ARTIFICIAL INTELLIGENCE

MINI PROJECT

CODE

Main.pl

% Import the Diseases Module

:- consult("Diseases.pl").

% Special Dynamic Facts

:- dynamic emergency/1.

:- dynamic diagnosis/3.

/\* MAIN SESSION HANDLER \*/

diagnose(PatientName) :-

    % Greet Patient

    format("Good day, ~w!~n~n", [PatientName]),

    % Display Program Details

    write\_ln("Medical Diagnostc System:."),

    write\_ln("Department of Computer Science, SVNIT\n"),

    % Display Confidentaility Clause

    write\_ln("The data collected from you during this session is for diagnostic purposes only and shall only be stored throughout this session."),

    write\_ln("We will be asking you a series of questions regarding your symptoms to establish an idea of possible illnesses.\n"),

    % Get User Consent

    write("Would you like to proceed? [y / n]: "), read(Consent),

    (Consent = 'Y' ; Consent = 'y'),

    nl, nl, nl,

    % Prompt User of the Questionnaire

    write\_ln("\nWe will now begin with the questionnaire.\n"),

    % Get User Consent

    write("Would you like to proceed? [y / n]: "), read(Consent),

    (Consent = 'Y' ; Consent = 'y'),

    nl, nl, nl,

    % Get the HPI

    getHPI(PatientName),

    % Get the Chief Complaint and Diagnose

    (chiefComplaint(PatientName) ; true), !,

    % Parse Results

    diagnosedSummary(PatientName).

/\* Get the Patient's HPI \*/

getHPI(P) :-

    % Ask first if emergency

    write("ARE YOU IN AN EMERGENCY? [y / n] : "), read(EMERG),

    ((EMERG = 'y' ; EMERG = 'Y') -> assert(emergency(P)) ; true),

    % Temperature

    write("Please enter your temperature (in degrees celsius) : "), read(Temp),

    assert(temperature(P, Temp)),

    % Diastolic Blood Pressure

    write("Please enter your diastolic blood pressure (in mmHg) : "), read(DiasBP),

    % Systolic Blood Pressure

    write("Please enter your systolic blood pressure (in mmHg) : "), read(SysBP),

    assert(bloodPressure(P, DiasBP, SysBP)),

    % Height

    write("Please enter your height (in centimeters) : "), read(Height),

    assert(height(P, Height)),

    % Weight

    write("Please enter your weight (in kilograms) : "), read(Weight),

    assert(weight(P, Weight)).

/\* Chief Complaint Questionnaire \*/

chiefComplaint(P) :-

    % Gather the Chief Complaint, ask for some unique symptoms

    % VIRAL (and Malaria)

    %   - Fever, Malaise, Colds/RunnyNose

    % RESPIRATORY

    %   - Chest Pain, Coughing

    % GASTRO

    %   - LooseStools, Dehydration, Questionable Food Intake

    % CARDIO

    %   - Elevated BP

    % LIFESTYLE DISEASE

    %   - Weight Loss

    % VIRAL

    (viralQuestionnaire(P) ; true), !,

    % Check if the patient is already diagnosed

    (not(diagnosis(P, \_, \_)) ; emergency(P)),

    % RESPIRATORY

    (respiratoryQuestionnaire(P) ; true), !,

    % Check if the patient is already diagnosed

    (not(diagnosis(P, \_, \_)) ; emergency(P)),

    % GASTRO

    (gastroQuestionnaire(P) ; true), !,

    % Check if the patient is already diagnosed

    (not(diagnosis(P, \_, \_)) ; emergency(P)),

    % LIFESTYLE

    lifestyleQuestionnaire(P), !.

