda às perguntas 1 a 7 **SEM** utilizar predicados de obtenção de soluções múltiplas (*findall*, *setof* e *bagof*), e **SEM** usar qualquer biblioteca do SICStus.

Pergunta 1

Respondida Pontuação 1,000

Implemente o predicado *style_round_number(?DanceStyle, ?RoundNumber)* que permita associar um estilo de dança e ronda respetiva.

```
style_round_number(DanceStyle, RoundNumber) :- round(RoundNumber, DanceStyle)
```



```
style_round_number(Style, Round):-  
    round(Round, Style, _Min, _Pairs).
```

Pergunta 2

Não respondida Pontuação 1,000

Implemente o predicado *n_dancers(?RoundNumber, -NDancers)* que permite associar o número de dançarinos de cada ronda.

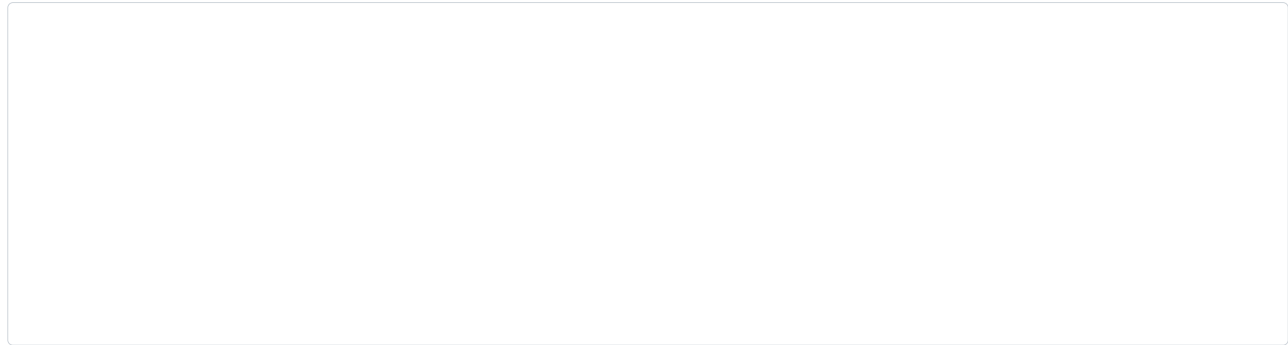
```
n_dancers(Round, NDancers):-  
    round(Round, _Style, _Mins, Pairs),  
    length(Pairs, Len),  
    NDancers is 2*Len.
```

Pergunta 3

Não respondida

Pontuação 1,500

Implemente o predicado *danced_in_round(?RoundNumber, ?Dancer)* que permite determinar quem participou em cada ronda.



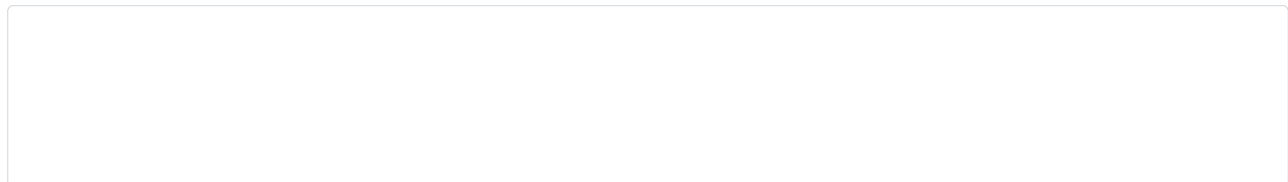
```
danced_in_round(Round, Dancer):-  
    round(Round, _Style, _Mins, Pairs),  
    member(Dancer-_Pair, Pairs).  
danced_in_round(Round, Dancer):-  
    round(Round, _Style, _Mins, Pairs),  
    member(_Pair-Dancer, Pairs).
```

Pergunta 4

Não respondida

Pontuação 1,500

Implemente o predicado *n_rounds(-NRounds)* que determina o número total de rondas.



```
n_rounds(Round):-  
    round(Round, _Style, _Mins, _Pairs),  
    \+( (round(R1,_,_,_), R1 > Round) ).
```

Pergunta 5

Não respondida

Pontuação 1,500

Implemente o predicado *add_dancer_pair(+RoundNumber, +Dancer1, +Dancer2)* que modifica a base de dados, acrescentando o par de dançarinos como participantes de uma ronda.

