Segunda lista de exercícios

Exercício 1:

Qual é o resultado das seguintes consultas em Prolog?

```
?- forall(member(X,[1,2,3]),write(X)).
?- forall(between(1,5,I),(write(I*I),write(' '))).
?- forall(between(10,20,I),write(I:' ')).
```

Exercício 2:

Considere o seguinte programa Prolog:

```
a:-a(0).
a(X):- X>10,!.
a(X):- write(X),write(' '), X1 is X+1,a(X1).
```

Qual será o resultado da seguinte consulta Prolog:

```
?- a.
```

Exercício 3:

Faça um predicado que gere a pirâmide abaixo. Use o predicado wN/1.

```
wN(0):-write(0),!.
wN(N):-write(N),N1 is N-1, wN(N1),write(N).
```

Exemplo de uso:

```
?- xxx(3).
   0
  101
 21012
3210123
```

Exercício 4:

Usando um acumulador, e somente as operações (+)(-)(*), desenvolva um predicado Prolog para calcular X elevado a Y. Assuma X e Y inteiros.

Exercício 5:

Dados os fatos:

```
d(0).
d(1).
```

Qual é o resultado da seguinte consulta Prolog:

```
?- findall([A,B],(d(A),d(B)),L).
```

Exercício 6:

Defina o predicado palindromo/1, que é verdadeiro se a lista é um palíndromo, por exemplo, `[a,b,c,d,c,b,a].`

Exercício 7:

Defina um predicado metIguais/1, que é verdadeiro se uma lista é formada por duas metades iguais. Use o append. Seguem dois exemplos de uso.

```
?-metIguais([a,b,c, a,b,c]).
Yes
?-metIguais([a,b,c, a,b,d]).
No
```

Exercício 8:

Faça um predicado insOrd/3, que insere um elemento numa lista mantendo-a ordenada. Faça duas regras: uma base e uma recursiva.

```
?-insOrd(4,[2,3,5,7],L).
L=[2,3,4,5,7] Yes
```

Exercício 9:

Faça um predicado que particiona/3 uma lista em duas, de tamanho igual se o número de elementos for par, senão uma delas terá um elemento a mais. Tire dois elementos de uma lista (se possível) e ponha cada um em uma lista resultado.

Exercício 10:

Faça o predicado merge/3, que junta duas listas ordenadas em uma terceira, mantendo a ordem. Como segue:

```
?- merge([a,b,b,k,z], [c,m,n,o], X).
X=[a,b,b,c,k,,m,n,o,z], yes
```

Exercício 11:

O que está errado no programa abaixo? Rode-o com trace, para:

```
?- max(4,3,M) e
?- max(3,4,M)
max(X,Y,M):-!, X>Y, M=X.
max(X,Y,M):-!, X<=Y, M=Y.
```

Exercício 12:

O que acontece com o predicado p, abaixo, quando o b é executado?

```
a. b. p:-!,a. p:-b.
```

Exercício 13:

Usando findall, defina e teste os predicados pred1/2, pred2/2 e pred3/2 que modificam uma lista, conforme ilustrado nos seguintes exemplos:

```
?- pred1([a,b,c,d,e],L).
L = [[a],[b],[c],[d],[e]]
?- pred2([a,b,c,d,e],L).
L = [pred(a,a),pred(b,b),pred(c,c),pred(d,d),pred(e,e)]
?- pred3([a,b,c,d,e],L).
L = [[element,a],[element,b],[element,c],[element,d],[element,e]]
```

Exercício 14:

Considere o seguinte problema: *“Há dois jarros com capacidades de 3 e 4 litros, respectivamente. Nenhum dos jarros contém qualquer medida ou escala, de modo que só se pode saber o conteúdo exato quando eles estão cheios. Sabendo-se que podemos encher ou esvaziar um jarro, bem como transferir água de um jarro para outro, encontre uma sequência de passos que deixe o jarro de 4 litros com exatamente 2 litros de água”*. Considere que o estado inicial pode ser representado pela lista [0,0], indicando que os jarros de 3 e 4 litros estão vazios inicialmente e a meta é [_,2]. Complete os predicados **transforma** abaixo para descrever todas as transformações possíveis de estados desse problema.

```
transforma('encher o jarro 1', [X,Y], [3,Y]) :- X < 3.
transforma('enchar o jarro 2', ...
transforma('esvaziar o jarro 1', ...
transforma('esvaziar o jarro 2', ...
transforma('transferir do jarro 1 para o 2', ...
transforma('transferir do jarro 2 para o 1', ...
%--- considerando que ainda restara agua no jarro de origem
transforma('transferir do jarro 1 para o 2', ...
transforma('transferir do jarro 2 para o 1', ...
```