/\* Questions for Viral Diseases \*/

viralQuestionnaire(P) :-

    % Patient Must Have a Fever

    fever(P),

    % Body Malaise

    write("Do you feel an overall weakness in your whole body? [y / n] : "), read(BM),

    ((BM = 'y' ; BM = 'Y') -> assert(malaise(P)) ; true),

    % Runny Nose / Colds

    write("Do you have a runny nose? [y / n] : "), read(RN),

    ((RN = 'y' ; RN = 'Y') -> assert(runnyNose(P)) ; true),

    write("Do you have colds? [y / n] : "), read(Lamig),

    ((Lamig = 'y' ; Lamig = 'Y') -> assert(cold(P)) ; true),

    % Test for Malaria (if there is body malaise)

    (malaise(P) -> (

        write("Do you feel chills (especially at night)? [y / n] : "), read(Chills),

        ((Chills = 'Y' ; Chills = 'y') -> assert(chills(P)) ; true),

        write("Are you having frequent headaches? [y / n] : "), read(HA),

        ((HA = 'Y' ; HA = 'y') -> assert(headache(P)) ; true),

        write("Are you having nausea? [y / n] : "), read(NA),

        ((NA = 'Y' ; NA = 'y') -> assert(nausea(P)) ; true),

        write("Are you vomiting often? [y / n] : "), read(Vomit),

        ((Vomit = 'Y' ; Vomit = 'y') -> assert(vomiting(P)) ; true),

        write("Have you been in a forest, jungle, fields, or bitten by a mosquito? [y / n] : "), read(RiskArea),

        ((RiskArea = 'Y' ; RiskArea = 'y') -> assert(malariaRiskArea(P)) ; true),

        % Get Malaria Certainty

        (malaria(P, MalariaCertainty))

    ) ; true),

    % Test for Dengue (if there is body malaise)

    (malaise(P) -> (

        (not(headache(P)) -> (write("Are you having frequent headaches? [y / n] : "), read(HAA),

        ((HAA = 'Y' ; HAA = 'y') -> assert(headache(P)) ; true)); true),

        write("Are you having rashes? [y / n] : "), read(Rash),

        ((Rash = 'Y' ; Rash = 'y') -> assert(rash(P)) ; true),

        % Get Dengue Certainty

        (dengue(P, DengueCertainty))

    ) ; true),

    % If malaise does not exist, declare certainties to 0

    (not(malaise(P)) -> DengueCertainty is 0, MalariaCertainty is 0 ; true),

    % Test for Flu (if there is colds/runnynose)

    (runnyNose(P) -> (

        (not(cough(P)) -> (write("Are you having cough? [y / n] : "), read(Cough),

        ((Cough = 'Y' ; Cough = 'y') -> assert(cough(P))) ; true); true),

        % Get Flu Certainty

        (flu(P, FluCertainty))

    ) ; true),

    % If there is no runny nose, declare certainty to 0

    (not(runnyNose(P)) -> FluCertainty is 0 ; true), !,

    % Check if any certainties are over 75, if not emergency

    (not(emergency(P)) -> (

        ((FluCertainty >= 75) -> assert(diagnosis(P, "Flu", FluCertainty)) ; true), !,

        ((DengueCertainty >= 75) -> assert(diagnosis(P, "Dengue", DengueCertainty)) ; true), !,

        ((MalariaCertainty >= 75) -> assert(diagnosis(P, "Malaria", MalariaCertainty)) ; true), !

    ); true), !,

    % If emergency, store diagnosis anyway

    ((emergency(P) -> (

        assert(diagnosis(P, "Flu", FluCertainty)),

        assert(diagnosis(P, "Dengue", DengueCertainty)),

        assert(diagnosis(P, "Malaria", MalariaCertainty))

    )); true), !.

/\* Questions for Respiratory Diseases \*/

respiratoryQuestionnaire(P) :-

    % Patient must have chest pain and Coughing

    % Chest Pain

    write("Are you having chest pain? [y / n] : "), read(CP),

    ((CP = 'y' ; CP = 'Y') -> assert(chestPain(P)) ; true),

    % Coughing

    (not(cough(P)) -> (

        write("Have you been coughing? [y / n] : "), read(Ubo),

        ((Ubo = 'y' ; Ubo = 'Y') -> assert(cough(P)) ; true)

