01. [Sebesta, 2000] Quais são os três principais usos da lógica simbólica na lógica formal?

02. [Sebesta, 2000] Quais são as duas partes de um termo composto?

03. [Sebesta, 2000] Qual é a forma geral de uma proposição na forma clausal?

04. [Sebesta, 2000] Dê uma descrição geral (não rigorosa) de resolução e unificação?

05. [Sebesta, 2000] Quais são as formas das cláusulas de Horn?

06. [Sebesta, 2000] Qual é o conceito básico da semântica declarativa?

07. [Sebesta, 2000] Quais são as três formas de um termo Prolog?

08. [Sebesta, 2000] Qual é a forma sintática e o uso das instruções relativas a fatos e a regras no Prolog?

09. [Sebesta, 2000] Explique as duas abordagens para comparar metas com fatos em um banco de dados.

10. [Sebesta, 2000] Explique a diferença entre uma busca primeiramente pela profundidade e uma primeiramente pela largura quando se discute como metas múltiplas são satisfeitas.

11. [Sebesta, 2000] Explique como o *backtracking* funciona no Prolog.

12. [Sebesta, 2000] Explique o que há de errado com a instrução Prolog `K is K + 1.`

13. [Sebesta, 2000] Quais são as duas maneiras pelas quais um programador Prolog pode controlar a ordem de comparação de padrões durante a resolução?

14. [Sebesta, 2000] Explique a estratégia de programação gerar-e-testar do Prolog.

15. [Sebesta, 2000] Explique a pressuposição de mundo fechado usada pelo Prolog. Por que ela é uma limitação?

16. [Sebesta, 2000] Explique o problema da negação com o Prolog. Por que ele é uma limitação?

17. [Sebesta, 2000] Explique a ligação entre demonstração automática de teoremas e o processo de inferência do Prolog.

18. [Sebesta, 2000] Explique a diferença entre linguagens baseadas em procedimentos e não baseadas em procedimentos.

19. [Sebesta, 2000] Explique porque os sistemas Prolog devem fazer *backtracking*.

20. [Sebesta, 2000] Qual é a relação entre resolução e unificação no Prolog?

21. [Sebesta, 2000] Compare o conceito de tipificação de dados da Ada com o do Prolog.

22. [Sebesta, 2000] Descreva como uma máquina com múltiplos processadores poderia ser usada para implementar a resolução. O Prolog poderia, como atualmente é definido, usar esse método?

23. [Sebesta, 2000] Escreva uma descrição Prolog de sua árvore familiar (baseado somente em fatos) retrocedendo até seus avós e incluindo todos os descendentes. Certifique-se de incluir todas as relações.

24. [Sebesta, 2000] Escreva um conjunto de regras para relações familiares, incluindo todas as relações de avós ao longo de duas gerações. Agora, adicione-as aos fatos do Problema 23 e elimine tantos fatos quantos puder.

25. [Sebesta, 2000] Escreva um programa Prolog que seja bem-sucedido se a interseção de dois parâmetros de lista de dados estiver vazia.

26. [Sebesta, 2000] Escreva um programa Prolog que retorne uma lista que contém a união dos elementos de duas listas dadas.

27. [Sebesta, 2000] Escreva um programa Prolog que retorne o último elemento de uma lista dada.

28. [Sebesta, 2000] Explique duas maneiras pelas quais as capacidades do processamento de lista da Scheme e do Prolog são similares.

29. [Sebesta, 2000] De que maneira as capacidades de processamento de lista da Scheme e do Prolog são diferentes?

