Teste 1 – 2018

Considere os seguintes predicados numa base de dados em Prolog relativa aos voos diários de algumas companhias aéreas.

O predicado *airport/3* contém informação relativa a aeroportos, incluindo o seu nome, identificador ICAO (código internacional único para cada aeroporto) e o país onde se localiza.

```
%airport(Name, ICAO, Country).
airport('Aeroporto Francisco Sá Carneiro', 'LPPR', 'Portugal').
airport('Aeroporto Humberto Delgado', 'LPPT', 'Portugal').
airport('Aeropuerto Adolfo Suárez Madrid-Barajas', 'LEMD', 'Spain').
airport('Aéroport de Paris-Charles-de-Gaulle Roissy Airport', 'LFPG', 'France').
airport('Aeroporto Internazionale di Roma-Fiumicino - Leonardo da Vinci', 'LIRF', 'Italy').
```

O predicado *company/4* contém informação relativa a companhias aéreas, incluindo o seu identificador ICAO (código único para cada companhia aérea), nome, ano de fundação, e país onde está sediada.

```
%company(ICAO, Name, Year, Country).
company('TAP', 'TAP Air Portugal', 1945, 'Portugal').
company('RYR', 'Ryanair', 1984, 'Ireland').
company('AFR', 'Société Air France, S.A.', 1933, 'France').
company('BAW', 'British Airways', 1974, 'United Kingdom').
```

O predicado *flight/6* contém informação relativa a voos diários, contendo a designação do voo (que é única), os identificadores dos aeroportos de origem e destino, a hora de partida (em formato inteiro, concatenando horas e minutos), a duração do voo (em minutos), e o identificador da companhia aérea que opera o voo. Considere todos os tempos no mesmo fuso horário. Todos os voos indicados têm partida e chegada no mesmo dia.

```
%flight(Designation, Origin, Destination, DepartureTime, Duration, Company).
flight('TP1923', 'LPPR', 'LPPT', 1115, 55, 'TAP').
flight('TP1968', 'LPPT', 'LPPR', 2235, 55, 'TAP').
flight('TP842', 'LPPT', 'LIRF', 1450, 195, 'TAP').
flight('TP843', 'LIRF', 'LPPT', 1935, 195, 'TAP').
flight('FR5483', 'LPPR', 'LEMD', 630, 105, 'RYR').
flight('FR5484', 'LEMD', 'LPPR', 1935, 105, 'RYR').
flight('AF1024', 'LFPG', 'LPPT', 940, 155, 'AFR').
flight('AF1025', 'LPPT', 'LFPG', 1310, 155, 'AFR').
```

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 ***short(+Flight)*** que recebe o identificador de um voo e sucede se o voo for curto. Um voo é considerado curto se tiver uma duração inferior a 1h30.

Exemplos:

```
| ?- short('TP842').
no
| ?- short('TP1923').
yes
```

Pergunta 2

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***shorter(+Flight1, +Flight2, -ShorterFlight)*** que recebe os identificadores de dois voos, e retorna no terceiro argumento o identificador do voo com menor duração. Se os dois voos tiverem a mesma duração, o predicado deve falhar.

Exemplos:

```
| ?- shorter('TP842', 'TP1923', Shorter).
Shorter = 'TP1923' ? ;
no
| ?- shorter('TP843', 'TP842', Shorter).
no
```

Pergunta 3

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***arrivalTime(+Flight, -ArrivalTime)***, que recebe como parâmetro um voo, e determina a hora de chegada, no mesmo formato em que se encontra a hora de partida.

Exemplos:

```
 | ?- arrivalTime('TP1923', Arrival).
Arrival = 1210 ? ;
no
| ?- arrivalTime('TP843', Arrival).
Arrival = 2250 ? ;
no
```

Pergunta 4

Respondida Pontuou 1,400 de 1,500 Destacar pergunta

Implemente o predicado ***countries(+Company, -ListOfCountries)*** que recebe o nome de uma companhia aérea e retorna a lista de todos os países onde essa companhia opera. Considera-se que uma companhia opera num determinado país se operar algum voo de/para esse país. Note que a lista não deve conter países repetidos.