Caso a ronda não exista, o predicado deve falhar.

Caso um dos dançarinos participe já da ronda indicada, o predicado deve também falhar.



```
add_dancer_pair(Round, Dancer1, Dancer2):-
    \+ danced_in_round(Round, Dancer1),
    \+ danced_in_round(Round, Dancer2),
    retract( round(Round, Style, Minutes, Pairs) ),
    assert( round(Round, Style, Minutes, [Dancer1-Dancer2|Pairs]) ).
```

Pergunta 6

Não respondida

Pontuação 1,500

Implemente o predicado *total_dance_time(+Dancer, -Time)* que determina o número total de minutos dançados pelo dançarino *Dancer*.

```
total_dance_time(Dancer, Time):-
    danceTime(Dancer, [], 0, Time).

danceTime(Dancer, Rounds, Temp, Time):-
    danced_in_round(Round, Dancer),
    \+ member(Round, Rounds),!,
    round(Round, _Style, Mins, _Pairs),
    NTemp is Temp + Mins,
    danceTime(Dancer, [Round|Rounds], NTemp, Time).
danceTime(_Dancer, _Rounds, Time, Time).
```

Pergunta 7

Não respondida Pontuação 1,000

Implemente o predicado *print_program/0* que imprime na consola o programa, no formato 'Estilo (Minutos) - Pares'. Exemplo (note que o predicado deve suceder):

```
| ?- print_program.  
waltz (8) - 1  
quickstep (4) - 3  
...  
rhumba (5) - 2  
yes
```



```
print_program:-  
    round(Round, Style, Mins, Pairs),  
    length(Pairs, NPairs),  
    write(Style),  
    write(' ('), write(Mins),  
    write(') - '), write(NPairs),  
    nl,  
    fail.  
print_program.
```

Informação

Nas perguntas seguintes, pode fazer uso de predicados de obtenção de múltiplas soluções (*findall*, *setof* e *bagof*) e das bibliotecas do SICStus.

Pergunta 8

Não respondida

Pontuação 1,500

Implemente o predicado *dancer_n_dances(?Dancer, ?NDances)* que associa um dançarino com o número de rondas em que participou.

```
dancer_n_dances(Dancer, NRounds):-
    bagof(Round, danced_in_round(Round, Dancer), List),
    length(List, NRounds).
```

Pergunta 9

Não respondida

Pontuação 2,000

Implemente o predicado *most_tireless_dancer(-Dancer)* que devolve o dançarino (ou um dos dançarinos em caso de empate) que dançou durante mais tempo na competição.

```
most_tireless_dancer(Tireless):-
    setof(Time-Dancer, Round^(
        danced_in_round(Round, Dancer),
        total_dance_time(Dancer, Time)
    ), List),
    reverse(List, [_Time-Tireless | _]).
```

Informação

Considere o seguinte excerto de código em Prolog para o predicado *predX(+Arg1, ?Arg2)*:

```
predX([],0).
predX([X|Xs],N):-
    X =.. [_|T],
    length(T,2),
    !,
    predX(Xs,N1),
    N is N1 + 1.
predX([_|Xs],N):-
    predX(Xs,N).
```

Pergunta 10

Não respondida

Pontuação 1,000

Descreva de forma sucinta (ex., numa só frase) o que faz o predicado *predX/2*.



O predicado *predX(+L,-N)* recebe uma lista *L* de termos e retorna em *N* o número de termos em *L* de aridade 2.

