ΤΕΙ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΕΡΓΑΣΤΗΡΙΟ

ΛΑΡΙΣΑ 2003

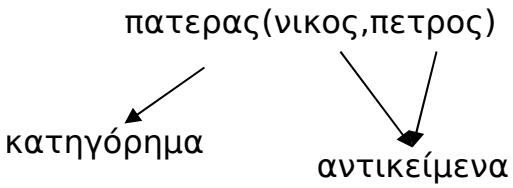
ΠΕΡΙΕΧΟΜΕΝΑ

Α. ΕΝΝΟΙΕΣ ΣΤΗΝ PROLOG

Η Prolog είναι δηλωτική γλώσσα προγραμματισμού και αποτελείται από:
ΓΕΓΟΝΟΤΑ
ΚΑΝΟΝΕΣ
ΛΙΣΤΕΣ
ΔΕΝΔΡΑ
ΑΡΧΕΙΑ

1) ΓΕΓΟΝΟΣ: είναι μια σχέση.
Αυτά τα ορίζουμε εμείς. Όμως υπάρχουν και προκαθορισμένα κατηγορήματα δηλ. γεγονότα στην Prolog.

Ένα ΓΕΓΟΝΟΣ αποτελείται από το ΚΑΤΗΓΟΡΗΜΑ (predicate) και τα ΑΝΤΙΚΕΙΜΕΝΑ ή ΟΡΟΥΣ (terms). Η δομή ενός ΓΕΓΟΝΟΤΟΣ έχει ως εξής:



ΠΡΟΣΟΧΗ :
το ΚΑΤΗΓΟΡΗΜΑ είναι πάντα με μικρά γράμματα και ενιαία λέξη χωρίς κενά
τα ΑΝΤΙΚΕΙΜΕΝΑ όταν είναι σταθερές γράφονται με μικρά
τα ΑΝΤΙΚΕΙΜΕΝΑ όταν είναι μεταβλητές γράφονται με κεφαλαίο το πρώτο γράμμα
υπενθυμίζεται ότι υπάρχει και η ανώνυμη μεταβλητή (_).

Το (,) παίζει το ρόλο του **και** ενώ στο τέλος ενός ΓΕΓΟΝΟΤΟΣ μπαίνει πάντοτε (.)

1Α) ΕΝΟΠΟΙΗΣΗ
Είναι η σύνδεση μιας μεταβλητής με ένα αντικείμενο οπότε όταν αυτή γίνει η Prolog απαντά θετικά. Η ΕΝΟΠΟΙΗΣΗ υλοποιείται με το (=). Κατά την ενοποίηση γίνεται και λογικός έλεγχος.
Άπαξ και γίνει ενοποίηση μεταβλητής με μια σταθερά τότε η μεταβλητή θεωρείται δεσμευμένη.

Σημείωση:
Για τις πράξεις χρησιμοποιείται το is
Μόνο λογικό έλεγχο μεταξύ αντικειμένων μπορούμε να κάνουμε με τη χρήση του ==

2) ΚΑΝΟΝΕΣ
Ισχύει κάτι όταν ισχύει κάτι άλλο. Δηλαδή ένα ΓΕΓΟΝΟΣ ισχύει όταν ισχύει κάποιο άλλο ή άλλα.

ΓΕΓΟΝΟΣ1 :- ΓΕΓΟΝΟΣ2, ΓΕΓΟΝΟΣ3.

Με τους ΚΑΝΟΝΕΣ δηλώνουμε πληροφορίες, δεδομένα (βάση γνώσης, βάση γεγονότων).
γιαγια(X,Y):-μαμα(X,Z),μαμα(Z,Y).

Το (:-) είναι το **όταν**.

Στον τρόπο δήλωσης των ΚΑΝΟΝΩΝ στη Prolog έγκειται η διαφοροποίηση του δηλωτικού της χαρακτήρα από τις διαδικαστικές γλώσσες προγραμματισμού (procedural).

Παράδειγμα διαδικαστικού
Θέλω να βρω κάποιο δρόμο: ΠΑΡΕ, ΠΕΡΠΑΤΑ, ΣΤΡΙΨΕ κτλ
χρήση εντολών

Θέλω να εξηγήσω ποιο κτίριο είναι το δημαρχείο: ΕΙΝΑΙ, ΕΧΕΙ, ΒΡΙΣΚΕΤΑΙ
χρήση δηλώσεων

Γενικά
θνητός(Χ):-ανθρωπος(Χ).
Ο Χ είναι θνητός όταν ο Χ είναι άνθρωπος
Αν ο Χ είναι άνθρωπος τότε είναι θνητός

Η σύνδεση κατά έναν ΚΑΝΟΝΑ μεταξύ γεγονότων γίνεται με (,) και η ένωση γεγονότων με (;).

Έτσι, Ο Χ είναι πλούσιος όταν είναι μηχανικός ή τοπογράφος και έχει αυτοκίνητο.
Γράφεται σαν κανόνας
πλουσιος(Χ)**:-**
μηχανικος(Χ)**;**
τοπογραφος(Χ)**,**
εχει(Χ,αυτοκινητο)**.**

1Β) Εξαιτίας, των ΚΑΝΟΝΩΝ και των υφιστάμενων ΓΕΓΟΝΟΤΩΝ υπάρχουν στην Prolog τα φαινόμενα
της ΑΝΑΠΤΥΞΗΣ → οδηγεί σε ενοποίηση μεταβλητών.
και ΕΠΑΝΑΔΡΟΜΗΣΗΣ → οδηγεί σε απελευθέρωση μεταβλητών.

Εξαιτίας αυτών ισχύει στη Prolog το εξής:
Μπορούμε να βρούμε όλες τις πιθανές λύσεις για ένα ερώτημα και αυτή η διαδικασία που συνδυάζει τα δύο προηγούμενα φαινόμενα λέγεται ΜΗ ΠΡΟΣΔΙΟΡΙΣΤΙΚΟΤΗΤΑ.

