CSPC 54 : Prolog Assignment-8

Name : Rajneesh Pandey,

Roll no. 106119100,

Class : CSE-B

## Instructions 1

****

**CODE :**

% | Connectives whit the traditional name

:- op(650, xfy, <->).        /\* equivalence  \*/

:- op(640, xfy, -->).        /\* implication  \*/

:- op(630, xfy, ∨).          /\* disjunction  \*/

:- op(620, xfy, ∧).          /\* conjunction  \*/

:- op(610, fy,  ~).          /\* negation     \*/

% | remove biconditionals using equivalence

cnf(A,F) :- doubleImplication(A,Z), Z=F.

% | cnf remove conditional using equivalence

cnf(A,F) :- implication(A,Z), Z=F.

% | cnf conjuntion elemental and

cnf(A,F) :- conjuntion(A,Z), Z=F.

% | cnf disjunction elemental or

cnf(A,F) :- disjunction(A,Z), Z=F.

% | cnf negation definition

cnf(A,F) :- negation(A,Z), Z=F.

% | otherwise atomic PROP'

cnf(A,F) :- atomic(A,Z), Z=F.

% | double implication <->

doubleImplication(~(A<->B),F) :- cnf(A-->B,X) , cnf(B-->A,Y) , cnf(~(X∧Y),F).

doubleImplication(A<->B,F) :- cnf(A-->B,X) , cnf(B-->A,Y) , cnf((X∧Y),F).

% | cnf denials implication ->

implication(~(A-->B),F) :- cnf(A-->B,W) , cnf(~(W),F).

% | cnf implication elemental ->

implication(A-->B,F) :- cnf(~(A),X) , cnf(B,Y) , cnf(X∨Y,F).

% | cnf conjuntion elemental and

% | Using De Morgan’s laws

conjuntion(~(A∧B),F) :- cnf(~(A),X) , cnf(~(B),Y) , disjunction((X∨Y),F).

conjuntion(A∧B,F) :- cnf(A,X) , cnf(B,Y) , (X∧Y) = F.

% | cnf disjunction elemental or

disjunction(A∨(B∧C),F) :- disjunction(A∨B,X) , disjunction(A∨C,Y) , (X∧Y) = F.

disjunction((A∧B)∨C,F) :- disjunction(A∨B,X) , disjunction(A∨C,Y) , (X∧Y) = F.

% | cnf Denials disyunciones'

% | Using De Morgan’s laws

disjunction(~(A∨B),F) :- cnf(~(A),X) , cnf(~(B),Y) , conjuntion((X∧Y),F).

disjunction(A∨B,F) :- cnf(A,X) , cnf(B,Y) , (X∨Y) = F.

% | cnf of two negation

negation(~(~X),F) :- cnf(X,F).

% | cnf of negation

negation(~A,F) :- cnf(A,Z), ~Z = F.

% | cnf atomic proposition definition

atomic(A,F) :- A=F.

**INPUT / OUTPUT:**

*cnf*(a∨(b∧c),F).

|  |  |
| --- | --- |
| **F** |  |
| (a∨b)∧(a∨c) | *1* |

*cnf*((a∧b)∨c,F).

|  |  |
| --- | --- |
| **F** |  |
| (a∨b)∧(a∨c) | *1* |

*cnf*(a-->b,F).

|  |  |
| --- | --- |
| **F** |  |
| ~a ∨ b | *1* |

*cnf*(~(a-->b),F).

|  |  |
| --- | --- |
| **F** |  |
| a ∧ ~b | *1* |

*cnf*(a<->b,F).

|  |  |
| --- | --- |
| **F** |  |
| (~a ∨ b)∧(~b ∨ a) |  |

**Screenshot:**

**Graphical user interface, application

Description automatically generated**

## Instructions 2 :

A picture containing text

Description automatically generated

**Code:**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Write and implement sequence of instructions to check whether %%

%%  a number is prime, even, odd, integer or real.              %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% is\_even

is\_even(N) :- Y is N mod 2, Y=0.

% is\_odd

is\_odd(N) :- Y is N mod 2, Y=1.

% is\_prime

is\_prime(N):- N>2, M is 2, \+is\_div(N,M).

is\_div(X,Y):- X mod Y =:= 0.

is\_div(X,Y):- Y =< sqrt(X), Y1 is Y+1, is\_div(X,Y1).

% Tests whether X is an integer.

is\_integer(X):- integer(X).

% Tests whether X is a real number.

is\_real(X):- not(integer(X)).

**Input / Output :**

*is\_even*(28).

*1***true**

*is\_odd*(30).

**false**

*is\_prime*(17).

*1***true**

*is\_integer*(18).

*1***true**

*is\_real*(18.004).

*1***true**

**Screenshot :**

**Graphical user interface

Description automatically generated**