Домашна 2 PROLOG Бази на податоци

Задача 1 фамилија

А) родени различен град

%rodeni_razlicen_grad(Kolku).

rodeni_razlicen_grad(Kolku):-

findall(X,rodenRazlicenGrad(X),L),

dolzina(L,Kolku).

Со **findall** најди ги сите лице за кои важи условот, за да избројиш колку се користи **dolzina** што враќа должина на листа.

%rodenRazlicenGrad(L) True ako liceto so sifra L e rodeno vo razlicen grad od gradot na roditelite

rodenRazlicenGrad(L):-

lice(L,_,_,_,_,GLice,_),

familija(T,M,D), clen(L,D),

rodenGrad(T,GTatko), rodenGrad(M,GMajka),

GLice\==GTatko, GLice\==GMajka.

rodenRazlicenGrad - најди го лицето со шифра проследена на влез, најди ја фамилијата каде лицето е член на листата со деца, најди го градот каде е роден секој од родителите провери дали градот во кој е родено лицето е различно од градовите каде се родени родителите.

%rodenGrad(L,G) G e gradot vo koj e rodeno liceto so sifra L

rodenGrad(L,G):-lice(L,_,_,_,_,G,_).

%clen(X,L) True ako X e element na L.

clen(X,[X|_]).

clen(X,[_|L]):-clen(X,L).

%dolzina(L,N) N e dolzina na listata L

dolzina([],0).

dolzina([_|L],N):-dolzina(L,M), N is M+1.

Б) Предци

%predci(Sifra,L)

predci(Sifra,L):-findall(P,predok_uslov(Sifra,P),L).

со **findall** најди ги сите лица за кој е исполнет условот (да се предци на лицето проследено на влез, да се од ист пол и да имаат близок роденден).

%predok_uslov(L,P) P e predok na L, L i P se od ist pol i imaat blizok rodenden.

predok_uslov(L,P):-predok(L,P),

lice(L,_,_,Pol,LData,_,_),

lice(P,_,_,Pol,PData,_,_),

blizokRodenden(LData,PData).

predok_uslov(L,P) – со предикатот **predok(L,P)** провери дали **P** е прекод на **L,** потоа провери дали се од ист пол, и на крај провери дали имаат близок роденден со предикатот **blizokRodenden**

%predok(L,P) P e predok na L

predok(L,P):-roditel(L,P).

predok(L,P):-roditel(L,R),predok(R,P).

predok(L,P) – провери дали **P** е родител на **L** ако е врати **True** , ако не најди го родителот на **L** и провери дали **P** е негов предок.

%roditel(L,R) True ako R e roditel na L

roditel(L,R):-familija(R,_,D),clen(L,D).

roditel(L,R):-familija(_,R,D),clen(L,D).

roditel(L,R) - True ако **R** е татко во фамилија каде **L** е член на листата со деца или ако **R** е мајка во фамилијата каде **L** е член на листата со деца.

%blizokRodenden(Data1,Data2) Data1 i Data2 se razlikuvaat za edna nedela godinata e nebitna.

blizokRodenden(datum(D1,M,_),datum(D2,M,_)):- D1>=D2,Raz=D1-D2, Raz=<7.

blizokRodenden(datum(D1,M,_),datum(D2,M,_)):- D1<D2,Raz=D2-D1, Raz=<7.

blizokRodenden(datum(D1,M1,_),datum(D2,M2,_)):- MRaz is M1 - M2, MRaz=1,

D1_new is D1+30, Raz is D1_new - D2, Raz=<7.

blizokRodenden(datum(D1,M1,_),datum(D2,M2,_)):- MRaz is M2 - M1, MRaz=1,

D2_new is D2+30, Raz is D2_new - D1, Raz=<7.

blizokRodenden - Доколку и двата датуми се во ист месец провери дали разликата помегу деновите е помала или еднаква на 7. Доколку датумите се во последователни месеци додади 30 на денот за датумот кој е во предходниот месец и потоа пресметај ја разликата, доколку е помала или еднаква на 7 врату **True**.

Задача 2 телефони

А) Најмногу остварени комуникации

%najbroj(X,Y) X ime, Y prezime na brojot koj napravil razgovori so najmnogu drugi broevi

najbroj(X,Y):- maxRazgovori(Tel), getImePrez(Tel,X,Y).

Најди го телефонскиот број кој напрваил разговори со најмногу други броеви, и потоа најди го името и презимето на сопственикот на бројот.