Η ΜΗ ΠΡΟΣΔΙΟΡΙΣΤΙΚΟΤΗΤΑ της Prolog αναιρείται με την ΤΟΜΗ (!).
Δηλαδή μόλις βρει η Prolog μια λύση να σταματά τον έλεγχο στα άλλα ΓΕΓΟΝΟΤΑ.
Η ΤΟΜΗ δεν πρέπει να γίνεται πολύ νωρίς γιατί αποκλείουμε λύσεις, ούτε πολύ αργά γιατί επιβαρύνουμε την Prolog με παραπάνω εξερευνήσεις.

2Β) FAIL
Με τη χρήση αυτού πετυχαίνουμε άρνηση και επαναδρόμηση. Είναι αντίστοιχο του προκαθορισμένου κατηγορήματος not().

3) ΛΙΣΤΕΣ
Είναι συμπλέγματα αντικειμένων
Δηλ. [α,γ,η,ξ].
Μία λίστα έχει κεφαλή και ουρά.
Πχ στην προηγούμενη λίστα ΚΕΦΑΛΗ: α, ΟΥΡΑ:[γ,η,ξ].
Ή αλλιώς μια λίστα μπορεί να δηλωθεί ως [Χ|Υ]

4)ΔΕΝΔΡΑ
Ένα δένδρο απεικονίζεται ως εξής

F(X,g(A,b)).

Ότι ισχύει για τις μεταβλητές σχετικά με την ενοποίηση, ισχύει για τις λίστες και τα δένδρα ανά στοιχείο τους.

ΤΕΛΕΣΤΕΣ
Είναι κατηγορήματα τα οποία εμείς χρησιμοποιούμε για να υλοποιούμε δικές μας πράξεις.

ΠΡΟΚΑΘΟΡΙΣΜΕΝΑ ΚΑΤΗΓΟΡΗΜΑΤΑ ΚΑΙ ΚΑΤΗΓΟΡΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗ ΔΟΜΗΜΕΝΩΝ ΟΡΩΝ
write, read, get, get0, nl, assert, asserta, assertz, retract, not
functor, arg, name

ΧΡΗΣΙΜΕΣ ΣΥΜΒΟΥΛΕΣ

1. Ευθυγράμμιση
2. Εισαγωγή σχολίων /* */ ή %
3. Προσοχή στα (.) (,) (;) (:-)
4. Προσοχή στις μεταβλητές και στις σταθερές
5. Για τη λήψη όλων των αποτελεσμάτων της ενοποίησης χρήση του πλήκτρου ; και όχι του enter.

Β. ΕΙΣΑΓΩΓΙΚΕΣ ΕΝΝΟΙΕΣ - ΠΑΡΑΔΕΙΓΜΑΤΑ

Εργασία 1η: ΠΡΟΚΑΘΟΡΙΣΜΕΝΑ ΚΑΤΗΓΟΡΗΜΑΤΑ
Να γραφεί στο περιβάλλον της prolog:

X=Y, Y=a.	Press enter
Y=K, K=L, L=d.	Press enter
Z=U, U=V, V=H, H=a, E=Z, E=d.	Press enter
A is 7+3, B is 2*A.	Press enter
B=0, B is 2*9.	Press enter
A is 8/2, B is A+1, A=<B.	Press enter
A < 7.	Press enter
atom(15).	Press enter
atom(petros).	Press enter
X=Y, Y=A, atom(X).	Press enter
atom(X).	Press enter
integer(15).	Press enter
integer(petros).	Press enter
X=a, integer(X).	Press enter
integer(X).	Press enter
X=1, Y=6, Z is Y+X, integer(15).	Press enter
X=Y, var(X).	Press enter
X=a, var(X).	Press enter
var(a).	Press enter
var(23).	Press enter
X=S, S=D, O=D-X, var(O).	Press enter
X==Y.	Press enter
X=Y, X==Y.	Press enter
X=a, Y=a, X==Y.	Press enter
X=a, Y==X.	Press enter

Εργασία 2η: **αρχείο:** agones.pl
Έχουμε ποδοσφαιρικούς αγώνες μεταξύ κάποιων ελληνικών ομάδων και τα τελικά σκορ αυτών.
(παράδειγμα χρήσης ανισότητας)

```
agonas(pao,osfp,0,0).
agonas(aek,pao,0,6).
agonas(osfp,aek,5,0).
agonas(osfp,pao,6,6).
agonas(aris,aek,0,0).
agonas(aek,osfp,0,1).
agonas(osfp,aris,2,2).
agonas(aek,aris,2,3).
```

Εκτελέσατε τα ερωτήματα,
1. Βρείτε και εμφανίστε όλες τις ισοπαλίες
2. Ποια ομάδα έφερε ισοπαλία με την ΑΕΚ εντός έδρας και έβαλε εκτός έδρας 2 γκολ στον ολυμπιακό.
3. Πόσες ισοπαλίες έφερε ο παναθηναϊκός εντός έδρας.
4. Πόσες νίκες είχε ο Ολυμπιακός εντός έδρας και με ποιες ομάδες.

Εργασία 3η: **αρχείο:** pelates.pl
Έχουμε πελάτες ξενοδοχείου με ημέρα άφιξης και ημέρα αναχώρησης.

```
pelatis(potamianos,deytera,pempti).
```

```
pelatis(mhlarakis,kyriaki,kyriaki).
pelatis(gribas,triti,tetarti).
pelatis(karapoulios,deytera,deytera).
pelatis(fwtiadis,tetarti,paraskeyi).
```

Εκτελέσατε τα ερωτήματα,
1. Ποιοι πελάτες έφθασαν και έφυγαν την ίδια μέρα.
2. Ποιος πελάτης ήρθε Τρίτη και ποιος ήρθε την ημέρα αναχώρησης του πρώτου.

Εργασία 4η: **αρχείο:** gonios.pl
Έχουμε τους γονείς για κάθε άτομο. (παράδειγμα χρήσης του _)

```
male(bill).
male(joe).

female(sue).
female(tammy).

parent(bill,joe).
parent(sue,joe).
parent(joe,tammy).
```

Εκτελέσατε τα ερωτήματα,
1. Βρες τα άτομα που έχουν γονιό και ανέφερε και το όνομα του.
2. Ποια από τα άτομα έχουν γονιό.