    ) ; true),

    % Check if the patient has chest pain and is coughing

    chestPain(P), cough(P),

    % Check for phlegm

    write("Are you coughing with phlegm? [y / n] : "), read(Dura),

    ((Dura = 'y' ; Dura = 'Y') -> assert(phlegm(P)) ; true),

    % Check for Tuberculosis if coughing for more than 2 weeks

    write("Have you been coughing for 2 weeks or more? [y / n] : "), read(MahabangUbo),

    ((MahabangUbo = 'y' ; MahabangUbo = 'Y') -> assert(longCough(P)) ; true),

    (longCough(P) -> (

        write("Do you cough up blood? [y / n] : "), read(UboDugo),

        ((UboDugo = 'Y' ; UboDugo = 'y') -> assert(bloodCough(P)) ; true),

        % Get TB Certainty

        tuberculosis(P, TBCertainty)

    ) ; true),

    % Set certainty to 0 if not longCough

    (not(longCough(P)) -> TBCertainty is 0 ; true),

    % Check for Pneumonia if feverish

    (fever(P) -> (

        write("Are you having shortness of breath? [y / n] : "), read(ShortBreath),

        ((ShortBreath = 'Y' ; ShortBreath = 'y') -> assert(shortnessBreath(P)) ; true),

        % Get Dengue Certainty

        pneumonia(P, PneuCertainty)

    ) ; true),

    % Set certainty to 0 if not feverish

    (not(fever(P)) -> PneuCertainty is 0 ; true),

    % Check for Bronchitis if Wheezing

    write("Are you wheezing? [y / n] : "), read(Wheeze),

    ((Wheeze = 'Y' ; Wheeze = 'y') -> assert(wheezing(P)) ; true),

    % Get Bronchitis Certainty

    ((wheezing(P) -> (bronchitis(P, BronchitisCertainty)) ; true)),

    % Set certainty to 0 if not wheezing

    (not(wheezing(P)) -> BronchitisCertainty is 0 ; true),

    % Check if any certainties are over 75, if not emergency

    (not(emergency(P)) -> (

        ((TBCertainty >= 75) -> assert(diagnosis(P, "Tuberculosis", TBCertainty)) ; true), !,

        ((PneuCertainty >= 75) -> assert(diagnosis(P, "Pneumonia", PneuCertainty)) ; true), !,

        ((BronchitisCertainty >= 75) -> assert(diagnosis(P, "Bronchitis", BronchitisCertainty)) ; true), !

    ); true), !,

    % If emergency, store diagnosis anyway

    ((emergency(P) -> (

        assert(diagnosis(P, "Tuberculosis", TBCertainty)),

        assert(diagnosis(P, "Pneumonia", PneuCertainty)),

        assert(diagnosis(P, "Bronchitis", BronchitisCertainty))

    )); true), !.

/\* Questions for Gastrointestinal Diseases \*/

gastroQuestionnaire(P) :-

    % Check if the patient has loose bowel movement

    % LBM

    write("Are you having loose bowel movement? [y / n] : "), read(LBM),

    ((LBM = 'y' ; LBM = 'Y') -> assert(looseStools(P)) ; true),

    % Must have looseStools

    looseStools(P),

    % Check for diarrhea

    % Check for Abdominal Pain

    write("Are you experiencing abdominal pain? [y / n] : "), read(AwitTiyan),

    ((AwitTiyan = 'y' ; AwitTiyan = 'Y') -> assert(abdominalPain(P)) ; true),

    % Check for Dehydration

    write("Are you dehydrated? [y / n] : "), read(Dehyd),

    ((Dehyd = 'y' ; Dehyd = 'Y') -> assert(dehydration(P)) ; true),

    % Get Diarrhea Certainty

    diarrhea(P, DiarrheaCertainty),

    % Check for Cholera only if there is diarrhea

    (((diarrhea(P, DCer), DCer >= 75) -> (

        % Check for more profuse bowel movement

        write("Is your bowel movement really severe (totally liquid)? [y / n] : "), read(AwitAwitTiyan),

        ((AwitAwitTiyan = 'y' ; AwitAwitTiyan = 'Y') -> assert(fasterStools(P)) ; true),

        % Check for more profuse dehydration

        write("Is your dehydration severe? [y / n] : "), read(AwitDehydration),

        ((AwitDehydration = 'y' ; AwitDehydration = 'Y') -> assert(fasterDehydration(P)) ; true),

