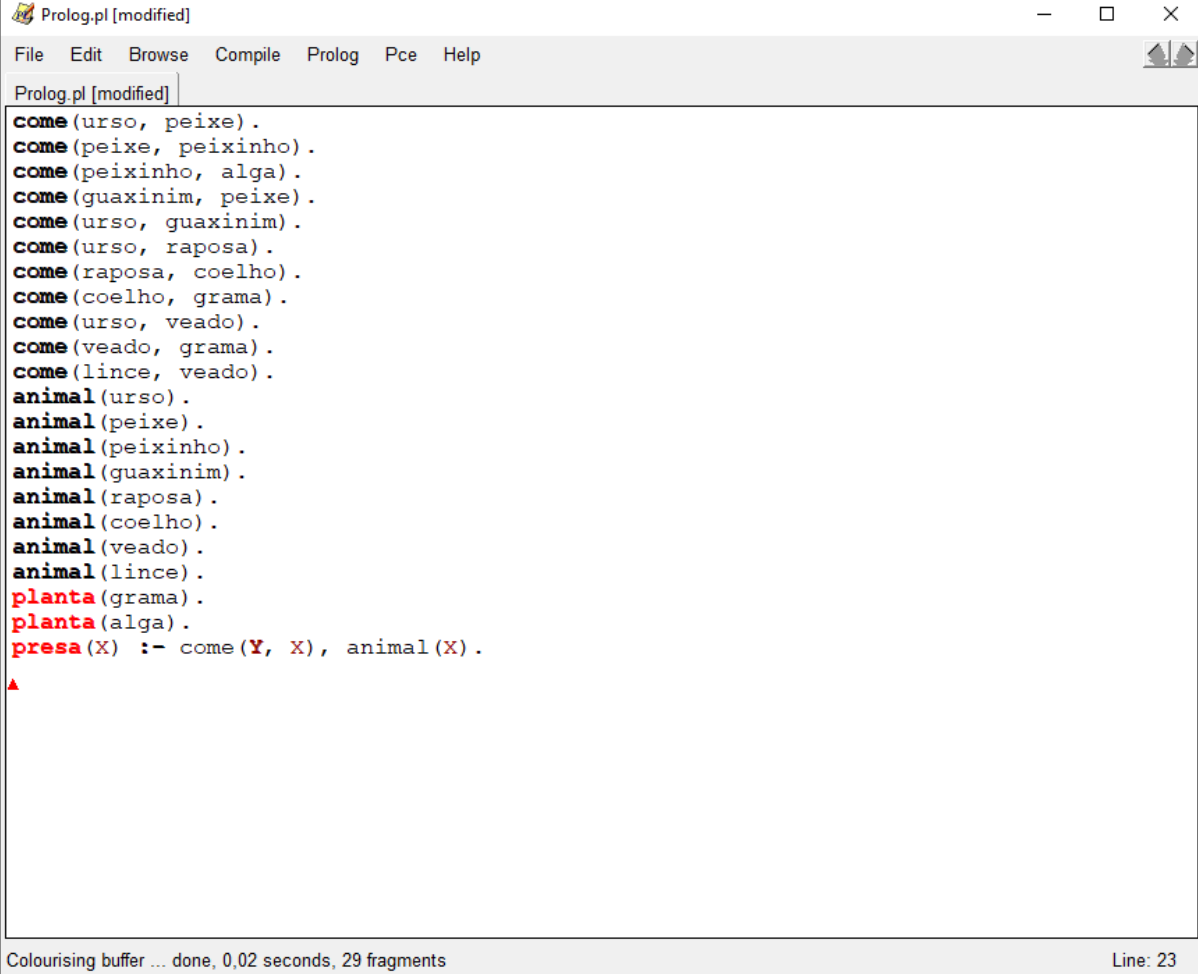


Banco de dados do Prolog:



The screenshot shows a window titled "Prolog.pl [modified]" with a menu bar (File, Edit, Browse, Compile, Prolog, Pce, Help) and a toolbar. The main text area contains the following Prolog code:

```
come(urso, peixe).
come(peixe, peixinho).
come(peixinho, alga).
come(guaxinim, peixe).
come(urso, guaxinim).
come(urso, raposa).
come(raposa, coelho).
come(coelho, grama).
come(urso, veado).
come(veado, grama).
come(lince, veado).
animal(urso).
animal(peixe).
animal(peixinho).
animal(guaxinim).
animal(raposa).
animal(coelho).
animal(veado).
animal(lince).
planta(grama).
planta(alga).
presa(X) :- come(Y, X), animal(X).
```

A red triangle cursor is positioned at the end of the last line. The status bar at the bottom indicates "Colourising buffer ... done, 0,02 seconds, 29 fragments" and "Line: 23".