Γ. ΚΑΝΟΝΕΣ

Εργασία 5α: **αρχείο:** praxeis.pl

Για την κατανόηση των κανόνων και χρήση των πράξεων γράψτε:
Για τους κανόνες ισχύει κάνε πράξη μεταξύ δύο αριθμών όταν το αποτέλεσμα αυτής καθορίζεται από συγκεκριμένο τύπο.

```
sumurize(X,Y,Sum):-
        Sum is X+Y.
minus(X,Y,Min):-
        Min is X-Y.
multiply(X,Y,Product):-
        Product is X*Y.
divide(X,Y,Div):-
        Div is X/Y.
```

Εκτελέσατε τα ερωτήματα,
1. Βρες το άθροισμα 32 και 54.
2. Βρες τον πολλαπλασιασμό του 32 με το 54.
3. Βρες την αφαίρεση του 32 με το 54.
4. Βρες τη διαίρεση του 67 με το 5.

Εργασία 5b: **αρχείο:** numb_let.pl

Ορίζουμε ότι τα γράμματα είναι μεταξύ a και z σε μικρά ή κεφαλαία και επίσης ένα αριθμητικό διάστημα.

```
isletter(Ch):-
        Ch>="a",
        Ch=<"z".
isletter(Ch):-
        Ch>="A",
        Ch=<"Z".
belongs(Ch):-
        Ch>=0,
        Ch=<255.
```

Εκτελέσατε τα ερωτήματα,
1. Είναι το x γράμμα, Είναι το 2 γράμμα
2. Είναι το hello γράμμα, Είναι το A γράμμα
3. Ανήκει το 90 στο διάστημα αριθμών
4. Ανήκει το 256 στο διάστημα αριθμών.

Εργασία 5c: **αρχείο:** aniso.pl

Έστω ένας μαθητής ο οποίος πήρε τρεις βαθμούς Β1, Β2, Β3 και θέλουμε να δούμε αν περνάει τη βάση και σε κανένα μάθημα να μην έχει 0.

```
o_mathitis_pernaei(B1,B2,B3):-
B1>0,
B2>0,
B3>0,
M is (B1+B2+B3)/3,
M>=10.
```

Έτσι αν α. ο Γιώργος πήρε 11, 12, 8 τι γίνεται;
β. ο Τάκης πήρε 10, 0 ,20 τι γίνεται;

Εργασία 6η: **αρχείο:** kanones1.pl

Έχουμε άτομα με τα χόμπι τους. Τα γεγονός έχει τη δομή:
Στο παράδειγμα ισχύει ο κανόνας ότι αρέσει στον tom αρέσει και στον bill

```
likes(ellen,tennis).
likes(john,football).
likes(tom,baseball).
likes(eric,swimming).
likes(mark,tennis).

likes(bill,Activity):-
  likes(tom, Activity).
```

Εκτελέσατε τα ερωτήματα,
1. Τι αρέσει στον Μπιλ.

Εργασία 7η: **αρχείο:**kanones2.pl

Ανάλογα με το τι δηλώνεται σε άλλα γεγονότα μπορώ να κάνω ερωτήσεις σε ένα άλλο γεγονός το οποίο θα ισχύει μόνο αν ισχύουν τα προηγούμενα.

```
male(john).
male(fred).
male(harry).

female(mary).
female(julie).
female(susan).
female(anne).

blonde(john).
dark(harry).
dark(fred).

brunnete(mary).
brunnete(anne).
blonde(susan).
blonde(julie).

likes(john,Person):-
female(Person),
blonde(Person),
rich(Person).

likes(fred,Person):-
female(Person),
brunnete(Person).

likes(harry,Person):-
female(Person),
rich(Person).

likes(mary,Person):-
male(Person),
dark(Person).

likes(julie,Person):-
male(Person),
dark(Person),
rich(Person).

rich(Person):-
```

owns(Person,gold).

owns(fred,gold).
owns(john,car).
owns(julie,gold).
owns(anne,house).

Εκτελέσατε τα ερωτήματα,
1. Ποιοι είναι άνδρες.
2. Τι αρέσει στο john, στη mary.
3. Αρέσει κάτι στη julie.
4. Υπάρχει περίπτωση δύο άτομα να αληλλοσυμπαθιούνται

Εργασία 8η: **αρχείο:**enosi.pl
Μπορούμε να δούμε την εφαρμογή ενός άλλου είδους κανόνων όπου ισχύει ένα γεγονός το οποίο αποτελείται από δύο άλλα. Όχημα είναι και το τρακτέρ και το αμάξι.

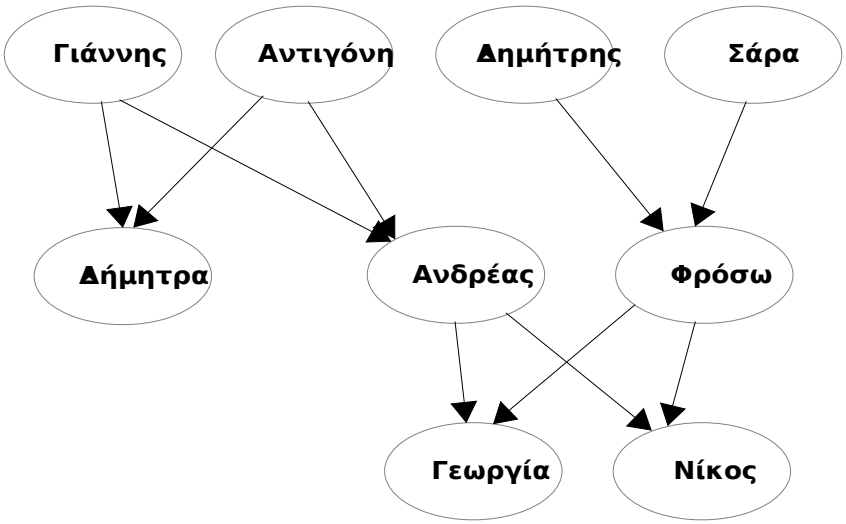
car(chrysler,130000,3,red,12000).
car(ford,90000,4,gray,25000).
car(datsun,8000,1,red,30000).

truck(ford,80000,6,blue,8000).
truck(datsun,50000,5,orange,20000).
truck(toyota,25000,2,black,25000).

vehicle(Make,Odometer,Age,Color,Price):-
car(Make,Odometer,Age,Color,Price)
;
truck(Make,Odometer,Age,Color,Price).

