UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FEELT - FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA DE COMPUTAÇÃO

LUCAS ALBINO MARTINS

Matrícula: 12011ECP022

TRABALHO 04: 4ª. Lista de Exercícios: Lista de exercícios em Prolog.

Disciplina: Programação Lógica e Inteligência Artificial

Uberlândia
2020

Faça o exercício 4 e implemente em PROLOG, à sua escolha, pelo menos 6 exercícios da lista abaixo (o exercício 4 não conta!).
Lista de Exercícios
(Obs.: nos predicados "+" significa entrada e "-" significa saída)

1) Escreva um programa Prolog que faça o produto interno entre duas listas do mesmo tamanho, exemplo:

```
/* Escreva um programa Prolog que faça o produto interno entre duas lista
s do mesmo tamanho
*/

produto_interno(A,B,ProdInt):-
    produto_interno(A,B,0,ProdInt).

produto_interno([],[],Ac,Ac).
produto_interno([A|As],[B|Bs],Ac,ProdInt):-
    NAc is Ac + A*B,
    produto_interno(As,Bs,NAc,ProdInt).

/*
Executando o programa.
 produto_interno([1,2,-1],[-3,0,-5],X).
X = 2.
*/
```

4) Preencha a tabela abaixo com as soluções às questões Prolog:

Questões	Solução	True\false
01	?- [H | T] = [[[ ]]].	True
02	?- [H | T] = [[a, b]].	True
03	?- [H1, H2 | T] = [[a, b]].	False
04	?- [H1, H2 | T] = [[ ] | [a]].	True
05	?- [_ | T] = [a, b | [c]].	True
06	?- [H | _] = [[a] | b, c].	False
07	?- [[1], X | X] = [X, Y, Z].	True
08	?- [X | X] = [[ ]].	True
09	?- [X | X] = [[ ], [ ]].	False
10	?- [X, X] = [[ ], [ ]].	True
11	?- [X, Y,_| Z] = [1, 2, 3, 4].	True
12	?- [X | [_ ,Z]] = [1, 2, 3, 4] .	False
13	?- [X | [Y | Z]] = [1, 2, 3, 4].	True
14	?- [1 | [Y, X]] = [X, 2, 3].	False

15	?- 2 – 2 + C = A – B + 5	False
16	?- 3 + 0 = 4 - 1	False
17	?- 3 + 02 =:= 4 - 1	False
18	?- pai(joao,X) = pai(Y,maria)	True
19	?-pai(Joao,X) = pai(adao,ada)	True
20	?- 1 + 2 is 3	False

6) Seja o seguinte programa PROLOG

g([ ], [ ]).
g([H | T], [H | R]) :- 0 is H mod 2, g(T, R).
g([H | T], R) :- 1 is H mod 2, g(T, R).
Qual a resposta deste programa à questão g([2, 1, 5, 4], Z)?
?- g([2,1,5,4],Z).

Z = [2, 4] .

O que este programa faz?
Justifique sua resposta através de um *trace* da questão.

O programa por sua vez verifica em uma lista quais valores são divisíveis por 2. Quando o valor é divisível por 2 ele retorna imprimindo os valores da lista que são divisíveis por 2, caso contrário ele devolve a lista vazia quando não possui nenhum valor divisível por dois.

```
g([ ], [ ]).
g([ H | T ], [ H | R ]) :- 0 is H mod 2, g( T, R). % resultado dá divisao
 for zero ele salva na nova lista.
g([ H | T ], R) :- 1 is H mod 2, g( T, R). % resultado da divisão for 1 n
ão adiciona o valor a nova lista.

/*Teste da função.

?- g([4,8,12,20],Z).
Z = [4, 8, 12, 20] .

?- g([4,8,7,12,9,20],Z).
Z = [4, 8, 12, 20] .

*/
```

11) Escreva um predicado Prolog múltiplo/1 que recebe uma lista L e verifica se existe uma múltipla ocorrência de algum elemento. Exemplo: ?- múltiplo([a,b,c,b]).
true

```
/* Escreva um predicado Prolog múltiplo/1 que recebe uma lista L
   e verifica se existe uma múltipla ocorrência de algum elemento
*/

pertence(H,[H|_]).
pertence(X,[_|T]) :- pertence(X,T).

multiplo([H|T]) :- pertence(H,T).
multiplo([_|T]) :- multiplo(T).

/* Saída do programa
?- multiplo([1,2,3,4,5]).
false.

?- multiplo([1,2,3,1,4,5,3]).
true .

*/
```