30. Desenvolver um programa em Prolog que inverta os elementos de uma lista, como no exemplo a seguir.

```
?- reverso([a, b, c, d], Lista), write(Lista), nl.
[d,c,b,a]
Yes.
```

31. Desenvolver um programa em Prolog que apresente a somatória dos elementos contidos em uma lista, como no exemplo a seguir.

```
?- somatorio([1, 2, 3, 4, 5], N), write(N), nl.
15
Yes.
```

32. Desenvolver um programa em Prolog que apresente a média dos elementos contidos em uma lista, como no exemplo a seguir.

```
?- media([1, 2, 3, 4, 5], M), write(M), nl.
3
Yes.
```

33. Desenvolver um programa em Prolog que apresente os elementos contidos em uma lista em dobro, como no exemplo a seguir.

```
?- dobro([1, 2, 3, 4, 5], Lista), write(Lista), nl.
[2,4,6,8,10]
Yes.
```

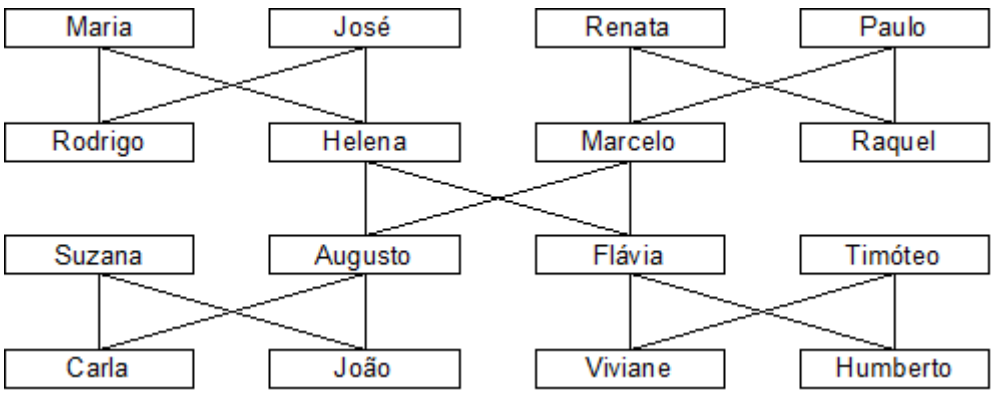
34. Desenvolver um programa em Prolog que apresente a quantidade de vezes que um determinado elemento aparece dentro de uma lista, como no exemplo a seguir.

```
?- ocorrencia(b, [a, b, c, d, e], N), write(N), nl.
1
Yes.
```

35. Desenvolver um programa em Prolog que apresente o fatorial de um número, como no exemplo a seguir.

```
?- fatorial(5, N), write(N), nl.
120
Yes.
```