Εκτελέσατε τα ερωτήματα,
1. Ποιο όχημα γενικά έχει τιμή 25000 Euro.
2. Τι θα γίνει αν ρωτήσω car(renault,13,40000,red,12000).
3. Τι θα γίνει αν ρωτήσω car(ford, 90000, gray, 4, 25000).
4. Τι θα γίνει αν ρωτήσω car(1, red, 30000, 80000, datsun).

Εργασία 9η: **αρχείο:** oikogeneia.pl
Έχουμε άτομα με τους γονείς τους και προσπαθούμε χρησιμοποιώντας γεγονότα βασιζόμενα σε δύο βασικά να διαγνώσουμε διάφορες συγγενικές σχέσεις σύμφωνα με το παρακάτω γενεαλογικό δέντρο.



```
pateras(nikos, andreas).
pateras(andreas, giannis).
pateras(frosw, dimitris).
pateras(gewrgia, andreas).
pateras(dimitra, giannis).

mitera(gewrgia, frosw).
mitera(nikos, frosw).
mitera(andreas, antigoni).
mitera(frosw, sara).
mitera(dimitra, antigoni).

giagia(E1,G1):-
 mitera(E1,M1),
 mitera(M1,G1).

giagia(E2,G2):-
 pateras(E2,P2),
 mitera(P2,G2).

papous(E3,PA3):-
 pateras(E3,P3),
 pateras(P3,PA3).

papous(E4,PA4):-
mitera(E4,M4),
pateras(M4,PA4).

theia(E5,U5):-
pateras(E5,P5),
pateras(P5,PA5),
pateras(U5,PA5).

adelfia(A1,A2):-
 pateras(A1,O),
 mitera(A1,J),
 pateras(A2,O),
 mitera(A2,J),
A1/=A2.
```

Εκτελέσατε τα ερωτήματα,
1. Ποιου είναι θεία η Δήμητρα
2. Ποιος ο παππούς του Νίκου
3. Ποια η γιαγιά της Γεωργίας
4. Ποιος έχει γιαγιά και παππού. Τι πρόβλημα υπάρχει;
5. Ποιος ο αδελφός της Δήμητρας
6. Τι θα συμπληρώνατε ώστε να μπορούσε κάποιο άλλο άτομο να έχει θεία τη Δήμητρα και παππού κάποιον άλλο από του προαναφερθέντες.

Εργασία 10η: **αρχείο:**pa8isi.pl

Δημιουργείστε ένα έμπειρο σύστημα το οποίο ανάλογα τα συμπτώματα που έχει ένας ασθενής να διαγιγνώσκει την πάθηση.

```
has_symptom(sofia,diaroia).
has_symptom(sofia,emeto).
has_symptom(giorgos,ponokefalo).
has_symptom(dimitris,ponokoilo).
```

```
has_symptom(dimitris,pyreto).
has_symptom(zenia,ponokoilo).
has_symptom(zenia,pyreto).
has_symptom(dionisis,ponokoilo).

pathisi(X,kourasi) :- has_symptom(X,ponokefalo).

pathisi(X,griph) :- has_symptom(X,pyreto), !, has_symptom(X,ponokoilo).

pathisi(X,krywma) :- has_symptom(X,ponokoilo), !.

pathisi(X,dhlhthriasi) :- has_symptom(X,diaroia), has_symptom(X,emeto).
```

Εκτελέσατε τα ερωτήματα,
1. Να βρεθεί η πάθηση της sofia, dimitris, zenia, dionisis
2. Να βρεθεί ποιος έχει griph;
3. Να βρεθούν όλες οι παθήσεις για όλα τα άτομα
4. Τι γίνεται άμα βγουν τα θαυμαστικά;
5. Τι γίνεται άμα μπει θαυμαστικό στον πρώτο κανόνα;

Εργασία 11η: **αρχείο:**fail.pl

Τα δύο παρακάτω προγράμματα δεν διαφέρουν σε τίποτα απλά μπορούμε να δούμε τα ίδια πράγματα να δηλώνονται με fail και not.

```
age(petros,30).
age(larian,25).
age(kostas,57).
age(na8anail,30).
age(iakovos,56).
age(ilarios,46).

service(kostas,26).
service(petros,10).
service(na8anail,19).
service(iakovos,24).
service(larian,5).
service(ilarios,21).

manager(larian).
manager(ilarios).

shopfloor(petros).
shopfloor(iakovos).
shopfloor(na8anail).
shopfloor(kostas).

wage(petros,4500).
wage(iakovos,6500).
wage(na8anail,5600).
wage(kostas,6700).
```

A.
```
synta3i(Employee):-
service(Employee,Years),
Years>20,
ikanos(Employee).

ikanos(Employee):-
```

B.
```
synta3i(Employee):-
service(Employee,Years),
Years>20,
ikanos(Employee).

ikanos(Employee):-
```

```
age(Employee,Years),
Years<50,
!,fail.

ikanos(Employee):-
manager(Employee),
!,fail.

ikanos(Employee):-
shopfloor(Employee),
wage(Employee,Pay),
Pay<15000,
Pay>5000.
```

```
age(Employee,Years),
not(Years<50),
not(manager(Employee)),
shopfloor(Employee),
wage(Employee,Pay),
Pay<15000,
Pay>5000.
```

Εκτελέσατε τα ερωτήματα,

1. Ποιοι υπάλληλοι μπορούν να πάρουν σύνταξη.

Εργασία 12η: **αρχείο:**fail2.pl

```
filovolo(kerasia).
filovolo(sykia).

ai8ali(peyko).
ai8ali(kyparisi).
ai8ali(elato).

filovolo(X):-
ai8ali(X),
!,fail.

filovolo(_).
```

Ερωτήματα

1. Είναι φυλλοβόλο η συκιά;
2. Ποια είναι αειθαλή;

Γράψτε το παραπάνω πρόγραμμα με τη χρήση της Not.