12) Escreva um predicado Prolog meio/2 que, dada uma lista, retorna o elemento que o ocupa a posição do meio dessa lista. Se a lista tiver comprimento par o predicado deve falhar.
Exemplo: ?- meio([a,b,c,d,e],X).
X = c
true
?- meio([a,b,d,e],X).
False

```
/* Escreva um predicado Prolog meio/2 que, dada uma lista,
   retorna o elemento que o ocupa a posição do meio dessa lista.
   Se a lista tiver comprimento par o predicado deve falhar.
   Exemplo: ?- meio([a,b,c,d,e],X).
    X = c
    true
    ?- meio([a,b,d,e],X).
    False
*/

elemento_meio([X],X).
elemento_meio([_|T],X) :-
    elimina_ultimo(T,PrefixoT),
    elemento_meio(PrefixoT,X).

elimina_ultimo([_],[]).
elimina_ultimo([H|T],[H|R]) :- elimina_ultimo(T,R).

/* Teste
Saída do programa
?- elemento_meio([1,2,3],X).
X = 2 .

?- elemento_meio([1,2,3,4],X).
false.
*/
```

15) Para a determinação da raiz quadrada de um número real x, 0 < x < 2, podemos utilizar o seguinte algoritmo:
Sejam a0 = x, c0 = 1 – x e, para i>0,
ai = ai-1 * (1 + ci-1/2)
ci = (ci-1)2 * (3 + ci-1)/4
Pode-se demonstrar que o limite de an com n tendendo ao infinito é a raiz quadrada de N.
Faça um programa Prolog que determine a raiz quadrada de um número real no intervalo acima, com pelo menos 5 dígitos significativos. (Dica: para tal verifique se | ai – ai-1 | < 0.0000001)
Considerações:
- Existe uma função nativa abs(X) que retorna o módulo de um número X;
- O programa deve ter o predicado principal de funtor prog1 e aridade 2, com os parâmetros X e RaizX.

```
/* Para a determinação da raiz quadrada de um número real x, 0 < x < 2, p
odemos utilizar o seguinte algoritmo:
Sejam a0 = x, c0 = 1 – x e, para i>0,
ai = ai-1 * (1 + ci-1/2)
ci = (ci-1)2 * (3 + ci-1)/4
Pode-
se demonstrar que o limite de an com n tendendo ao infinito é a raiz quad
rada de N.
Faça um programa Prolog que determine a raiz quadrada de um número real n
o intervalo acima,
 com pelo menos 5 dígitos significativos. (Dica: para tal verifique se |
ai – ai-1 | < 0.0000001)
Considerações:
- Existe uma função nativa abs(X) que retorna o módulo de um número X;
- O programa deve ter o predicado principal de funtor prog1 e aridade 2,
com os parâmetros X e RaizX.
*/

raiz_quadrada(X,RaizX) :-
    X > 0, X < 2,
    C0 is 1- X,
    raiz_quadrada(X,C0,RaizX).

raiz_quadrada(A1,C1,RaizX) :-
    A2 is A1 * (1+ C1/2),
    C2 is C1^2*(3+ C1)/4,
    Dif is abs( A2- A1),
    Dif >= 0.0000001,
    raiz_quadrada(A2,C2,RaizX).

raiz_quadrada(A1,C1,A2) :-
    A2 is A1*(1+C1/2),
    C2 is C1^2*(3+C1)/4,
```

```
    Dif is abs(A2-A1),
    Dif < 0.0000001.

/*
Teste do programa.
Para valores menores que 2.

?- raiz_quadrada(1.2,RaizX).
RaizX = 1.0954451150103321 .

Para valores maiores que 2.

?- raiz_quadrada(4,RaizX).
false.
*/
```

19) Faça um programa que reverte uma lista. Predicado reverso(Lista,Atsil)
Exemplo: ?-reverso([r,o,m,a],X) X = [a, m, o, r]

```
/* Faça um programa que reverte uma lista. Predicado reverso(Lista,Atsil)
Exemplo: ?-reverso([r,o,m,a],X) X = [a, m, o, r]
*/

inverte(Lista ,ListaInv) :- inverte(Lista,[],ListaInv).
inverte([],Ac,Ac).
inverte([H|T],Ac,ListaInv) :- inverte(T,[H|Ac],ListaInv).

/*
Teste
Saída do programa:

?- inverte([a,b,c],X).
X = [c, b, a].

?- inverte([r,o,m,a],X).
X = [a, m, o, r].

*/
```