        % Get cholera certainty

        cholera(P, CholeraCertainty)

    )) ; true),

    (not((diarrhea(P, DCCer), DCCer >= 75)) -> CholeraCertainty is 0 ; true),

    % Check if any certainties are over 75, if not emergency

    (not(emergency(P)) -> (

        ((DiarrheaCertainty >= 75) -> assert(diagnosis(P, "Diarrhea", DiarrheaCertainty)) ; true), !,

        ((CholeraCertainty >= 75) -> assert(diagnosis(P, "Cholera", CholeraCertainty)) ; true), !

    ); true), !,

    % If emergency, store diagnosis anyway

    ((emergency(P) -> (

        assert(diagnosis(P, "Diarrhea", DiarrheaCertainty)),

        assert(diagnosis(P, "Cholera", CholeraCertainty))

    )); true), !.

/\* Questions for Lifestyle Diseases \*/

lifestyleQuestionnaire(P) :-

    % HYPERTENSION - Patient must have elevated blood pressure

    (highBP(P) -> (

        % Check for Headache

        write("Are you experiencing headaches? [y / n] : "), read(SakitUlo),

        ((SakitUlo = 'y' ; SakitUlo = 'Y') -> assert(headache(P)) ; true),

        % Vision Changes

        write("Are you experiencing vision changes? [y / n] : "), read(IbaPagtingin),

        ((IbaPagtingin = 'y' ; IbaPagtingin = 'Y') -> assert(blurryVision(P)) ; true),

        % Get Hypertension Certainty

        hypertension(P, HypertensionCertainty)

    ) ; true),

    (not(highBP(P)) -> HypertensionCertainty is 0 ; true),

    % DIABETES - Check for all symptoms

    % Check for Increased Urination

    write("Are you experiencing increased urination? [y / n] : "), read(DamingIhi),

    ((DamingIhi = 'y' ; DamingIhi = 'Y') -> assert(increasedUrine(P)) ; true),

    % Check for Increased Thirst

    write("Are you experiencing increased thirst? [y / n] : "), read(Uhaw),

    ((Uhaw = 'y' ; Uhaw = 'Y') -> assert(increasedThirst(P)) ; true),

    % Check for Weight Loss

    write("Have you been losing weight lately? [y / n] : "), read(GettingLighter),

    ((GettingLighter = 'y' ; GettingLighter = 'Y') -> assert(weightLoss(P)) ; true),

    % Check for Family History

    write("Do you have family history of Diabetes? [y / n] : "), read(FamHistoryDiabetes),

    ((FamHistoryDiabetes = 'y' ; FamHistoryDiabetes = 'Y') -> assert(diabetesFamily(P)) ; true),

    % Get Diabetes Certainty

    (diabetes(P, DiabetesCertainty) ; true),

    % Check if any certainties are over 75, if not emergency

    (not(emergency(P)) -> (

        ((HypertensionCertainty >= 75) -> assert(diagnosis(P, "Hypertension", HypertensionCertainty)) ; true), !,

        ((DiabetesCertainty >= 75) -> assert(diagnosis(P, "Diabetes", DiabetesCertainty)) ; true), !

    ); true), !,

    % If emergency, store diagnosis anyway

    ((emergency(P) -> (

        assert(diagnosis(P, "Hypertension", HypertensionCertainty)),

        assert(diagnosis(P, "Diabetes", DiabetesCertainty))

    )); true), !.

/\* Get the Patient's Diagnosis Summary \*/

diagnosedSummary(P) :-

    %  EMERGENCY

    (emergency(P) -> (

        write("EMERGENCY DIAGNOSIS\n"),

        % Display all the listed Diagnosis

        getTopDiagnosis(P, [Name | [Certainty | \_]]),

        (((Certainty > 0) -> (

            write\_ln("\n\nYou have been diagnosed with the following:"),

            format("~s with a certainty level of ~w%~n", [Name, Certainty])

        )) ; true), !