Εργασία 13η: **αρχείο:**write_read.pl

Πρόγραμμα χρήσης των εντολών write και read.

```
pateras(john,jim).
pateras(spyros,nik).
pateras(alkis,menelaos).
mitera(john,helen).
mitera(spyros,sofia).
mitera(alkis,nefeli).

read_frase(Fourth, Sixth):-
read(_),
read(_),
read(_),
read(Fourth),
read(_),
read(Sixth).


question:-
write('Hello, write your search:'),
read_frase(Parent,Name_of_child),
find(Parent, Name_of_child, Name_of_parent),
write('The solution is:'),
nl,
write(Name_of_parent).

find(patera, Child, Name_of_parent):-
pateras(Child, Name_of_parent),
!.

find(pateras, Child, Name_of_parent):-
pateras(Child, Name_of_parent),
!.

find(mitera, Child, Name_of_parent):-
mitera(Child, Name_of_parent),
```

!.

find(_, _, agnwstos).

Εκτελέσατε τα ερωτήματα,
question. και
1. poia. einai. i. mitera. toy. john.
2. dwse. mou. ton. patera. toy. alkis.
3. poios. einai. o. pateras. tou. menelaos.
4. pes. mou. parakalw. poia. einai. i. mitera.toy. john.
5. poios. einai. o. pateras. tou. chris.
6. Τι γίνεται άμα μπει θαυμαστικό στο

find(patera, Child, Name_of_parent):-
!,
pateras(Child, Name_of_parent).

Και εκτελέσουμε poios. einai. o. pateras. tou. chris.

Εργασία 14η:
Στο περιβάλλον της Prolog εισάγεται τα παρακάτω.
assert(male(john)).
assert(female(nadia)).
assert(human(X):-female(X)).
assert(human(Y):-male(Y)).

Εκτελέσατε τα ερωτήματα,
1. male(C).
2. female(G).
3. female(chris).
4. human(F).

Εργασία 15η: **αρχείο:r_w_as_re**.pl
Δείτε την εργασία 11 με τον παρακάτω τρόπο χρησιμοποιώντας τις εντολές assert και retract:

```
learn_sth:-
write('What do you want me to learn?'),
nl,
read(_),
read(Parent),
read(_),
read(Child),
read(_),
read(_),
read(Name_of_Parent),
learn(Parent, Child, Name_of_Parent),
write('OK').

learn(pateras, Child, Name_of_father):-!,
assert(pateras(Child, Name_of_father)).
learn(patera, Child, Name_of_father):-!,
assert(pateras(Child, Name_of_father)).

learn(mitera, Child, Name_of_mother):-!,
assert(mitera(Child, Name_of_mother)).
read_frase(Fourth, Sixth):-
read(_),
read(_),
```

```
read(_),
read(Fourth),
read(_),
read(Sixth).

question:-
write('Hello, write your search:'),
read_frase(Parent,Name_of_child),
find(Parent, Name_of_child, Name_of_parent),
write('The solution is:'),
nl,
write(Name_of_parent).

find(patera, Child, Name_of_parent):-
pateras(Child, Name_of_parent),
!.
find(pateras, Child, Name_of_parent):-
pateras(Child, Name_of_parent),
!.
find(mitera, Child, Name_of_parent):-
mitera(Child, Name_of_parent),
!.

find(_, _, agnwstos).

delete_sth:-
write('What do you want me to delete?'),
nl,
read(_),
read(Parent),
read(_),
read(Child),
read(_),
read(_),
read(Name_of_Parent),
delete(Parent, Child, Name_of_Parent),
write('OK. It`s done').

delete(pateras, Child, Name_of_father):-!,
retract(pateras(Child, Name_of_father)).
delete(patera, Child, Name_of_father):-!,
retract(pateras(Child, Name_of_father)).

delete(mitera, Child, Name_of_mother):-!,
retract(mitera(Child, Name_of_mother)).
```

Εκτελέσατε τα ερωτήματα,

```
1. learn_sth. και
o. pateras. toy. alkis. einai. o. spyros. και
question. και
dwse.mou. ton. patera. tou. alkis.
2. learn_sth. και
o. theios. toy. alkis. einai. o. george. και
question. και
poios. einai. o. theios. tou. alkis.
3. learn_sth. και
o. pateras. toy. dimitri. nomizw. oti. einai. o. spyros. και
question. και
dwse.mou. ton. patera. tou. dimitri.
```

4. delete_sth. Και
oti. pateras. toy. alkis. einai. o. spyros. και
question. και
dwse.mou. ton. patera. tou. alkis.

Εργασία 16η: Υλοποίηση μετρητή **αρχείο:**counter.pl

```
xroma(orxidea,black).
xroma(rose,red).
xroma(kyklamino,pink).
xroma(toulipa,red).
xroma(garyfalo,red).
xroma(giasemi,white).

count(X):-
xroma(_,X),
retract(counter(M)),
Mn is M+1,
assert(counter(Mn)),
fail.

count(_).

counttell(X):-
assert(counter(0)),
count(X),
retract(counter(T)),
write('we have '),
write(T),
write(' '),
write(X),
write('flowers').
```

Ερωτήματα

1. Μέτρα και πες μου πόσα λουλούδια έχουν κόκκινο χρώμα.
2. Αν εκτελέσουμε xroma(X,red). Τι θα γίνει; Ποια η διαφορά με το συνολικό πρόγραμμα.

Εργασία 17η: Εφαρμογή εισαγωγής δεδομένων

Η παρακάτω εφαρμογή η οποία θα γραφτεί στο Notepad έχει ως στόχο να αναδείξει πως μπορούμε με τρόπο ερωτοαποκρίσεων με την Prolog, να δημιουργήσουμε νέα γεγονότα που σχετίζονται με το φύλο και το όνομα κάποιων ατόμων. Αυτά τα γεγονότα δεν προϋπάρχουν στον κώδικα της εφαρμογής.

Σε αυτό μας βοηθά το προκαθορισμένο γεγονός της Prolog assertz. Οπότε και έτσι εισάγεται το κάθε νέο γεγονός στο τέλος της βάσης γνώσης που δημιουργείται από την εφαρμογή, όταν τρέχει σε περιβάλλον Prolog.
Το προκαθορισμένο γεγονός asserta εισάγει κάθε νέο γεγονός στην αρχή της βάσης γνώσης. Αυτή είναι η διαφορά του από το assertz.
Το προκαθορισμένο read διάβαζει και αποθηκεύει το τι απαντάμε εμείς στην κάθε ερώτηση που μας κάνει η prolog.

```
initialise:-
write('What is your name?'),
nl,
read(Name),
assertz(user(Name)),
```

```
write('Are you male or female?'),
nl,
read(Gender),
add_gender_fact(Gender,Name).

add_gender_fact(male,Name):-
assertz(male(Name)).

add_gender_fact(female,Name):-
assertz(female(Name)).
```

Το γεγονός add_gender_fact στην ουσία δημιουργεί με τη βοήθεια της assertz τα νέα γεγονότα στην βάση γνώσης.

