**Artificial Intelligence**

**-Assignment 1. Prolog-**

**Student: Marcu Andrei Cristian**

**Group : CEN 2.2A**

1. Given the following knowledge base:

direct\_train(craiova, bucharest).

direct\_train(sibiu, craiova).

direct\_train(deva, sibiu).

direct\_train(brasov, deva).

direct\_train(pitesti, brasov).

direct\_train(ploiesti, pitesti).

direct\_train(constanta, ploiesti).

This KB contains facts about direct train route between two cities. Journeys between unlinked

cities are possible by chaining together direct train routes. Write a recursive predicate that tells

if a journey is possible between two cities.

My solution is:

%base case, direct route from X to Y

alternative\_route(X,Y) :- direct\_train(X,Y).

%recursive predicate that checks if there is a route between X and Y through other cities

alternative\_route(X,Y) :- direct\_train(X,Z), alternative\_route(Z,Y).

How it works:

Using recursion the predicate is taking every route from a city to the others to check if there is any alternative way to the destination.

Example:

?- alternative\_route(deva,craiova). -> alternative\_route(sibiu, craiova).

There is a direct way from sibiu to craiova -> returns true.

2. Suppose we are given a KB with the following facts:

translate(unu, one).

translate(doi, two).

translate(trei, three).

translate(patru, four).

translate(cinci, five).

translate(sase, six).

translate(sapte, seven).

translate(opt, eight).

translate(noua, nine).

Write a predicate which translate a list of Romanian number words to the corresponding list of

English number words.

Example:

For this list: [unu, doi, trei, cinci] you have to obtain [one, two, three, five].

My solution is:

%base case with empty lists

convert([],[]).

%predicate that converts a list of Romanian words into their equivalents in English. This predicate takes each word

%step by step from the list ListOfRomanianWords and saves it's equivalent in the result list(TranslatedWords)

convert([X|ListOfRomanianWords], [Y|TranslatedWords]) :- translate(X, Y), convert(ListOfRomanianWords, TranslatedWords).

How it works: It takes the head of the ListOfRomanianWords until the list is empty and saves it’s equivalent in English in the TranslatedWords list.

Example:

?- convert([unu,sase,noua], Result). -> Result = [one, six, nine]

3. Given a number N write a predicate that duplicates each element of a given list N times.

Example:

[1, 2, 3] duplicate 3 times each element will result in [1, 1, 1, 2, 2, 2, 3, 3, 3]

My solution is:

duplicate([InitialList], 1, [InitialList]) :- !.

%base case and preventing useless backtracking with cut operator. If the list has only one element we multiply it N times

duplicate([InitialList], N, [InitialList|X]) :-

I is N - 1,

I > 0,

duplicate([InitialList], I, X).

%taking each element from the list one by one and duplicate it N times

duplicate([InitialList|T], N, X) :-

duplicate([InitialList], N, Y),

duplicate(T, N, Z),

!, append(Y, Z, X).

How it works:

It creates a list for each element in the initial list duplicated by N times using the predicate duplicate([InitialList], N, [InitialList|X]) . After that it takes the next elements from the list using the recursive call of the predicate duplicate([InitialList|T], N, X), having the first parameter the Tail of the list. Then we concatenate the lists created before in X.

Example :

?- duplicate([1,2,3], 2, Result). -> Result = [1, 1, 2, 2, 3, 3].

4. Write a predicate that Insert an element at a given position into a list.

Example:

Initial list: [1, 2, 3]. Element to insert: x. Position where to insert: 2. Resulting list: [1, x,

2, 3]

My solution :

insertAt(List,0,Val,[Val|List]).

%base case

insertAt([List|Tail],Pos,Val,[List|Rest]):-

Pos1 is Pos-1,

insertAt(Tail,Pos1,Val,Rest).

%Iterates through the first Pos-1 elements and saves them.

%When it reaches position Pos it inserts the Value

%Then it continues copying the next elements from position Pos+1

How it works:

It appends the first Pos – 1 elements in the list until it reaches Pos, then inserts the Val at Pos, and in the end it continues appending the elements starting from Pos + 1.

Example:

?-insertAt([1,2,3,4], 2, 15, Result). -> Result = [1, 2, 15, 3, 4].

5. Build a predicate that will generate a list with all consecutive integers within a given range.

Example:

Range: 5 -> 9. Resulting list: [5, 6, 7, 8, 9]

My solution :

listLeftRight(L,L,[L]) :- !.

%base case with cut operator

listLeftRight(L,R,[L|Tail]) :-

L =< R,

Aux is L+1,

listLeftRight(Aux,R,Tail).

%we iterate from L to R and we insert each L in the list until it is equal to R

How it works:

We iterate from L to R until L = R, and everytime before iterating we save L in our list.

Example:

?- listLeftRight(4,9,Result). -> Result= [4, 5, 6, 7, 8, 9].