Ficha de Problemas da disciplina de Algoritmia Avançada do 3º ano da Licenciatura em Engenharia Informática da Instituto Superior de Engenharia do Porto

Ficha nº 3

1. Considere o seguinte programa e tente responder sem utilizar o computador:

```
mulher(maria,30).
mulher(rosa,15).
mulher(teresa,21).

homem(pedro,18).
homem(tiago,43).

par(X,Y):-  homem(X,IX),
            mulher(Y,IY).

mostra_par:- par(X,Y), write(X), write(' '), write(Y), nl,
             fail; true.
```

a) Quantos pares seriam apresentados após a chamada do predicado mostra_par? Seriam apresentados 6 pares.

b) Quantos pares seriam apresentados se o predicado fosse alterado das formas seguintes?

1) `par(X,Y):- homem(X,IX), mulher(Y,IY), !.` 1 par.

2) `par(X,Y):- homem(X,IX), !, mulher(Y,IY).` 3 pares.

3) `par(X,Y):- !, homem(X,IX), mulher(Y,IY).` 6 pares.

c) Qual é o funcionamento da construção 'fail; true' ? Esta condição força a apresentar todas as combinações possíveis pelo predicado par(X,Y).

d) Adicione o seguinte predicado ao programa anterior. Este predicado permite escolher pares cuja soma das idades seja menor ou igual ao valor MAX Por exemplo: escolhe_par(X,Y,43,S).

```
escolhe_par(X,Y,MAX,SOM):-
                homem(X,IX),
                mulher(Y,IY),
                SOM is IX+IY,
                SOM =< MAX.
```

Modifique o predicado para que este termine a execução no caso de o ulizador introduzir query cuja MAX seja inferior a 36.Por exemplo: `escolhe_par(X,Y,30,S).`

```
escolhe_par(X,Y,MAX,SOM):-
  MAX >= 36,
  homem(X,IX),
  mulher(Y,IY),
  SOM is IX+IY,
  SOM =< MAX.
```

2. Considere a seguinte base de conhecimento:

```
cor(céu, azul).
cor(céu, cinzento).
cor(céu, preto).
cor(mar, azul).
cor(mar, verde).
cor(mar, cinzento).
cor(via, cinzenta).
cor(via, castanha).
cor(via, preta).

transporte(céu,avião).
transporte(céu,helicóptero).
transporte(céu,foguete).
transporte(mar,navio).
transporte(mar,lancha).
transporte(via,carro).
transporte(via,camião).
transporte(via,mota).
transporte(via,autocarro).
```

1) Implemente um predicado que simule a negação da chamada de outro predicado com todos os argumentos instanciados.

Por exemplo, a chamada a nega(cor(céu,azul)) **falha enquanto que a chamada a** nega(transporte(mar,carro)) **tem sucesso.**

```
nega(X):- call(X),!,fail.
nega(_).
```

2) Implemente um predicado *se_então_senão* **que tenha 3 argumentos. O primeiro é o predicado de teste, o segundo é a consequência que ocorrerá se o predicado de teste tiver sucesso e o terceiro é a consequência caso o predicado de teste falhe.**

Por exemplo, a chamada:

```
?- se_então_senão(cor(mar,verde),
                  (write(ok),nl,write('...'),nl),
                  (write(falhou),nl,write('...'),nl)).
```

resulta em:

```
ok
...
yes
```

enquanto que a chamada:

```
?- se_então_senão(cor(mar,vermelho),
                  (write(ok),nl,write('...'),nl),
                  (write(falhou),nl,write('...'),nl)).
```

resulta em:

```
falhou
...
yes
```

3) Implemente um predicado *se_então* que tenha 2 argumentos. O primeiro é o predicado de teste e o segundo é a consequência que ocorrerá se o predicado de teste tiver sucesso. Independentemente do sucesso ou insucesso do predicado de teste, o *se_então* deverá ter sempre sucesso.

4) Implemente um predicado *questão(Valor)* que peça ao utilizador um atributo (cor ou transporte) e um objecto (céu, mar ou via) e pergunte ao utilizador "Qual o valor do/da <atributo> do/da <objecto>?" (por exemplo: "Qual o valor do/da cor do/da mar?". O mesmo utilizador responde a essa questão. Se a resposta for válida o predicado tem sucesso, caso contrário volta a ser posta a mesma questão ao utilizador, até que este dê uma resposta acertada.

Por exemplo:
?- questão(Valor).
Qual é o atributo? cor.
Qual é o objecto? mar.
Qual o valor do/da cor do/da mar? vermelho.
Qual o valor do/da cor do/da mar? castanho.
Qual o valor do/da cor do/da mar? branco.
Qual o valor do/da cor do/da mar? verde.

Valor=verde

Nota: o operador =.. , usado na forma *Predicado=..Lista*, converte bidireccionalmente um predicado numa lista em que o primeiro elemento da lista é o nome do functor e os outros elementos da lista são os vários argumentos do predicado.