A- Por que o Prolog é considerada uma definição recorrente ou recursiva?

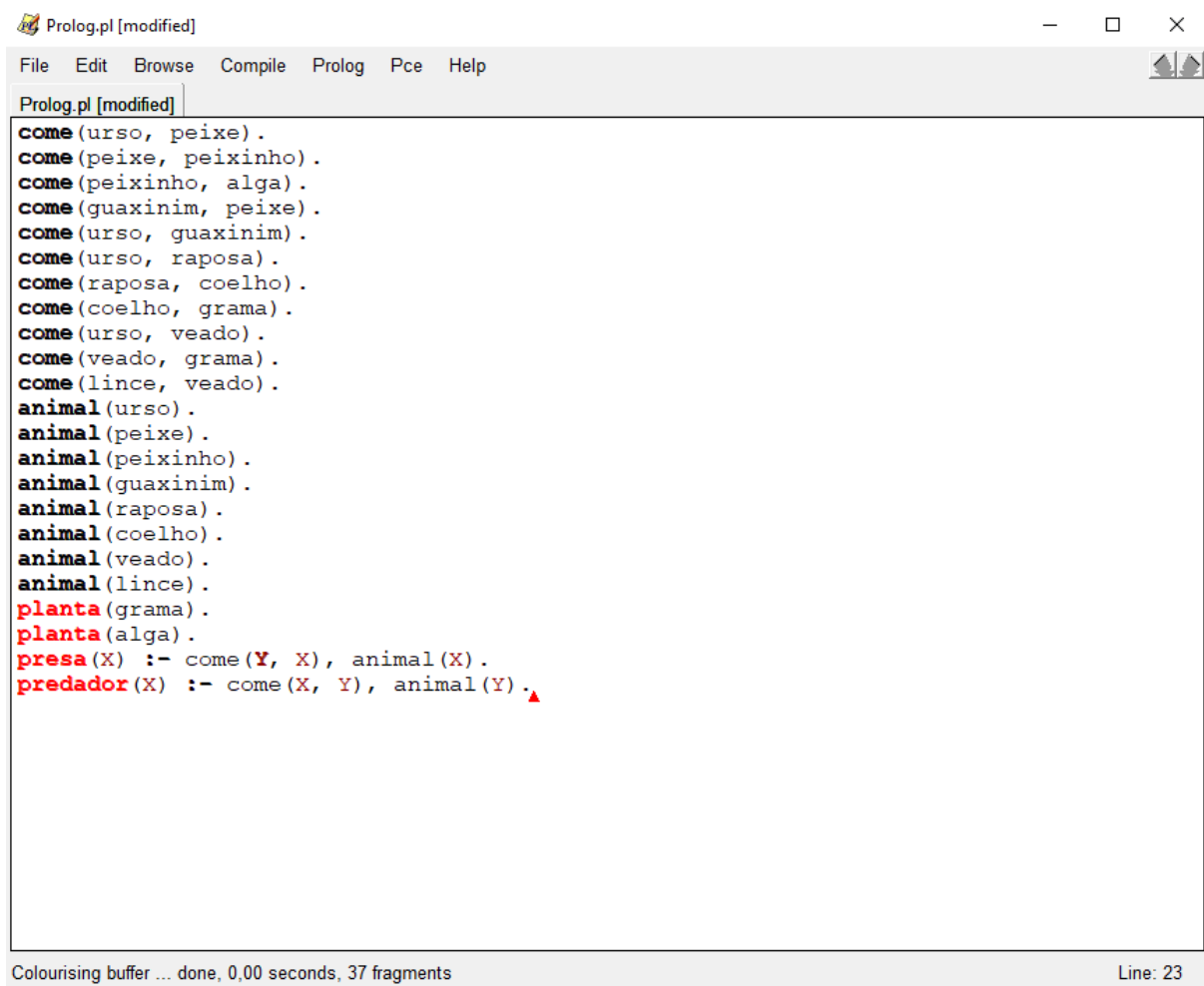
R: O Prolog é classificado como uma linguagem de programação que suporta recursão, pois permite que os desenvolvedores criem funções e procedimentos usando regras que podem se chamar novamente. Isso possibilita aos programadores escrever programas capazes de resolver problemas complexos de maneira eficiente.

