UNIVERSIDADE FEDERAL DE UBERLÂNDIA

GABRIEL RIBEIRO BERNARDI - 11821BCC036

AARE - Avaliação Parcial da Aula 3

UBERLÂNDIA, MG

2020

Observação: algumas questões estão explicadas no próprio código para um melhor entendimento da solução

Exercício 1

```
somadigito(A, A) :- A<10.
somadigito(A, B) :- A>=10,
                    A1 is A // 10,
                    A2 is A mod 10,
                    somadigito(A1, B1),
                    B is B1 + A2,
                    !.
```

Esse exercício faz a soma dos dígitos de um determinado número, dividindo-o em unidades, dezenas, centenas e assim por diante, conforme a necessidade.

Exercício 2

```
%Criando variaveis referentes ao restp da divisao por 10
variable(0).
variable(1).
variable(2).
variable(3).
variable(4).
variable(5).
variable(6).
variable(7).
variable(8).
variable(9).

cryptogram(X, Y, Z) :- variable(X), variable(Y), variable(Z),
                       X > 0,
                       X =\= Y, X =\= Z, Y =\= Z,
                       R1 is X * 100 + Y * 10 + Z,
                       R2 is X * 10 + 3,
                       R3 is X * 100 + Z * 10 + Y,
                       R3 is R1 + R2.
```

Nessa solução temos que definir algumas variáveis referentes aos valores de resto, quando fazemos a divisão de um determinado valor por 10. Após isso, temos que o programa mostrará as possíveis soluções para o criptograma proposto.

Exercício 3

```
ehPar(X) :- 0 is mod(X,2).
```

```
assombroso(N) :- N == 1 -> !;                %Se N igual a 1, para a
execução, fazendo o corte
                 N < 1 -> false;             %Se N menor que 1, nao eh
numero assombroso
                 ( ehPar(N) -> Y is N // 2, %Se o numero for par,
faz-se a divisao por 2
                    assombroso(Y);
                    Y is N * 3 + 1,          %Se o numero for impar,
multiplica o valor por 3 e soma 1
                    assombroso(Y)
                 ).
```

Nesse exercício do assombroso, temos que fazer a entrada de um valor inteiro positivo, em que quando realizamos as operações, temos que chegar no valor 1. Se o valor de entrada for negativo, o número não pode ser assombroso. Caso o valor de entrada da função recursiva seja par, realiza-se a operação de divisão do valor por 2, no entanto, caso o valor de entrada seja ímpar, multiplica-se o valor por 3 e soma 1 a fim de tornar o número resultante par.

Exercício 4

```
ehTriangulo(N, Tn) :-
                        ehTriangulo(N, 0, Tn).

ehTriangulo(0, Tn, Tn).

ehTriangulo(N, Cont, Tn) :-
                            N > 0,
                            C is N - 1,
                            ContN is Cont + N,
                            ehTriangulo(C, ContN, Tn).
```

Nessa solução temos que calcular o seguinte, dado o número de bolinhas que constituem um lado do triângulo, calcule a quantidade de bolinhas que compõe o triângulo. No entanto, pode verificar também se, caso dado a quantidade de bolinhas que constituem um lado do triângulo e a quantidade de bolinhas que poderiam constituir o triângulo, se isso é verdadeiro ou não.

Exercício 5

```
verificaDiv7(X) :- 0 is mod(X,7).

div7(7).
```

```
div7(14).
div7(21).

div7(X) :-
            X > 21,
            A is X mod 10,
            B is X // 10,
            C is A * 2,
            D is B - C,
            verificaDiv7(D).
```

Nessa solução temos que verificar se, dado determinado valor, verifica-se se o mesmo é divisível por 7. No entanto, para a solução dessa questão, não basta somente utilizar a operação de “mod” por 7, mas sim, fazer as operações solicitadas no enunciado da questão.

Exercício 6

```
% Alguns numeros primos pre-definidos
primo(2).
primo(3).

primo(A) :-
            integer(A),
            A > 3,
            A mod 2 =\= 0,
            \+ multi(A,3).

multi(B,C) :-
                B mod C =:= 0.

multi(B,C) :-
                C * C < B,
                L2 is C + 2,
                multi(B,C2).

goldbach(4,[2,2]).
goldbach(B,C) :-
                B mod 2 =:= 0,
                B > 4,
                goldbach(B,C,3).

goldbach(B,[A,D],A) :-
```

```
                        D is B - A,
                        primo(D),
                        A < D.

goldbach(B,C,A) :-
                    A < B,
                    primo2(A,A1),
                    goldbach(B,C,A1).

primo2(A,A1) :-
                A1 is A + 2,
                primo(A1), !.

primo2(A,A1) :-
                A2 is A + 2,
                primo2(A2,A1).
```