Lista de Exercícios 13
Programação Lógica: Prolog

Exercício 01) Verifique se os seguintes termos podem ser unificados, se for possível, informe o resultado da unificação.

a) `p(X, 3)` e `q(2, 3)`
b) `p(X, 3)` e `p(2, 3)`
c) `p(x, 3)` e `p(2, 3)`
d) `p(X, 3)` e `p(Y, Y)`
e) `filho(carlos, maria)` e `filho(Y, Y)`
f) `[1,2 | X]` e `[1, 2, 7, 3, 4]`
g) `[1,2 | X]` e `[1, 2]`

Exercício 02) Considere o seguinte programa em Prolog com uma consulta.

```
01: gosta(maria, peixe).
02: gosta(pedro, vinho).
03: gosta(maria, vinho).
04: gosta(pedro, maria).
```

```
?- gosta(maria, X), gosta(pedro, X).
X = vinho .
```

Para essa consulta, mostre as árvores de execução do algoritmo de unificação quando:

a) Um termo não pôde ser unificado, logo antes de iniciar o *backtracking*.
b) A consulta foi unificada com sucesso e retornou a resposta.

Exercício 03) Escreva predicados Prolog para cada um dos problemas a seguir. Não se pode usar nenhum predicado pré-definidos da biblioteca de Prolog ou qualquer outra de terceiros a não ser aqueles vistos nos slides, mas é permitido reimplementar predicados auxiliares caso julgue necessário.

a) Obter o número de elementos de uma lista.

```
?- nelementos([1, 2, 3], S).
S = 3 .
```

b) Obter o maior valor de uma lista de inteiros.

```
?- maior([4, 5, 2, 3, 1], M).
M = 5 .
```

c) Obter o valor médio de uma lista de inteiros.

```
?- medio([4, 5, 2, 3, 1], M).
M = 3.0 .
```

d) Inserir um elemento no fim da lista.

```
?- inserirfim(4, [1, 2, 3], L).
L = [1,2,3,4] .
```

e) Obter o último elemento de uma lista.

```
?- ultimo([1, 2, 3, 4], U).
U = 4 .
```

f) Verificar se um elemento X é adjacente a um elemento Y.

```
?- adjacente(3, 4, [1, 2, 3, 4, 5, 6]).
true .
```

g) Gerar uma lista com os elementos de uma faixa (inclusive).

```
?- gerar(5, 10, L).
L = [5, 6, 7, 8, 9, 10] .
```

h) Reverter uma lista. Dica: use o predicado concatenar.

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

i) Incrementar em uma unidade cada elemento de uma lista de inteiros.

```
?- incrementar([5, 6, 7, 8], L).
L = [6, 7, 8, 9] .
```

j) Linearizar uma lista de inteiros. Dica: use o predicado concatenar.

```
?- linearizar([[1,2,3], [4,5], [6], [7,8]], L).
L = [1, 2, 3, 4, 5, 6, 7, 8] .
```

k) Compactar uma lista de elementos consecutivos.

```
?- compactar([a,a,a,a,b,c,c,a,a,d,e,e,e,e], L).
L = [[4,a], [1,b], [2,c], [2,a], [1,d], [4,e]] .
```

l) Remover de uma lista um elemento todas as suas ocorrências.

```
?- remover(3, [1,3,2,3,4], L).
L = [1, 2, 4] .
```

m) Rotacionar uma lista uma posição.

```
?- rotacionar([1, 2, 3, 4, 5], L).
L = [2, 3, 4, 5, 1] .
```

n) Rotacionar uma lista *n* posições.

```
?- rotacionarn(2, [1, 2, 3, 4, 5], L).
L = [3, 4, 5, 1, 2] .
```

o) Ordenar uma lista de inteiros.

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

Exercício 04) Escreva uma descrição Prolog de sua árvore genealógica retrocedendo até seus avós e incluindo todos os descendentes. Você deve se basear somente nos fatos: `pai` e `mae`. Em seguida, escreva um conjunto de regras para as relações familiares: `avos`, `pais`, `irmaos`, `tios` e `primos`. Dica: use a consulta `findall(Object,Goal,List)` que produz uma lista (`List`) de todos os objetos (`Object`) que satisfazem a meta (`Goal`). Por exemplo, para obter os nomes de todos os avôs de Carlos.

```
?- avos(carlos, R).
R = [joao, jose, maria, ana] .
```

Exercício 05) Considere o seguinte banco de dados com dados de clientes de uma determinada loja que vende smartphones**.**

Código	Nome	Idade	Profissão	Produto
01	Gabriel	21	Estudante	Iphone 4s por R$1300,00, Galaxy S3 por R$1400,00
02	Bruna	30	Médico	Iphone 5s por R$2400,00, Capa Iphone 5s por R$80,00
03	Rafaela	25	Arquiteto	Iphone 5s por R$2500,00, Galaxy S4 por R$2300,00
04	Victor	39	Advogado	BlackBerry Z10 por R$1800,00
05	Beatriz	18	Estudante	Iphone 4s por R$1300,00, Iphone 4 por R$1000,00
06	João	28	Engenheiro	Nokia Lumia por R$1700,00

a) Para cada uma das linhas do banco de dados, crie fatos em Prolog.
b) Defina cláusulas para consultar as seguintes informações:
   i. `listar_clientes(L)`**:** obter a lista (`L`) de todos os nomes dos clientes.
   ii. `listar_dados_cliente(Codigo, Dados)`: obter todos os dados do cliente (`Dados`) para um determinado cliente (`Codigo`). Não precisa retornar nos dados o código do cliente.
   iii. `listar_smartphones(L)`: obter os nomes não repetidos de todos os smartphones vendidos pela loja em uma lista (`L`).
   iv. `listar_estudantes(L)`: obter o nome de todos os clientes da loja que sejam estudantes em uma lista (`L`).
   v. `preco_medio(X)`: obter o preço médio (`X`) de todos os smartphones vendidos na loja.