36. Considerando a árvore genealógica e os fatos apresentados a seguir, apresente as seguintes regras em Prolog.




```
% Fatos de progenitor
progenitor('Maria', 'Rodrigo').
```

```
progenitor('Maria', 'Helena').
progenitor('José', 'Rodrigo').
progenitor('José', 'Helena').
progenitor('Renata', 'Marcelo').
progenitor('Renata', 'Raquel').
progenitor('Paulo', 'Marcelo').
progenitor('Paulo', 'Raquel').
progenitor('Helena', 'Augusto').
progenitor('Helena', 'Flávia').
progenitor('Marcelo', 'Augusto').
progenitor('Marcelo', 'Flávia').
progenitor('Suzana', 'Carla').
progenitor('Suzana', 'João').
progenitor('Augusto', 'Carla').
progenitor('Augusto', 'João').
progenitor('Flávia', 'Viviane').
progenitor('Flávia', 'Humberto').
progenitor('Timóteo', 'Viviane').
progenitor('Timóteo', 'Humberto').

% Fatos de masculino
masculino('José').
masculino('Paulo').
masculino('Rodrigo').
masculino('Marcelo').
masculino('Augusto').
masculino('Timóteo').
masculino('João').
masculino('Humberto').

% Fatos de feminino
feminino('Maria').
feminino('Renata').
feminino('Helena').
feminino('Raquel').
feminino('Suzana').
feminino('Flávia').
feminino('Carla').
feminino('Viviane').

% Regra pai
pai(Pai, Filho_a) :-

% Regra mãe
mae(Mae, Filho_a) :-

% Regra filho
filho(Filho, Pai_Mae) :-

% Regra filha
filha(Filha, Pai_Mae) :-

% Regra avô
avoo(Avo, Neto_a) :-

% Regra avó
avoa(Avo, Neto_a) :-

% Regra neto
neto(Neto, Avo_a) :-
```

```
% Regra neta
neta(Neta, Avo_a) :-

% Regra bisavô
bisavoo(Bisavo, Bisneto_a) :-

% Regra bisavó
bisavoa(Bisavo, Bisneto_a) :-

% Regra bisneto
bisneto(Bisneto, Bisavo_a) :-

% Regra bisneta
bisneta(Bisneta, Bisavo_a) :-

% Regra irmaos
irmaos(Irmao1, Irmao2) :-

% Regra irmão
irmao(Irmao, Irmao_a) :-

% Regra irmã
irma(Irma, Irmao_a) :-

% Regra tio
tio(Tio, Sobrinho_a) :-

% Regra tia
tia(Tia, Sobrinho_a) :-

% Regra sobrinho
sobrinho(Sobrinho, Tio_a) :-

% Regra sobrinha
sobrinha(Sobrinha, Tio_a) :-

% Regra primo
primo(Primo, Primo_a) :-

% Regra prima
prima(Prima, Primo_a) :-

% Regra casados
casados(Esposo, Esposa) :-
casados(Esposa, Esposo) :-

% Regra sogro
sogro(Sogro, Genro_Nora) :-

% Regra sogra
sogra(Sogra, Genro_Nora) :-

% Regra genro
genro(Genro, Sogro_a) :-

% Regra nora
nora(Nora, Sogro_a) :-

% Regra cunhado
cunhado(Cunhado, Conjugue) :-
```

```
% Regra cunhada
cunhada(Cunhada, Conjugue) :-
```

37. Desenvolver um programa em Prolog que apresente o Máximo Divisor Comum (MDC) entre dois números inteiros, como no exemplo a seguir.

```
?- mdc(18, 60, X), write(X), nl.
6
Yes.
```

38. Desenvolver um programa em Prolog que verifique se uma lista é palíndroma, ou seja, se a lista é igual a ela própria lida de trás para a frente, como no exemplo a seguir.

```
?- palindroma([a, b, c, b, a]), nl.
true
Yes.
```

39. Desenvolver um programa em Prolog que apresente o produto dos termos da série de Fibonacci. A série de Fibonacci é formada pela sequência `1, 1, 2, 3, 5, 8, 13, 21, 34, ...`. A série de Fibonacci é de grande importância matemática, e a lei básica é que a partir do terceiro termo, todos os termos são a soma dos dois últimos. O número de termos será fornecido pelo usuário, devendo ser um valor inteiro e positivo. Por exemplo, caso o número de termos fornecido pelo usuário seja `7`, o programa deverá apresentar como resposta o valor `3120`, ou seja, `1 * 1 * 2 * 3 * 5 * 8 * 13`, como no exemplo a seguir.

```
?- produto(7, X), write(X), nl.
3120
Yes.
```

40. Desenvolver um programa em Prolog que calcule o valor da série infinita

$$H = 1^1/1! - 2^2/2! + 3^3/3! - 4^4/4! + \ldots$$

O número de termos será fornecido pelo usuário, devendo ser um valor inteiro e positivo.

Por exemplo, caso o número de termos fornecido pelo usuário seja 5, o programa deverá apresentar como resposta o valor `18.875`, ou seja, $1^1/1! - 2^2/2! + 3^3/3! - 4^4/4! + 5^5/5!$.

Caso o usuário forneça um valor inválido para o número de termos, o programa deverá presentar como resposta o valor `-1`.

```
:- serieH(5, H), write(H), nl.
18.875
```