B- Explique porque são denominados como fatos (fato 1 e fato 3) os itens acima.

R: Os elementos mencionados acima são conhecidos como fatos, pois são declarações que podem ser verdadeiras ou falsas. No contexto do Prolog, os fatos são utilizados para representar o conhecimento sobre o mundo.

No exemplo de diálogo apresentado, a afirmação “come(X, peixe)” indica que X se alimenta de peixe. Essa afirmação é verdadeira para ursos e guaxinins, mas falsa para outros animais como coelhos e veados.

C- Formule uma regra de Prolog que define o predicado *predador*.



```
Prolog.pl [modified]
File Edit Browse Compile Prolog Pce Help
Prolog.pl [modified]
come(urso, peixe).
come(peixe, peixinho).
come(peixinho, alga).
come(guaxinim, peixe).
come(urso, guaxinim).
come(urso, raposa).
come(raposa, coelho).
come(coelho, grama).
come(urso, veado).
come(veado, grama).
come(lince, veado).
animal(urso).
animal(peixe).
animal(peixinho).
animal(guaxinim).
animal(raposa).
animal(coelho).
animal(veado).
animal(lince).
planta(grama).
planta(alga).
presa(X) :- come(Y, X), animal(X).
predador(X) :- come(X, Y), animal(Y).
```

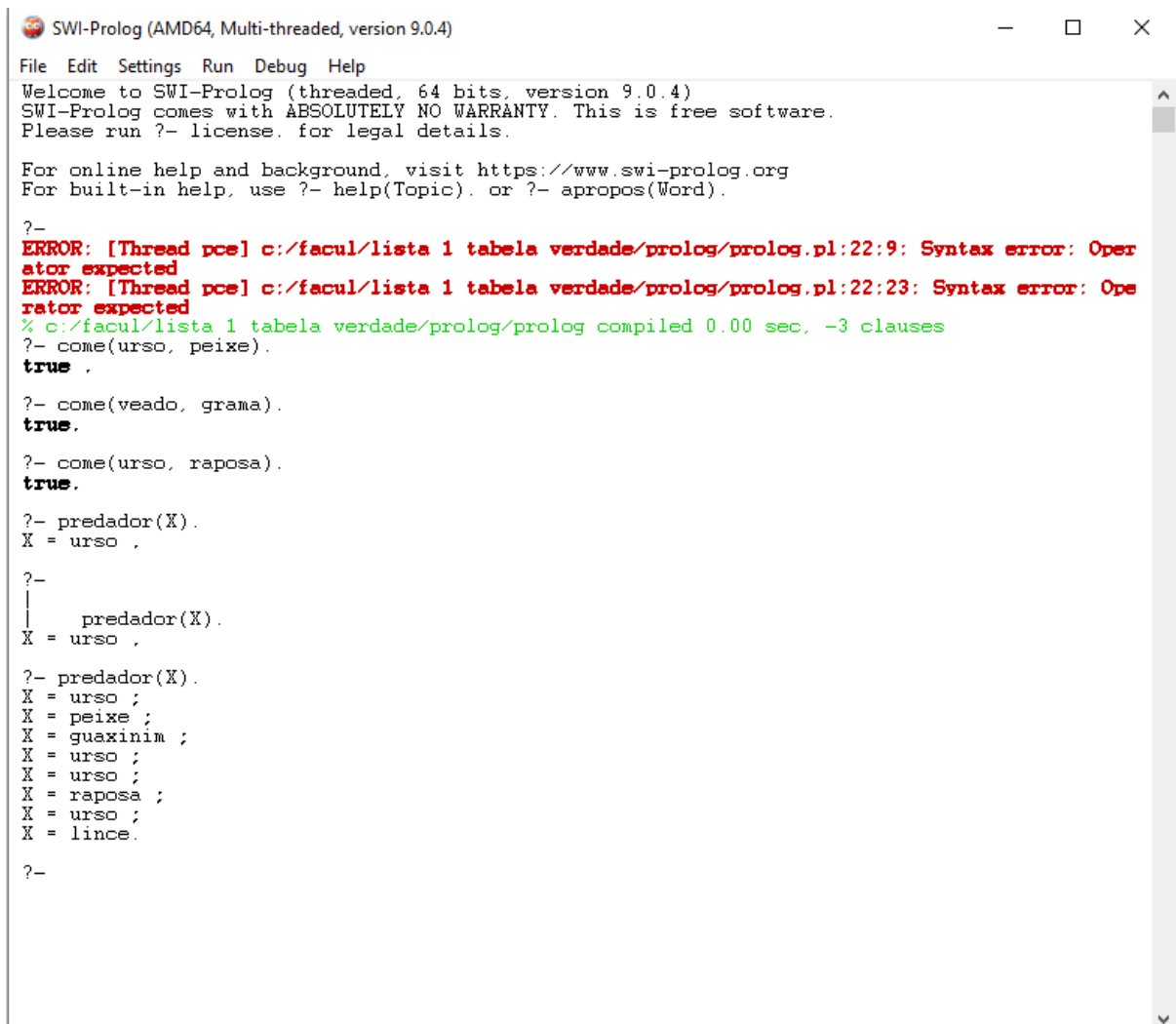
Colourising buffer ... done, 0,00 seconds, 37 fragments

