Mini-Teste 1 Modelo (página 1 de 2)

Informação

Destacar pergunta

Texto informativo

No concurso FEUPGotTalent, cada estudante pode participar mostrando as suas habilidades num qualquer tema, académico ou extra-curricular. Os interessados inscrevem-se, dando o número de estudante, idade e o nome da sua atuação:

```
%participant(Id,Age,Performance)
participant(1234, 17, 'Pé coxinho').
participant(3423, 21, 'Programar com os pés').
participant(3788, 20, 'Sing a Bit').
participant(4865, 22, 'Pontes de esparguete').
participant(8937, 19, 'Pontes de pen-drives').
participant(2564, 20, 'Moodle hack').
```

As atuações são apreciadas por um júri de ***E*** elementos.

Ao longo da atuação (que tem um máximo de 120 segundos), se um elemento do júri achar que o participante não deve passar à próxima fase, carrega num botão. Ficam registados os tempos em que cada elemento do júri carregou no botão. Se não carregou, ficam registados 120 segundos.

```
%performance(Id,Times)
performance(1234,[120,120,120,120]).
performance(3423,[32,120,45,120]).
performance(3788,[110,2,6,43]).
performance(4865,[120,120,110,120]).
performance(8937,[97,101,105,110]).
```

Passam à próxima fase os ***N*** participantes que mais se aguentaram em palco, somados os tempos de cada elemento do júri,

desde que pelo menos um dos elementos do júri não tenha carregado no botão.

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

Pergunta 1

Resposta guardada

Pontuação 1,00

Destacar pergunta

Enunciado da pergunta

Implemente o predicado madeItThrough(+Participant), que sucede se Participant é um participante que já atuou e em cuja atuação pelo menos um elemento do júri não carregou no botão.

```
| ?- madeItThrough(1234).
yes

| ?- madeItThrough(2564).
no

| ?- madeItThrough(3788).
no
```

```
madeItThrough(Participant) :-

    performance(Participant, Buttons),

    member(120, Buttons).
```

Pergunta 2

Resposta guardada

Pontuação 1,50

Destacar pergunta

Enunciado da pergunta

Implemente o predicado juriTimes(+Participants, +JuriMember, -Times, -Total), que devolve em Times o tempo de atuação de cada participante na lista Participants (pela mesma ordem) até que o júri número JuriMember (de 1 a E) carregou no botão, e em Total a soma desses tempos.

```
| ?- juriTimes([1234,3423,3788,4865,8937],1,Times,Total).
Times = [120,32,110,120,97],
Total = 479

| ?- juriTimes([1234,3423,3788,4865,8937],2,Times,Total).
Times = [120,120,2,120,101],
Total = 463
```

```
juriTimes([X], J, [T], T) :-
    performance(X, Times),
    nth1(J, Times, T).
juriTimes([X|Participants], J, [T|Times], Total) :-
    juriTimes(Participants, J, Times, Total1),
    juriTimes([X], J, [T], T),
   Total is Total1 + T.
nth1(1, [X|_], X) :- !.
```

nth1(N, [_|L], X) :- N1 is N-1, nth1(N1, L, X).

Pergunta 3

Resposta guardada

Pontuação 1,00

Destacar pergunta

Enunciado da pergunta

Implemente o predicado patientJuri(+JuriMember) que sucede se o júri JuriMember já se absteve de carregar no botão pelo menos por duas vezes.

```
| ?- patientJuri(3).
no

| ?- patientJuri(4).
yes
```

```
patientJuri(J) :-
    juriTimes([X, Y], J, [TX, TY], _),
    X =\= Y,
   TX =:= 120,
   TY =:= 120.
```

Pergunta 4

Resposta guardada

Pontuação 1,50

Destacar pergunta

Enunciado da pergunta

Implemente o predicado bestParticipant(+P1, +P2, -P) que unifica P com o melhor dos dois participantes P1 e P2. O melhor participante é aquele que tem uma maior soma de tempos na sua atuação (independentemente de estar ou não em condições de passar à próxima fase). Se ambos tiverem o mesmo tempo total, o predicado deve falhar.

```
| ?- bestParticipant(3423,1234,Z).
Z = 1234

| ?- bestParticipant(1234,1234,Z).
no
```

```
bestParticipant(P1, P2, P) :-
    performance(P1, Times1), sum_list(Times1, T1),
   performance(P2, Times2), sum_list(Times2, T2),
   (
      (T1 > T2, P = P1);
      (T1 < T2, P = P2)
    ).
sum_list([], 0).
sum_list([X|L], R) :-
```

sum_list(L, R1),

R is R1+X.