%maxRazgovori(Tel) Tel e brojot koj napravil razgovori so najmnogu drugi broevi

maxRazgovori(Tel):-findall((T,N),brojPovici(T,N),[M|L]), max(L,M,Tel).

со помош на **findall** за секој телефонски број најди го бројот на телефонски броеви со кој има направено разговор. Потоа предади ја листата и првиот член на предикатот **max и** најди кој има најмногу.

%max(L,(TempX,TempMax),MaxX) MaxX e najgolem element vo L kade L se sostoi od (X,N) elementi

%kade N e svojstvo na X spored koe se bara najgolem, primer X e tel broj a N e broj na povici

%(TempX,TempMax) e momentalniot najgolem element

max([],(X,_),X):-!.

max([(X,N)|L],(_,TempMax),Y):- N>TempMax, max(L,(X,N),Y).

max([(_,N)|L],(TempX,TempMax),Y):- N=<TempMax, max(L,(TempX,TempMax),Y).

max – изминувај ја листата елемент по елемент доколку моменталиот елемент има вредност по која се споредува поголема од привремената максимална вредност, постави го привремениот максимален елемент на моменталниот, во спротивен случај продолжи со итерирање, итерирај се додека не ја испразниш листата тогаш привремениот максимален станува излезен елемент.

%brojPovici(T,N) N e broj na tel broevi so koj T ima napraveno razgovor

brojPovici(T,N):- setof(T2,razgovor(T,T2),L),dolzina(L,N).

brojPovici – Ги наоѓаме сите различни броеви со кој везниот телефон има постигнато разговор, користиме **setof** (се елиминираат дупликати во случај да постојат и дојдовени и појдовни разговори помеѓу два телефони) и потоа ја враќаме должината на листата

%razgovor(T1,T2) True ako T1 i T2 imale telefonski razgovor

razgovor(T1,T2):-povikan(T1,T2).

razgovor(T1,T2):-povikan(T2,T1).

Razgovor – Врати **True** доколку првиот тел бил повикан од вториот или обратно доколку првиот го повикал вториот.

%povikan(Tel1,Tel2) True ako Tel2 go povikal Tel1

povikan(T1,T2):- telefon(T2,_,_,L),clen(povik(T1,_),L).

povikan – Врати **True** доколку во листата со појдовни повици од вториот телефон постои повик до првиот телефон.

getImePrez(Tel,Ime,Prezime):-telefon(Tel,Ime,Prezime,_).

Б) Омилен број

%omilen(X,Y)

omilen(X,Y):- maxVremetraenjeRazgovori(X,Y).

%maxVremetraenjeRazgovori(T1,T2) T2 e brojot so koj T1 ima ostvareno najgolemo vkupno traene na povici + sms

maxVremetraenjeRazgovori(X,Y):-findall((T,N),vremetraenjeRazgovori(X,T,N),[M|L]), max(L,M,Y),!.

со помош на **findall** за првиот тел најди со секој телефонски број колку вкупно трање на повици + смс има остварено. Потоа предади ја листата и првиот член на предикатот **max и** најди кој има најмногу.

%vremetraenjeRazgovori N e vkupno traene na povici + sms pomegu T1 i T2.

vremetraenjeRazgovori(T1,T2,N):-

vremetraenjePovici(T1,T2,N1),

brojPoraki(T1,T2,N2),

N is N1 + 100*N2.

vremetraenjeRazgovori – Најди вкупно времетраење на повици помеѓи телефоните пратени на влез, потоа најди го број на пораки и пресметај вкупно трање повици+смс како (времетрање повици) + 100 * (број пораки)

%vremetraenjePovici N e vkupno traene na povici pomegu T1 i T2

vremetraenjePovici(T1,T2,N):-

vremetraenjeDojdovni(T1,T2,N1),

vremetraenjeDojdovni(T2,T1,N2),

N is N1+N2.

vremetraenjePovici – Најди го времетрањето на разговорите каде вториот број го повикал првиот, и обратно времетраењето на разговорите каде првиот го повика вториот и врата ја сумата помеѓу времетраењата на разговорите.

%vremetraenjePovici N e vkupno traene na dodjovni povici pomegu T1 i T2 (T2 go povikal T1)

vremetraenjeDojdovni(T1,T2,N):- telefon(T2,_,_,L),clen(povik(T1,N),L).

vremetraenjeDojdovni(T1,T2,0):- telefon(T2,_,_,L),not(clen(povik(T1,_),L)).

vremetraenjeDojdovni – доколку во листата на појдовни повици од вториот број се појавува првиот врати го времетраењето на разговорот, спротивно врати 0.