Ερώτηση: εκτελέστε την ιnitialize και εισάγετε άτομα με το φύλο τους και έπειτα βρείτε ποια από τα άτομα που εισάγατε είναι άνδρες.

Εργασία 18α: Υπολογισμός παραγοντικού **αρχείο:**paragontiko.pl

```
paragontiko(1,1):-!.

paragontiko(G,P):-
Gn is G-1,
paragontiko(Gn,Pn),
P is G*Pn.
```

Ερωτήματα
Βρες το παραγοντικό του 3, 9, 78, 67.

Εργασία 18β: Υπολογισμός δύναμης αριθμού **αρχείο:**dinami.pl

```
power(_,0,1):-!.
power(X,Y,Z):-
L is Y-1,
power(X,L,M),
Z is M*X.
```

Ερωτήματα
1.Υπολογίστε το 5^5.

Εργασία 18γ: Υπολογισμός αριθμητικής προόδου **αρχείο:**dinami.pl

```
fib(1,1):-!.
fib(2,1):-!.

fib(Number,Term):-
Numb1 is Number-1,
Numb2 is Number-2,
fib(Numb1,Term1),
fib(Numb2,Term2),
Term is Term1+Term2.
```

Ερωτήματα
1.Υπολογίστε την αριθμητική πρόοδο του 6 και μετά του 76.

Α. ΛΙΣΤΕΣ

Εργασία 19η: Εισαγωγή στις λίστες
Σε περιβάλλον Prolog εκτελέσατε:
[a,b] = [a,H,K].
[a,b]=[X,Y].
[a,j]=[a,k].
[a,b]=[X,X].
[X,a,Y]=[[a,b],Z,[e,n]].
[_]=[a,n].
[_,_]=[a,n].
[X,X,X]=[[a],[a],[a]].
[[X,Y],[Z,b]]=[[Z,a],[Y,X]].

Επίσης:
[a]=[X|Y].
[A|B]=[_,_].
[a,b,c]=[A|_].
[a|X]=[Y|b].
[X,Y]=[].
[a,b,c,d]=[a,X|Y].
[[a,b]|X]=[[A,b],[c,d]].
[[A|B],[C|D]]=[[a,b,c],[d]].

Προσοχή στο [X,Y|Z,W] δε γράφουμε ποτέ έτσι
Προσέξτε επίσης τα:
[X|Y|Z] = [a,b,c,d].
[X|[Y|W]] = [a,b,c,d].

Τέλος, γράψτε:
L=[C|H], C=[a,h], H=[g].
Με τέτοιο τρόπο κατασκευάζω μια λίστα

Εαν γράψω X=[a|X]. δημιουργώ κυκλική λίστα

Εάν γράψω L=[e|An], An=[g|Am], Am=[h|As], As=[]. Δημιουργήσαμε μια ενοποίηση.

Εργασία 20η: Πράξεις σε λίστες

1. Αναγραφή όλων των στοιχείων λίστας

```
write_a_list([]).
write_a_list([H|T]):-
write(H),nl,
write_a_list(T).
```

Ερώτηση: write_a_list([1,2,3]).

2. Πότε ανήκει ένα στοιχείο σε μια λίστα

```
belongs(X,[X|_]):-!.
belongs(X,[_|Z]):-belongs(X,Z).
```

Ερώτηση:belongs(b,[g,b,j]).

Επίσης , εμφάνιση του ενός στοιχείου λίστας
nth_member(1,[M|_],M).

```
nth_member(N,[_|T],M):-N>1, N1 is N-1, nth_member(N1,T,M).
```

Ερώτηση: nth_member(6,[g,b,j,u,k,o],K).

3. Βρες το πρώτο και το τελευταίο στοιχείο μιας λίστας

```
prwto(B,[B|_]).
teleytaio(L,[L]):-!.
teleytaio(L,[_|T]):-teleytaio(L,T).
```

Ερωτήματα: Βρείτε το teleytaio(K,[h,j,k,l,h,g,f]).

4. Απαρίθμησε από το πρώτο ως το τελευταίο τα στοιχεία μιας λίστας

```
prefix([],_).
prefix([H|T1],[H|T2]):-prefix(T1,T2).
suffix(S,S).
suffix([_|T],L):-suffix(T,L).
```

Ερωτήματα: prefix(L,[h,j,k,l,h,g,f]).
suffix([h,j,k,l,h,g,f],S).

5. Μέτρα πόσα στοιχεία έχει μια λίστα.

1ος τρόπος: τερματική επιστροφή

```
length_of([],Result,Result):-!.
length_of([_|T],Result,Counter):-
Ncounter is Counter+1,
length_of(T,Result,Ncounter).
```

Ερώτηση: length_of([a,g,n],X,0).

2ος τρόπος: αρχική επιστροφή

```
length_of([],0):-!.
length_of([_|T],L):-
length_of(T,TailLength),
L is TailLength + 1.
```

Ερώτηση: length_of([a,g,n],X).

6. Άθροισμα στοιχείων λίστας - *αρχική επιστροφή* – ***παράδειγμα στην τάξη***

```
sum([],0,0):-!.
sum([K|O],Sum,N):-
sum(O,S1,N1),
Sum is K+S1, N is 1+N1.
```

Ερώτηση: sum([3,4,5],N,_).

7. Προσθήκη του 1 σε όλα τα μέρη μιας λίστας

```
add([],[]):-!.
add([H|T],[H1,T1]):-
H1 is H+1,
add(T,T1).
```

Ερώτηση: add([3,5,9],N).
Τι πρέπει να αλλάξω για προσθήκη του 2.

8. Βάλε τόσα a σε μια λίστα.

```
list([],0):-!.
```

```
list([a|X],N):-
N1 is N-1,
list(X,N1).
```

Ερωτήματα: list(H,4).