    ) ; true),

    % NON-EMERGENCY

    (not(emergency(P)) -> (

        % Check if there is a valid diagnosis

        ((not(diagnosis(P, \_, \_)) -> write\_ln("This system cannot provide a diagnosis. Please visit a higher health institution.")); true), !,

        % Display only the top diagnosis

        ((diagnosis(P, \_, \_) -> (

            % Get the top

            getTopDiagnosis(P, [Name | [Certainty | \_]]),

            write\_ln("\n\nYou have been diagnosed with the following:"),

            format("~s with a certainty level of ~w%~n", [Name, Certainty])

        )) ; true), !

    ) ; true).

/\* UTILITY PREDICATES \*/

% Print Diagnosis

printAllDiseases(P) :-

    listDiseases(P, DiseaseList),

    printList(DiseaseList).

% Get top diagnosis

getTopDiagnosis(P, Disease) :-

    listDiseases(P, DiseaseList),

    listMax(DiseaseList, Disease), !.

% Get the maximum in the disease list

listMax([X],X) :- !, true.

listMax([[\_|C]|R], [Mn|Mc]) :-

    listMax(R, [Mn|Mc]), Mc >= C.

listMax([[N|C]|R], [N|C]) :-

    listMax(R, [\_|Mc]),

    C > Mc.

% Get the disease list

listDiseases(P, []) :- not(diagnosis(P,\_,\_)), !.

listDiseases(P, [Disease | Tail]) :-

    retract(diagnosis(P, D, C)), !,

    Disease = [D,C],

    listDiseases(P, Tail), !.

% Print a List

printList([]).

printList([[N | [C | \_]] | T]) :-

    format("~s with a certainty of ~w%~n", [N,C]),

    printList(T).

disease.pl

/\*

    CSINTSY MCO3

    AMOGUIS, Adriel Isaiah V.

    SUN, Benedict C.

    Database of Diseases

\*/

% Import the Symptoms Database

:- consult("Symptoms.pl").

/\* DISEASE DEFINITIONS - Weighted Symptoms \*/

% Malaria

malaria(P, C) :-

    L = [],

    ((fever(P) -> append([30], L, L1); not(fever(P)) -> append([], L, L1)) ; true), !,

    ((chills(P) -> append([20], L1, L2); not(chills(P)) -> append([], L1, L2)) ; true), !,

    ((headache(P) -> append([15], L2, L3); not(headache(P)) -> append([], L2, L3)) ; true), !,

    ((nausea(P) -> append([5], L3, L4); not(nausea(P)) -> append([], L3, L4)) ; true), !,

    ((vomiting(P) -> append([5], L4, L5); not(vomiting(P)) -> append([], L4, L5)) ; true), !,

    ((malaise(P) -> append([5], L5, L6); not(malaise(P)) -> append([], L5, L6)) ; true), !,

    ((malariaRiskArea(P) -> append([20], L6, L7); not(malariaRiskArea(P)) -> append([], L6, L7)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L7, C).

% Flu

flu(P, C) :-

    L = [],

    ((fever(P) -> append([50], L, L1); not(fever(P)) -> append([], L, L1))),

    ((cough(P) -> append([10], L1, L2); not(cough(P)) -> append([], L1, L2))),

    ((cold(P) -> append([20], L2, L3); not(cold(P)) -> append([], L2, L3))),

    ((runnyNose(P) -> append([20], L3, L4); not(runnyNose(P)) -> append([], L3, L4))),

    % Calculate & Return the Certainty Factor

    listSum(L4, C).

% Dengue

dengue(P, C) :-

    L = [],

    ((fever(P) -> append([50], L, L1); not(fever(P)) -> append([], L, L1)) ; true), !,

    ((headache(P) -> append([30], L1, L2); not(headache(P)) -> append([], L1, L2)) ; true), !,

    ((rash(P) -> append([10], L2, L3); not(rash(P)) -> append([], L2, L3)) ; true), !,

    ((malaise(P) -> append([10], L3, L4); not(malaise(P)) -> append([], L3, L4)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L4, C).