%brojPoraki(T1,T2,N) N e brojot na poraki pomegu T1 i T2

brojPoraki(T1,T2,N):-

brojPrimeniSMS(T1,T2,N1),

brojPrimeniSMS(T2,T1,N2),

N is N1+N2.

brojPoraki – вкупниот број на пораки помеѓу двата телефони е бројот на пораки кој ги примил првиот од вториот плус бројот на пораки кој ги примил вториот од првиот.

%brojPrimeniSMS(T1,T2,N) N e brojot na poraki koj T1 gi primil od T2

brojPrimeniSMS(T1,T2,N):- bagof(_,primenaSMS(T1,T2),L), dolzina(L,N).

brojPrimeniSMS(T1,T2,0):- not(primenaSMS(T1,T2)).

brojPrimeniSMS – со **bagof** најди ги сите пораки кој првиот ги примил од вториот, подоа врати ја должината на листата. Доколку првиот не примил порака од вториот врати 0. (Забелешка1: во овој случај bagof враќа false а не празна листа, па затоа е потребо и ова правило. Забелешка2: се користи bagof а не findall бидеќи во случај кога вториот број е променлива findall го враќа бројот на сите пораки кој ги примил првиот и во нашиот случај ова однесување е погрешно).

%primenaSMS(T1,T2) True ako T2 mu ispratil poraka na T1

primenaSMS(T1,T2):- sms(T2,L), clen(T1,L).

primenaSMS – провери дали постои порака каде вториот број е испраќач а во листата на примачи се наоѓа првиот.

Задача 3

А) изброи локација

%izbroj_lokacija(Lok,Br)

izbroj_lokacija(Lok,Br):-

findall(_,(klient(_,_,_,U),clen(usluga(Lok,_,_,_,_),U)),L1),

findall(_,(klient(_,_,_,U),clen(usluga(_,Lok,_,_,_),U)),L2),

dolzina(L1,N1), dolzina(L2,N2), Br is N1+N2.

Со **findall** најди го секое појавување на локацијата како изворна во услуга која е член на листата со услуги на некој клиент и секое појавување како дестинациска. Изброј ги должините на двете листи и нивниот збир врати го како резултат.
(**забелешка:** во случај кога една локација е почетна во две услуги за ист клиент findall ќе ги врати и двете)

Б) најмногу километри

%najmnogu_kilometri(X,Y).

najmnogu_kilometri(X,Y):-

findall((S,N),pominatiKilometri(S,N),[M|L]),

max(L,M,K),

klient(K,X,Y,_),!.

Со **findall** за секој клиент во базата најди колку километри има поминато потоа предади ја листата и првиот член на предикатот **max** (дефиниран во задача 2) и најди кој има најмногу, потоа врати го неговото име и презиме.

%pominatiKilometri(X,N) N e brojot na pominati kilometri na liceto X

pominatiKilometri(X,N):-

klient(X,_,_,L),

pominatiKilometriR(L,N).

pominatiKilometri – Најди ја листата со услуги за клиентот и предади ја на предикатот **pominatiKilometriR.**

%pominatiKilometriR(L,N). L e lista od usligi a N e vkupnoto rastojanie pomegu pocetnite i destinaciskite mesta

pominatiKilometriR([],0).

pominatiKilometriR([usluga(A,B,_,_,_)|L],N):-

pominatiKilometriR(L,Nr),

najkratokPat(A,B,P), N is Nr + P.

pominatiKilometriR – Итерирај ја листата од услуги услуга по услугаи, за моменталната услуга најди го најкраткиот пат (растојанието) помеѓу почетната и крајната локација и додади на резултатот.

%najkratokPat(A,B,N) N e najkratkoto rastojanie pomegu A i B.

najkratokPat(A,B,N):-setof(R,pat(A,B,R),[N|_]).

najkratokPat - Со **setof** најди го растојанието на сите патишта од почетната до крајната локација и врати го најмалото растојание (првото бидеќи **setof** ги сортира во растечки редослед).

% D e rastojanieto pomegu A i B dokolku postoi direkten pat

patDirekten(A,B,D):-rastojanie(A,B,D).

patDirekten(A,B,D):-rastojanie(B,A,D).