9. Αφαίρεσε όλα τα a από μια λίστα

```
afairw(_,[],[]):-!.

afairw(X,Z,[X|W]):-
!,
afairw(X,Z,W).

afairw(X,[A|B],[A|W]):-
afairw(X,B,W).
```

Ερωτήματα:
afairw(a,L,[f,a,d,a,f]).
afairw(c,B,[v,c,v,c,c,k]).
afairw(d,K,[b,h,d,d,d,r]).

10. Αφαίρεση από λίστα αρνητικών στοιχείων

```
positive([],[]).
positive([H|T],P):-
H<0,
!,
positive(T,P).

positive([H|T],[H|P]):-
positive(T,P).
```

Ερωτήματα: positive([3,-9,-7,9,70],C).

11. Διπλασίασε τα στοιχεία μιας λίστας

```
double([],[]):-!.
double([H|J],[H,H|Dotail]):-
double(J,Dotail).
```

Ερωτήματα: double([x,-9,7,g,a],C).

12. Ένωσε δύο λίστες.

```
syndese([],List,List):-!.
syndese([H|L1],List2,[H|L3]):-
syndese(L1,List2,L3).
```

Ερωτήματα:
1. syndese([9,7,g,a],[5,7,k],C).
2. syndese(L1,L2,[a,f,g]).
Ποια η διαφορά εάν δεν υπήρχε η τομή στο πρώτο κανόνα
3. syndese(L3,[l,k],[h,k,l,k]).

13. Αντιστροφή λίστας

```
antistrofi([],L,L):-!.
antistrofi([A|B],L1,L):-antistrofi(B,[A|L1],L).
```

Ερωτήματα:
antistrofi([a,b,c],[],L).

Πιο κατανοητό το ίδιο παράδειγμα

```
revert(List,RevList):-
rev(List,[],RevList).

rev([H|T],S,R):-
rev(T,[H|S],R).
rev([],R,R):-!.
```

14. Παραγωγή όλων των πιθανών συνδυασμών των στοιχείων μιας λίστας

```
perm([],[]).
perm([H|T],Perm):-
perm(T,SP),insert(H,SP,Perm).

insert(X,T,[X|T]).
insert(X,[H|T],[H|NT]):-
insert(X,T,NT).
```

Ερωτήματα: perm([1,2,3],P).

15. Δημιουργία των υπολιστών μιας λίστας

```
enosi([],L,L).
enosi([H|T],L,[H|LT]):-enosi(T,L,LT).
prefix(P,L):-enosi(P,_,L).
suffix(S,L):-enosi(_,S,L).
sublist(S,L):-enosi(_,S,P),enosi(P,_,L).
sublist2(S,L):-prefix(P,L),suffix(S,P).
sublist3(S,L):-prefix(S,L).
sublist3(S,[_|T]):-sublist3(S,T).
```

Ερωτήματα:
sublist(K,[a,a,d,f,g,j]). είναι το ίδιο με το sublist2(K,[a,g,h,j,j]).

Το sublist3(K,[h,j,k,o,k]). Διαφέρει από τα προηγούμενα.

Εργασία 21η: Εφαρμογή υπολογισμού μέσου όρου

Με τη χρήση του average πετυχαίνουμε την κλήση τριών άλλων γεγονότων τα οποία μας βοηθούν στον υπολογισμό του μέσου όρου όσων αριθμών δώσουμε εμείς.

Το init παρουσιάζει μια περιγραφή της εφαρμογής και των ερωτήσεων που θα θέσει σε εμάς η Prolog, οι οποίες είναι σχετικές με το πόσους αριθμούς θα εισάγουμε για να βρούμε το μέσο όρο τους. Επίσης αυτό καλεί το input. Με τη σειρά του το init δημιουργεί τη λίστα των αριθμών τους οποίους εμείς εισάγουμε για να υπολογισθεί εκ των υστέρων ο μέσος όρος τους.

Έπειτα καλείται το calc το οποίο στην ουσία παίρνει τη λίστα η οποία δημιουργήθηκε πριν, την αθροίζει μέσω του sum και βρίσκει τον μέσο όρο της.

Τέλος, καλείται το output το οποίο και επιστρέφει τον μέσο όρο των αριθμών που δώσαμε εμείς αρχικά.

```
average:-
init(N,List),
calc(N,List,Ave),
output(Ave).

init(N,List):-
nl,nl,
write('Program to calculate the average'),
```

```
nl,
write('of a list of numbers'),
nl,
nl,
write('How many numbers ? '),
read(N),
nl,
write('Now type in '),
write(N),
write(' numbers '),
nl,nl,
input(N,List).

calc(N,List,Ave):-
sum(List,Sum),
Ave is Sum/N.

output(Ave):-
nl,nl,
write('the average of your numbers is '),
write(Ave),
nl,nl.

input(0,[]):-!.
input(N,[H|T]):-
read(H),
M is N - 1,
input(M,T).

sum([],0):-!.
sum([H|T],S):-
sum(T,S1),
S is H + S1.
```

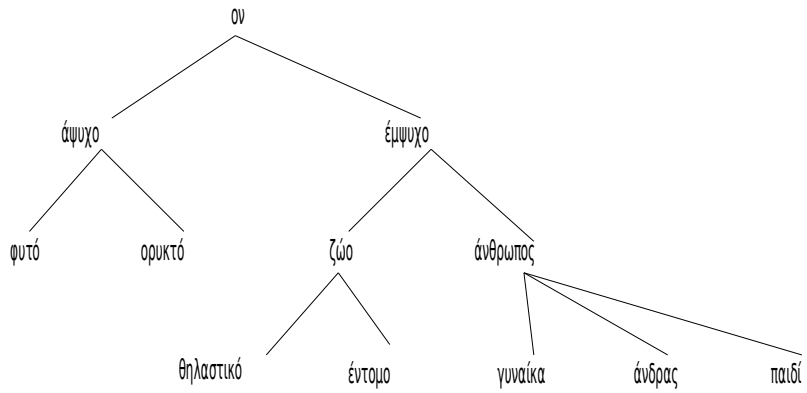
Δομή της λογικής της εφαρμογής

```
average-->initialization-->Μηνύματα στο χρήστη μέσω write
                           -->Ανάγνωση του πλήθους των αριθμών μέσω read
                           -->Δημιουργία λίστας των αριθμών μέσω input
          -->calculation   -->άθροιση των αριθμών με τη χρήση sum
                           -->και εύρεση μ.ο. με Ave is Sum /N
          -->output Παρουσίαση του μέσου όρου με μήνυμα μέσω write
```

Ερώτηση: Εκτελέστε average. βάλτε τρεις αριθμούς της αρεσκείας σας και δείτε το μέσο όρο τους.
Προσοχή η αριθμοί θα μπαίνουν εντός αγκυλών, έτσι [6].