Sugestão: Implemente um predicado que suceda caso uma companhia opere num determinado país.

Exemplos:

```
| ?- countries('TAP', List).
List = ['Portugal','Italy'] ? ;
no
| ?- countries('AAL', List).
List = [] ;
no
```

Pergunta 5

Respondida Pontuou 1,200 de 2,000 Destacar pergunta

De forma a determinar voos com escalas, é importante determinar possibilidade de ligação entre voos. Implemente o predicado ***pairableFlights/0***, que imprime uma lista de voos emparelháveis, no formato *Aeroporto - VooChegada \ VooPartida*. Entende-se por voos emparelháveis pares de voos em que o segundo voo sai do aeroporto onde chegou o primeiro, com uma diferença temporal entre 30 e 90 minutos. Note que o predicado sucede sempre.

Exemplos:

```
| ?- pairableFlights.
LPPT - AF1024 \ AF1025
LIRF - TP842 \ TP843
LPPT - TP1923 \ AF1025
yes
```

Pergunta 6

Respondida Pontuou 0,200 de 2,000 Destacar pergunta

Você adora viajar de avião, e deseja fazer uma viagem itinerante por vários países. Pretende-se saber quantos dias são necessários para completar a viagem. Considere que os voos indicados nos factos ***flight/6*** se repetem diariamente. Implemente o predicado ***tripDays(+Trip, +Time, -FlightTimes, -Days)***, que recebe uma lista com os países a visitar, por ordem, e a hora a partir da qual se pretende começar a jornada, e que devolve as horas dos voos a apanhar bem como o número de dias necessários para completar a viagem. Como o seu objetivo é a viagem (e não visitar o país), considere que lhe bastam 30 minutos entre a hora de chegada a um país e a hora de partida desse mesmo país.

Exemplos:

```
| ?- tripDays(['Portugal', 'Italy', 'Portugal', 'Spain', 'Portugal', 'France', 'Portugal'], 0, L, N).
L = [1450, 1935, 630, 1935, 1310, 940],
N = 4
| ?- tripDays(['Portugal', 'France', 'Portugal', 'Italy', 'Portugal', 'Spain', 'Portugal'], 0, L, N).
L = [1310, 940, 1450, 1935, 630, 1935],
N = 3
| ?- tripDays(['Portugal', 'Spain', 'Italy'], 0, L, N).
no
```

Informação

Destacar pergunta

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

Pergunta 7

Respondida Pontuou 1,500 de 1,500 Destacar pergunta

Implemente o predicado ***avgFlightLengthFromAirport(+Airport, -AvgLength)***, que determina a duração média (em minutos) dos voos que saem do aeroporto *Airport*.

Exemplos:

```
| ?- avgFlightLengthFromAirport('LPPR', Avg).
Avg = 80.0 ? ;
no
```

Pergunta 8

Respondida Pontuou 1,500 de 2,000 Destacar pergunta

Implemente o predicado ***mostInternational(-ListOfCompanies)*** que devolve uma lista com a(s) companhia(s) aérea(s) mais internacionais, isto é, aquelas que operam voos de e para um maior número de países.

Exemplos:

```
| ?- mostInternational(Companies).
Companies = ['TAP', 'RYR', 'AFR'] ? ;
no
```

Informação

Destacar pergunta

O predicado ***make_pairs(+L,+P,-S)*** permite emparelhar elementos de uma lista *L* com base num predicado *P* fornecido como segundo argumento, obtendo os pares em *S*:

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

make_pairs(L, P, [X-Y|Zs]) :-
    select(X, L, L2),
    select(Y, L2, L3),
    G =.. [P, X, Y], G,
    !,
    make_pairs(L3, P, Zs).

make_pairs([], _, []).
```

Pergunta 9

Respondida Pontuou 1,000 de 1,000 Destacar pergunta

Repare que o predicado dá resultados diferentes dependendo da ordem dos elementos na lista fornecida como primeiro argumento:

```
dif_max_2(X,Y) :- X < Y, X >= Y-2.

| ?- make_pairs([1,2,3,5],dif_max_2,S).
S = [1-2,3-5] ?
yes
| ?- make_pairs([1,3,2,5],dif_max_2,S).
no
```

Altere o predicado de forma a que este obtenha soluções independentemente da ordem dos elementos na lista.

Pergunta 10

Respondida Pontuou 1,000 de 1,000 Destacar pergunta

O predicado só funciona com listas de comprimento par, e em que todos os elementos são emparelháveis:

```
| ?- make_pairs([1,2,3,7],dif_max_2,S).
no
| ?- make_pairs([1,2,3,5,6],dif_max_2,S).
no
```

Altere o predicado de forma a que funcione também para casos como os apresentados, ignorando os elementos que não podem ser emparelhados.