%True ako postoi pat od A do B so dolzina N (bez ciklusi)

pat(A,B,N):-najdiSiteMesta(L),

izbrisi(L,A,L2),

izbrisi(L2,B,L3),

pat(L3,A,B,N).

pat(_,A,B,N):-patDirekten(A,B,N).

pat(L,A,B,N):-

clen(C,L), izbrisi(L,C,L2),

pat(L2,A,C,N1),patDirekten(C,B,N2),N is N1 + N2.

pat – Најпрво најди ги сите локации кои се појавуваат во базата потоа избри ши ги почетната и крајната од листата и прати ја на преоптоваретената верзија од предикатот **pat.**

Потоа доколку постои директен пат од почетната до крајната локација врати го неговото растојание, во спротивен случај најди локација која е член во листата со локации, избриши ја од листата, најди пат од почетната локација до новата (без да ги изминеш досега изминатите локации) и провери дали има директен пат од новата локација до крајната, доколку се е исполнето пресметај го растојанието помеѓу почетната и крајната како збир од растојанието помеѓу почетната и новата и растојанието помеѓу новата и крајната.
(Забелешка: листата со локации се користи за се води сметка кои локации се досега изминати, со цел да се избегнат циклуси).

%najdiSiteMesta(L) gi vraka site mesta za koi ima rastojanie vo bazata na podatoci

najdiSiteMesta(L):-

findall(A,rastojanie(A,_,_),La),

findall(B,rastojanie(_,B,_),Lb),

zalepi(La,Lb,Lab), izvadiDupli(Lab,L),!.

najdiSiteMesta – со **findall** најди ги сите почетни локации и сите крајни потоа спој ги двете листи и извади ги дупликатите.

%append

zalepi([],L2,L2).

zalepi([X|L1],L2,[X|L3]):-zalepi(L1,L2,L3).

%izvadiDupli(L1,L2). L2 e L1 bez duplikati

izvadiDupli([],[]).

izvadiDupli([X|L1],L2):-clen(X,L1),izvadiDupli(L1,L2).

izvadiDupli([X|L1],[X|L2]):-not(clen(X,L1)),izvadiDupli(L1,L2).

%izbrisi(L1,X,L2). izbrisi gi site pojavuvanja na X vo L1 i rezultatot vrati go vo L2

izbrisi([],_,[]).

izbrisi([X|L1],X,L2):-izbrisi(L1,X,L2).

izbrisi([X|L1],Y,[X|L2]):- X\==Y, izbrisi(L1,Y,L2).

B) најмногу заработил

%najmnogu_zarabotil(X)

najmnogu_zarabotil(X):-

findall((T,N),zarabotil(T,N),[M|L]),

max(L,M,X),!.

Со **findall** наоѓаме секое такси волизо колку заработило бо декември 2015 потоа предади ја листата и првиот член на предикатот **max** (дефиниран во задача 2) и најди кој заработил најмногу.

%zarabotil(T,N) taksistot so vozilo broj T zarabotil N vo dekemvri 2015

zarabotil(T,N):-

siteUslugi(L),

zarabotil(L,T,N).

zarabotil([],_,0).

zarabotil([usluga(A,B,C,datum(_,12,2015),T)|L],T,N):-

zarabotil(L,T,N1),

najkratokPat(A,B,P),

N is N1 + P*C.

zarabotil([usluga(_,_,_,datum(_,M,God),Taxist)|L],T,N):-

not((M\==12, God\==2015, Taxist\==T)),zarabotil(L,T,N).

zarabotil – Најпрво најди ги сите услуги во базата и листата од услуги прати ја во преоптоварената верзија на предикатот **zarabotil.**
Потоа итерирај ја листата од услуги услуга по услуга и доколку моменталната услуга е остварена во декември 2015 од такси возилото пратено на влез најди го растојанието помеѓу почетната и крајната локација и на резултатот додаи ја цената на услугата која се пресметува како поминати километри * цена од километар.
Доколку услугата не е остварена во декември 2015 или доколку такси возилото не е она пратено на влез продолжи со итерирање.
Итерирај се додека не ја изпразниш листата.

%siteUslugi(L) L e lista od site uslugi vo bazata na podatoci

siteUslugi(L):-findall(U,klient(_,_,_,U),L2),izramniMatrica(L2,L).

siteUslugi – со **findall** најди ги сите листи од услуги за секој клиент потоа со предикатот **izramniMatrica** израмни ја листата за да добиеш листа од услуги наместо лиса од листи од услуги.

%izramniMatrica(L1,L2) L2 e izramneta L1 kade L1 e matrica

izramniMatrica([],[]).

izramniMatrica([X|L],L2):-izramniMatrica(L,L3), zalepi(X,L3,L2).