Pergunta 11

Não respondida

Pontuação 1,000

O *cut* presente no código de *predX/2* é verde ou vermelho? Justifique brevemente.

O *cut* é vermelho pois afeta o conjunto de soluções devolvidas por *predX/2*. Se for removido, *predX/2* retorna diferentes valores de *N* (inferiores ao resultado correto) por backtracking.

Pergunta 12

Não respondida

Pontuação 0,500

Qual das seguintes afirmações sobre a unificação de duas variáveis *X* e *Y* está correta?

- ☐ a. A unificação produz sempre o mínimo de substituições possíveis para que *X* e *Y* sejam idênticos.
- ☐ b. A unificação pode suceder se *X* for uma lista de comprimento *N* e *Y* for um termo de aridade *N-1*.
- ☐ c. A unificação pode suceder múltiplas vezes por backtracking.
- ☐ d. A unificação sucede se *X* e *Y* forem termos com a mesmo functor e aridade.
- ☐ e. A unificação só pode suceder se *X* ou *Y* tiver alguma variável não-instanciada.

A sua resposta está incorreta.

Ver definição do MGU (most general unifier).

A resposta correta é: A unificação produz sempre o mínimo de substituições possíveis para que *X* e *Y* sejam idênticos.

Pergunta 13

Não respondida Pontuação 0,500

Qual das seguintes afirmações sobre listas de diferença (difference lists) está correta?

- ☐ a. Todas as restantes afirmações estão erradas.
- ☐ b. $[2,1] \setminus [2]$ é uma lista de diferença equivalente a $[1]$.
- ☐ c. O uso de listas de diferença permite reimplementar certos predicados de complexidade temporal quadrática em tempo linear.
- ☐ d. A única lista de diferença equivalente a $[]$ é $[] \setminus []$.
- ☐ e. O uso de listas de diferença permite calcular o comprimento do sufixo de uma lista em tempo constante.

A sua resposta está incorreta.

Considere-se o exemplo visto do predicado *append/3*, que passa de complexidade linear no tamanho da primeira lista para constante. Considere-se agora um outro predicado que chama o predicado *append/3* N vezes, resultando numa complexidade quadrática; usando listas de diferença, a complexidade passa a ser linear pois cada *append* é agora realizado em tempo constante.

A resposta correta é: O uso de listas de diferença permite reimplementar certos predicados de complexidade temporal quadrática em tempo linear.

Informação

Pretende-se definir operadores que permitam escrever factos no seguinte formato:

```
cathy dances quickstep.
asdrubal dances samba and rhumba.
fernanda dances waltz and quickstep and foxtrot.
```

Pergunta 14

Não respondida Pontuação 0,250

Qual a forma mais correta de definir o operador "and" em termos sintáticos e semânticos?

- ☐ a. `:- op(580, xfx, and).`
- ☐ b. `:- op(580, yfy, and).`
- ☐ c. `:- op(580, fx, and).`
- ☐ d. `:- op(580, xfy, and).`
- ☐ e. `:- op(580, fy, and).`

A sua resposta está incorreta.

É a única que permite ter sub-termos como `'and(waltz, and(quickstep, foxtrot))'`

A resposta correta é:

```
:- op(580, xfy, and).
```


Pergunta 15

Não respondida

Pontuação 0,250

Qual a forma mais correta de definir o operador "dances" em termos sintáticos e semânticos?

☐ a. `:- op(560, fx, dances).`

☐ b. `:- op(600, xf, dances).`

☐ c. `:- op(560, xfx, dances).`

☐ d. `:- op(560, xf, dances).`

☐ e. `:- op(600, xfx, dances).`

A sua resposta está incorreta.

É a única opção que permite termos com 'dances' na raiz da árvore, e dois operandos (pessoa e danças).