Line: 23

D- Adicione essa regra ao banco de dados do Exemplo acima e diga qual seria a resposta à consulta.

?predador(X)

R:



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
ERROR: [Thread pce] c:/facul/lista 1 tabela verdade/prolog/prolog.pl:22:9: Syntax error: Oper
ator expected
ERROR: [Thread pce] c:/facul/lista 1 tabela verdade/prolog/prolog.pl:22:23: Syntax error: Ope
rator expected
% c:/facul/lista 1 tabela verdade/prolog/prolog compiled 0.00 sec, -3 clauses
?- come(urso, peixe).
true.

?- come(veado, grama).
true.

?- come(urso, raposa).
true.

?- predador(X).
X = urso.

?-
|
|   predador(X).
X = urso.

?- predador(X).
X = urso ;
X = peixe ;
X = guaxinim ;
X = urso ;
X = urso ;
X = raposa ;
X = urso ;
X = lince.

?-
```

E- Encontre os resultados da consulta em cada caso no Problema

?animal(lince)

?planta(guaxinim)

?come(urso, peixinho)

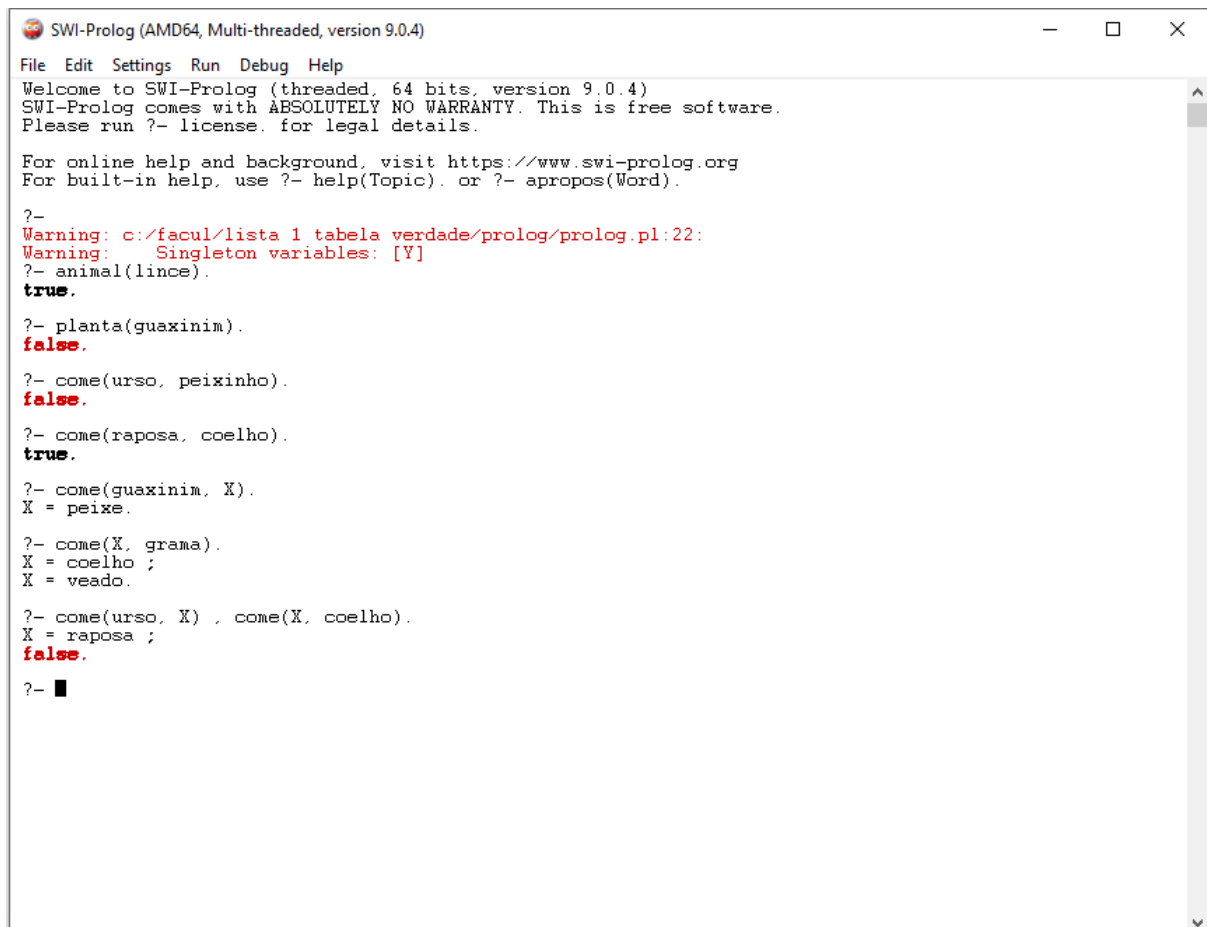
?come(raposa, coelho)

?come(guaxinim, X)

?come(X, grama)

?come(urso, X) e come(X, coelho)

R:



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
Warning: c:/facul/lista 1 tabela verdade/prolog/prolog.pl:22:
Warning: Singleton variables: [Y]
?- animal(lince).
true.

?- planta(guaxinim).
false.

?- come(urso, peixinho).
false.

?- come(raposa, coelho).
true.

?- come(guaxinim, X).
X = peixe.

?- come(X, grama).
X = coelho ;
X = veado.

?- come(urso, X) , come(X, coelho).
X = raposa ;
false.

?-
```