% Tuberculosis

tuberculosis(P, C) :-

    L = [],

    ((longCough(P) -> append([50], L, L1); not(longCough(P)) -> append([], L, L1)) ; true), !,

    ((chestPain(P) -> append([5], L1, L2); not(chestPain(P)) -> append([], L1, L2)) ; true), !,

    ((bloodCough(P) -> append([45], L2, L3); not(bloodCough(P)) -> append([], L2, L3)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L3, C).

% Pneumonia

pneumonia(P, C) :-

    L = [],

    ((cough(P) -> append([40], L, L1); not(cough(P)) -> append([], L, L1)) ; true), !,

    ((phlegm(P) -> append([10], L1, L2); not(phlegm(P)) -> append([], L1, L2)) ; true), !,

    ((chestPain(P) -> append([5], L2, L3); not(chestPain(P)) -> append([], L2, L3)) ; true), !,

    ((shortnessBreath(P) -> append([30], L3, L4); not(shortnessBreath(P)) -> append([], L3, L4)) ; true), !,

    ((fever(P) -> append([15], L4, L5); not(fever(P)) -> append([], L4, L5)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L5, C).

% Bronchitis

bronchitis(P, C) :-

    L = [],

    ((cough(P) -> append([50], L, L1); not(cough(P)) -> append([], L, L1)) ; true), !,

    ((chestPain(P) -> append([10], L1, L2); not(chestPain(P)) -> append([], L1, L2)) ; true), !,

    ((wheezing(P) -> append([40], L2, L3); not(wheezing(P)) -> append([], L2, L3)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L3, C).

% Diarrhea

diarrhea(P, C) :-

    L = [],

    ((looseStools(P) -> append([70], L, L1); not(looseStools(P)) -> append([], L, L1)) ; true), !,

    ((abdominalPain(P) -> append([20], L1, L2); not(abdominalPain(P)) -> append([], L1, L2)) ; true), !,

    ((dehydration(P) -> append([10], L2, L3); not(dehydration(P)) -> append([], L2, L3)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L3, C).

% Cholera

cholera(P, C) :-

    L = [],

    (((diarrhea(P, DC), DC >= 75) -> append([34], L, L1); not(diarrhea(P, DDC), DDC >= 75) -> append([], L, L1)) ; true), !,

    ((fasterStools(P) -> append([33], L1, L2); not(fasterStools(P)) -> append([], L1, L2)) ; true), !,

    ((fasterDehydration(P) -> append([33], L2, L3); not(fasterDehydration(P)) -> append([], L2, L3)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L3, C).

% Hypertension

hypertension(P, C) :-

    L = [],

    ((highBP(P) -> append([70], L, L1); not(highBP(P)) -> append([], L, L1)) ; true), !,

    ((headache(P) -> append([20], L1, L2); not(headache(P)) -> append([], L1, L2)) ; true), !,

    ((visionChanges(P) -> append([10], L2, L3); not(visionChanges(P)) -> append([], L2, L3)) ; true), !,

    ((obese(P) -> append([10], L3, L4); not(obese(P)) -> append([], L3, L4)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L4, C).

% Diabetes

diabetes(P, C) :-

    L = [],

    ((increasedUrine(P) -> append([25], L, L1); not(increasedUrine(P)) -> append([], L, L1)) ; true), !,

    ((increasedThirst(P) -> append([25], L1, L2); not(increasedThirst(P)) -> append([], L1, L2)) ; true), !,

    ((weightLoss(P) -> append([25], L2, L3); not(weightLoss(P)) -> append([], L2, L3)) ; true), !,

    ((diabetesFamily(P) -> append([25], L3, L4); not(diabetesFamily(P)) -> append([], L3, L4)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L4, C).

/\* UTILITY PREDICATES \*/

% This predicate gets the sum of all elements of a number list.

listSum([], 0).

listSum([E|L], Sum) :-

    listSum(L, S),

    Sum is S + E.

/\* OLD PREDICATES

% This predicate gets the sum of all elements of a number list.

listSum([X], X).

listSum([E|L], Sum) :-

    listSum(L, S),

    Sum is S + E.