A resposta correta é:

`:- op(600, xfx, dances).`

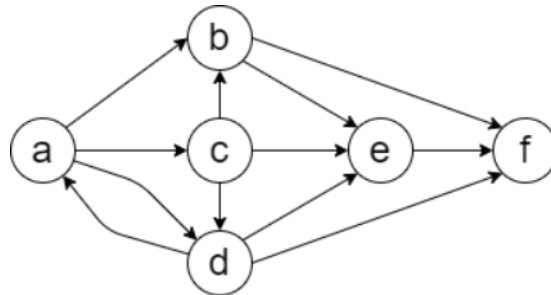
Informação

Pretende-se criar uma aplicação para navegação segura numa cidade em que algumas ruas são consideradas inseguras.

Para isso, considere que o predicado *edge/2* define arestas unidireccionais de um grafo dirigido (os nós representam interseções, e as arestas troços da rede viária).

Considere a seguinte base de conhecimento exemplo, que corresponde à imagem abaixo (para simplificação, usam-se apenas letras para representar as interseções):

```
edge(a,b).
edge(a,c).
edge(a,d).
edge(b,e).
edge(b,f).
edge(c,b).
edge(c,d).
edge(c,e).
edge(d,a).
edge(d,e).
edge(d,f).
edge(e,f).
```



Pergunta 16

Não respondida Pontuação 2,500

Implemente o predicado *shortest_safe_path(+Origin, +Destination, +ProhibitedNodes, -Path)* que devolve em *Path* um dos caminhos mais curtos (em termos de número de arestas percorridas) entre os vértices *Origin* e *Destination*, evitando percorrer qualquer um dos vértices presente na lista *ProhibitedNodes*.

Será avaliado apenas o primeiro resultado da execução do predicado (ie, não serão avaliados resultados obtidos posteriormente por backtracking).

Se *ProhibitedNodes* incluir *Origin* ou *Destination*, então não existe um caminho válido.

Se não existir um caminho válido, o predicado deve falhar.

Exemplos:

```
| ?- shortest_safe_path(a, f, [], P).
P = [a,b,f] ?
| ?- shortest_safe_path(a, f, [b], P).
P = [a,d,f] ?
| ?- shortest_safe_path(a, f, [b,d], P).
P = [a,c,e,f] ?
| ?- shortest_safe_path(a, f, [a], P).
no
```

```
shortest_safe_path(Ni, Nf, PNs, Path):-
    \+member(Ni, PNs),
    \+member(Nf, PNs),
    bfs([[Ni]], Nf, PNs, PathInv),
    reverse(PathInv, Path).

bfs( [ [Nf|T]|_], Nf, _, [Nf|T]).
bfs( [ [Na|T]|Ns], Nf, PNs, Sol):-
    findall(
        [Nb,Na|T],
        (edge(Na,Nb), \+member(Nb, [Na|T]), \+member(Nb, PNs)),
        Ns1),
    append(Ns, Ns1, Ns2),
    bfs(Ns2, Nf, PNs, Sol).
```

Pergunta 17

Não respondida

Pontuação 1,500

Implemente o predicado *all_shortest_safe_paths(+Origin, +Destination, +ProhibitedNodes, -ListOfPaths)* que devolve em *ListOfPaths* a lista de todos os caminhos mais curtos entre *Origin* e *Destination* que não passem pelos vértices em *ProhibitedNodes*.

Se não existir um caminho válido, então o predicado deve falhar.

Exemplos:

```
| ?- all_shortest_safe_paths(a,f,[],L).  
L = [[a,b,f],[a,d,f]] ? ;  
no  
| ?- all_shortest_safe_paths(a,f,[a],L).  
no
```

```
all_shortest_safe_paths(Ni, Nf, PNs, L):-  
    shortest_safe_path(Ni, Nf, PNs, AShortestPath),  
    !,  
    length(AShortestPath, N),  
    length(Path, N),  
    findall(Path, shortest_safe_path(Ni, Nf, PNs, Path), L).
```

[◀ TP2 - Entrega](#)[MT1 \(27/10/2022\) \(Part 1\) ▶](#)