F- Escreva, usando conceitos de Prolog, o significado de um **busca em profundidade**.

R: Em Prolog, uma busca em profundidade é um algoritmo de pesquisa que percorre os nós de uma árvore de busca seguindo a ordem da profundidade. O algoritmo inicia no nó raiz e, a partir de cada nó, explora todos os seus filhos antes de passar para os filhos dos filhos e assim sucessivamente.

G- Responda: Por que os conceitos de Prolog estão relacionados com a **lógica de predicados**? Faça uma sistematização com o conceito da regra de *Modus Ponens*.

R: As ideias fundamentais do Prolog estão relacionadas à lógica de predicados, uma vez que ambas se baseiam na noção de que o conhecimento pode ser modelado como uma coleção de afirmações. Estas declarações são referidas no Prolog como fatos e regras. Na lógica de predicados, os fatos são afirmações verdadeiras ou falsas. Regras são afirmações que ligam um conjunto de fatos a outro conjunto de fatos. Os fatos são usados no Prolog para expressar conhecimento do mundo. As regras são usadas para ilustrar como o conhecimento pode ser aplicado para fazer inferências. A regra Modus Ponens é uma regra de inferência na lógica de predicados que afirma que se uma afirmação A é verdadeira e se A implica B, então B também deve ser verdadeiro.

H- Por fim, faça uma conclusão do seu trabalho, procure na literatura (atenção para buscas científicas como, por exemplo, o *Google Scholar*) estudos sobre a lógica de predicados e a linguagem Prolog em que vocês entendam como interessantes no referido trabalho. Lembre que esse é o momento de finalizar o que foi desenvolvido pelos autores do trabalho.

R: Em conclusão, é evidente que a lógica de predicados e a linguagem Prolog são recursos inestimáveis para abordar e resolver problemas complexos. Além disso, essas áreas continuam sendo campos ativos de pesquisa tanto na Inteligência Artificial quanto na Ciência da Computação. Para embasar essa afirmação, realizamos uma pesquisa na literatura possuindo como fontes do Google Scholar, e foi identificado diversos estudos que destacam a importância e as aplicações práticas dessas abordagens. Portanto, é seguro afirmar que a lógica de predicados e o Prolog permanecem na vanguarda da pesquisa e desenvolvimento nessas disciplinas.