\*/

Symptoms.pl

/\*

    CSINTSY MCO3

    AMOGUIS, Adriel Isaiah V.

    SUN, Benedict C.

    DATABASE OF SYMPTOMS

\*/

% DYNAMIC FACTS

% NEW

% HPI Questions

:- dynamic temperature/2.

:- dynamic bloodPressure/3.

:- dynamic height/2.

:- dynamic weight/2.

% PHYSICAL

:- dynamic pale/1.

:- dynamic redSpots/1.

:- dynamic badPulse/1.

% COMPLAINTS

:- dynamic malaise/1.

:- dynamic headache/1.

:- dynamic runnyNose/1.

:- dynamic cold/1.

:- dynamic rash/1.

:- dynamic chills/1.

:- dynamic vomiting/1.

:- dynamic nausea/1.

:- dynamic chestPain/1.

:- dynamic cough/1.

:- dynamic phlegm/1.

:- dynamic shortnessBreath/1.

:- dynamic blurryVision/1.

:- dynamic looseStools/1.

:- dynamic abdominalPain/1.

:- dynamic increasedUrine/1.

:- dynamic increasedThirst/1.

:- dynamic weightLoss/1.

:- dynamic dehydration/1.

% Follow-Up

:- dynamic longCough/1.

:- dynamic wheezing/1.

:- dynamic bloodCough/1.

:- dynamic fasterStools/1.

:- dynamic fasterDehydration/1.

:- dynamic diabetesFamily/1.

:- dynamic malariaRiskArea/1.

% PREDICATES

% P - patient

highBP(P) :-

    bloodPressure(P, Diastolic, Systolic),

    Diastolic > 89, Systolic > 120.

highTemp(P) :-

    temperature(P, Temp),

    (Temp >= 37.5).

mildTemperature(P) :-

    temperature(P, Temp),

    (Temp >= 37),

    (Temp < 37.5).

fever(P) :-

    highTemp(P).

visionChanges(P) :-

    blurryVision(P).

obese(P) :-

    getBMI(P, BMI),

    BMI >= 30.

getBMI(P, BMI) :-

    height(P, Height),

    weight(P, Weight),

    % Convert height to meters squared

    mHeight is div(Height, 100),

    msHeight is mHeight\*\*2,

    BMI is div(Weight, msHeight).

Testing.pl

:- dynamic cough/1.

:- dynamic chestPain/1.

:- dynamic wheezing/1.

:- dynamic diagnosis/3.

% Bronchitis

bronchitis(P, C) :-

    L = [],

    ((cough(P) -> append([50], L, L1); not(cough(P)) -> append([], L, L1)) ; true), !,

    ((chestPain(P) -> append([10], L1, L2); not(chestPain(P)) -> append([], L1, L2)) ; true), !,

    ((wheezing(P) -> append([40], L2, L3); not(wheezing(P)) -> append([], L2, L3)) ; true), !,

    % Calculate & Return the Certainty Factor

    listSum(L3, C).

% This predicate gets the sum of all elements of a number list.

listSum([], 0).

listSum([E|L], Sum) :-

    listSum(L, S),

    Sum is S + E.

testDecision :-

    write\_ln("Enter a decision [Y/N]"),

    read(Decision),

    (Decision = 'Y' ; Decision = 'y').

% Get top diagnosis

getTopDiagnosis(P, Disease) :-

    listDiseases(P, DiseaseList),

    listMax(DiseaseList, Disease), !.

% Get the maximum in the disease list

listMax([X],X) :- !, true.

listMax([[\_|C]|R], [Mn|Mc]) :-

    listMax(R, [Mn|Mc]), Mc >= C.

listMax([[N|C]|R], [N|C]) :-

    listMax(R, [\_|Mc]),

    C > Mc.

% Get the disease list

listDiseases(P, []) :- not(diagnosis(P,\_,\_)), !.

listDiseases(P, [Disease | Tail]) :-

    retract(diagnosis(P, D, C)), !,

    Disease = [D,C],

    listDiseases(P, Tail), !.