```
| ?- make_pairs([1,2,3,7],dif_max_2,S).
S = [1-2] ?
```

```
| ?- make_pairs([1,2,3,5,6],dif_max_2,S).
S = [1-2,3-5] ?
```

```
| ?- make_pairs([1,3,4,5,2,6],dif_max_2,S).
S = [1-3,4-5] ?
```

```
make_pairs(L, P, [X-Y|Zs]) :-
   select(X, L, L2),
   select(Y, L2, L3),
   G =.. [P, X, Y], G,
   make_pairs(L3, P, Zs).

make_pairs(L, P, Zs) :-
   select(_X, L, L2),
   select(_Y, L2, L3),
      make_pairs(L3, P, Zs).
```

Pergunta 11

Respondida Pontuou 1,900 de 2,000 Destacar pergunta

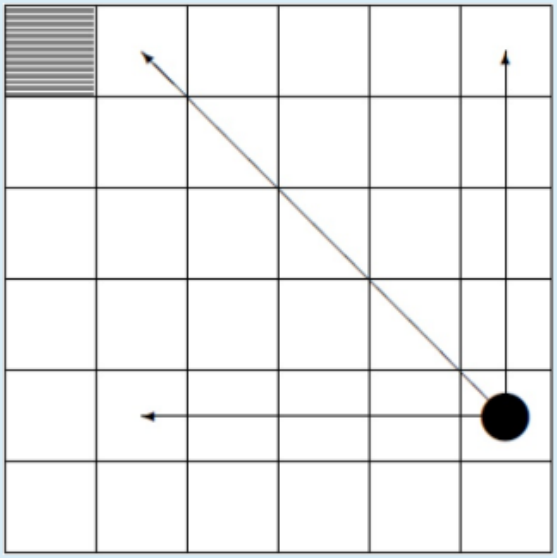
Com base no predicado ***make_pairs/3***, implemente o predicado ***make_max_pairs(+L,+P,-S)***, que obtém em *S* uma lista com o número máximo de pares possível.

```
| ?- make_max_pairs([1,3,4,5,2,6],dif_max_2,S).
S = [1-3,5-6,2-4]
```

Pergunta 12

Não respondida Pontuação 3,000 Destacar pergunta

O Whitoff é um jogo para dois jogadores, jogado num tabuleiro quadrado. Consiste em movimentar um disco um qualquer número de casas na horizontal, vertical ou diagonal, mas sempre na direção do canto superior esquerdo. Inicialmente, o jogador 1 coloca o disco numa qualquer posição. Seguidamente, começando pelo jogador 2, os jogadores vão movimentando o disco à vez. Vence o jogo quem colocar o disco no canto superior esquerdo.



Se o jogador 1 colocar o disco numa de um conjunto de posições específicas, consegue sempre ganhar o jogo. Por exemplo, a casa (1, 1) é uma dessas casas, e a casa (2, 3) é outra.

Construa um programa em Prolog cujo predicado principal ***whitoff(+N,-W)*** recebe a dimensão do tabuleiro e devolve a lista de casas vencedoras. Será valorizada uma implementação eficiente, isto é, que obtenha soluções para dimensões grandes (p. ex., 50) em menos de 1 segundo.

```
| ?- whitoff(3,L).
L = [(1,1),(3,2),(2,3)]

| ?- whitoff(7,L).
L = [(1,1),(3,2),(2,3),(6,4),(4,6)]

| ?- whitoff(10,L).
L = [(1,1),(3,2),(2,3),(6,4),(4,6),(8,5),(5,8)]
```