Pergunta 5

Resposta guardada

Pontuação 1,00

Destacar pergunta

Enunciado da pergunta

Implemente o predicado allPerfs, que imprime na consola os números dos participantes que já atuaram, juntamente com o nome da sua atuação e lista de tempos.

```
| ?- allPerfs.
1234:Pé coxinho:[120,120,120,120]
3423:Programar com os pés:[32,120,45,120]
3788:Sing a Bit:[110,2,6,43]
4865:Pontes de esparguete:[120,120,110,120]
8937:Pontes de pen-drives:[97,101,105,110]
yes
```

allPerfs :- allPerfs([]).

allPerfs(Visited) :-

idNotIn(Visited, Id),!,

printPerf(Id),

allPerfs([Id|Visited]).

```
allPerfs(_).
printPerf(Id) :-
   participant(Id, _, Performance),
   performance(Id, Times),
   write(Id),format(":",[]),write(Performance),format(":",[]),write(Times),nl.
idNotIn(Visited, Id) :-
   performance(Id,_),
   \+(member(Id, Visited)).
```

Informação

Destacar pergunta

Texto informativo

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

Pergunta 6

Resposta guardada

Pontuação 1,00

Destacar pergunta

Enunciado da pergunta

Implemente o predicado nSuccessfulParticipants(-T) que determina quantos participantes não tiveram qualquer clique no botão durante a sua atuação.

```
| ?- nSuccessfulParticipants(T).
T = 1
```

```
nSuccessfulParticipants(T) :-
    findall(P, performance(P, [120,120,120,120]), L),
    length(L, T).
```

Pergunta 7

Resposta guardada

Pontuação 1,50

Destacar pergunta

Enunciado da pergunta

Implemente o predicado juriFans(juriFansList), que obtém uma lista contendo, para cada participante, a lista dos elementos do júri que não carregaram no botão ao longo da sua atuação.

```
| ?- juriFans(L).
L = [1234-[1,2,3,4],3423-[2,4],3788-[],4865-[1,2,4],8937-[]]
```

```
juriFans(JuriFansList) :-
    findall(P-J, (performance(P, T), timesToFans(1, T, J)), JuriFansList).
```

```
timesToFans(_, [], []) :- !.

timesToFans(Idx, [120|Times], [Idx|Fans]) :- !,
    Idx1 is Idx+1,
    timesToFans(Idx1, Times, Fans).
timesToFans(Idx, [_|Times], Fans) :-
    Idx1 is Idx+1,
    timesToFans(Idx1, Times, Fans).
```

Pergunta 8

Resposta guardada

Pontuação 1,50

Destacar pergunta

Enunciado da pergunta

O seguinte predicado permite obter participantes, suas atuações e tempos totais, que estejam em condições de passar à próxima fase: para um participante poder passar, tem de haver pelo menos um elemento do júri que não tenha carregado no botão durante a sua atuação.

```
:- use_module(library(lists)).

eligibleOutcome(Id,Perf,TT) :-

performance(Id,Times),
    madeItThrough(Id),
```

```
    participant(Id,_,Perf),
    sumlist(Times,TT).
```

Fazendo uso deste predicado, implemente o predicado nextPhase(+N, -Participants), que obtém a lista com os tempos totais, números e atuações dos N melhores participantes, que passarão portanto à próxima fase. Se não houver pelo menos N participantes a passar, o predicado deve falhar.

```
| ?- nextPhase(2,P).
P = [480-1234-'Pé coxinho',470-4865-'Pontes de esparguete']

| ?- nextPhase(3,P).
P = [480-1234-'Pé coxinho',470-4865-'Pontes de esparguete',317-3423-'Programar com os pés']

| ?- nextPhase(4,P).
no
```

```
nextPhase(N, ParticipantsSlice) :-
    setof(TT-Id-Perf, eligibleOutcome(Id, Perf, TT), Participants),
    keysort(Participants, ParticipantsSorted),
    reverse(ParticipantsSorted, ParticipantsSortedReverse),
    slice(ParticipantsSortedReverse, 0, N, ParticipantsSlice).

slice(_, 0, 0, []) :- !.

slice([X|L], 0, Right, [X|R]) :- !, Right1 is Right-1, slice(L, 0, Right1, R).

slice([_|L], Left, Right, R) :-
    Left1 is Left-1,
    Right1 is Right-1,
    slice(L, Left1, Right1, R).
```

Pergunta 9

Resposta guardada

Pontuação 1,00

Destacar pergunta

Enunciado da pergunta

Explique o que faz o predicado predX/3 apresentado abaixo. Indique ainda se o cut utilizado é verde ou vermelho, justificando a sua resposta.

```
predX(Q,[R|Rs],[P|Ps]) :-
    participant(R,I,P), I=<Q, !,
    predX(Q,Rs,Ps).
predX(Q,[R|Rs],Ps) :-
    participant(R,I,_), I>Q,
    predX(Q,Rs,Ps).
predX(_,[],[]).
```

predX(IdadeLimite, Participantes, Performances) retorna em Performances os nomes das performances realizadas por pessoas em Participantes com idade inferior ou igual a IdadeLimite.

O cut utilizado é verde, dado que cada expressão de predX faz a verificação dos seus requisitos, e não depende do backtracking de outras expressões para poder assumir que os seus próprios requisitos são cumpridos. Isto porque:

- A 1ª expressão verifica que o 2º argumento não é uma lista vazia e que I=<Q;

- A 2ª expressão verifica que o 2º argumento não é uma lista vazia e que I > Q;

- A 3ª expressão verifica que o 2º argumento é uma lista vazia.

Em suma, o cut é verde porque não afeta o conjunto de soluções possíveis, dado que cada expressão de predX faz toda a verificação que necessita para ser aplicada de forma correta.