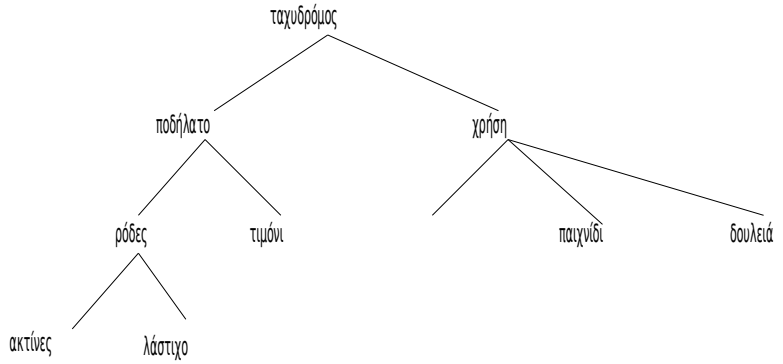
Ε. ΔΕΝΔΡΑ - ΤΕΛΕΣΤΕΣ

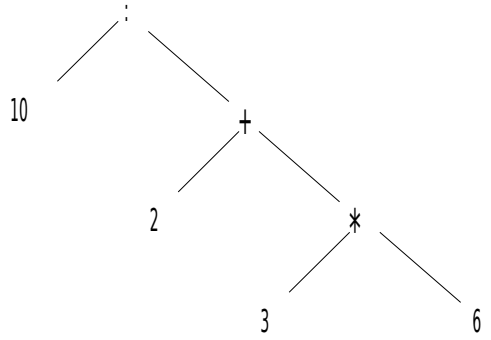
Εργασία 22η: Δένδρα



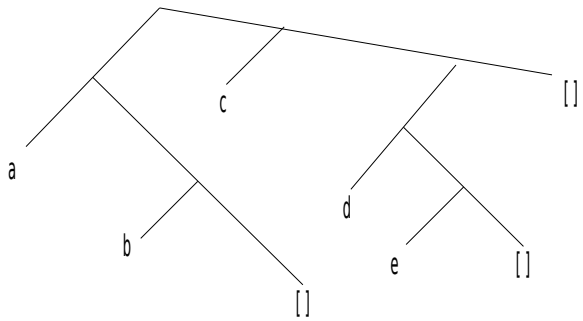
on(apsixo(fyto,orykto),
empsixo(zwo(8ilastiko,entomo),
an8rwpos(ginaika,andras,paidi))).

Ερώτημα
Μπορείτε να εκφράσετε μια αντίστοιχη σχέση με πριν

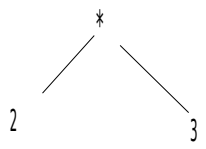


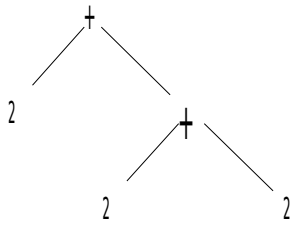


10/(2+3*6)

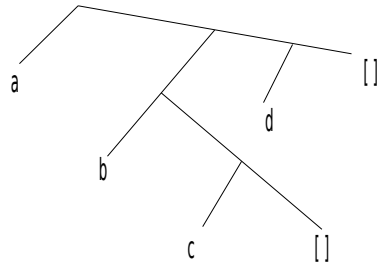


[[a,b],c,[d,e]]





same_result(*(2,3),+(2,+(2,2)).



.(a,.(.(b,.(c,[])),.(d,[]))
[a,[b,c],d]

Εργασία 23η: Σε περιβάλλον prolog εκτελέσατε
1. f(X,g(Y,a)) = f(Y,g(Z,Z)).

Προκαλούνται προς τα δεξιά ενοποιήσεις

Γράψε σε NOTEPAD:
paizei(alex,flaouto,plekei(mama,kaltses)).

Ερώτηση
paizei(X,flaouto,plekei(mama,Pote)).

Γράψε σε Notepad
spiti(alex,iolis,12).
spiti(niko,menelaou,23).
near(spiti(alex,iolis,12),spiti(niko,menelaou,23)).

Ρωτήστε σε περιβάλλον Prolog: near(spiti(X,iolis,12),spiti(Y,menelaou,23)).

Τώρα στο παραπάνω κώδικα στο notepad συμπληρώστε:
near(spiti(alex,iolis,12),spiti(jack,gewrgiou,3)).

Και εκτελέστε σε σε περιβάλλον Prolog:
near(spiti(X,iolis,12),spiti(J,gewrgiou,3)).

----
country(denmark,area(16633),population(5097000),capital(copenhagen), europe).
country(singapore,area(224),population(2584000),capital(singapore),asia).
country(greece,area(15000),population(10000000),capital(athens),europe).
country(france,area(50000),population(22584000),capital(paris),europe).

ec(Xwra,Hpeiros):-country(Xwra,_,_,_,Hpeiros).
popul_panw_apo(Xwra,X):-country(Xwra,_,population(Popul),_,_),Popul>X.

Ρωτήστε σε περιβάλλον Prolog:
ec(C,europe).
popul_panw_apo (C,6700000).

Εργασία 24η:Εκτελέστε σε περιβάλλον Prolog τα παρκάτω:
functor(f(a,b),F,N).
functor([a,b,c],F,N).
functor(father(X,Y),father,3).
functor(father(X,Y),father,2).
functor(X,father,3).

Ορίζει το όνομα και τις παραμέτρους μια σχέσης.
---
arg(2,go(treno,athens),A).
arg(3,f(g(a),h(u,o),k(j,s)),A).
arg(2,[a,f,g],A).
arg(1,a+f,C).
arg(1,f(s,g),g).
arg(1,f(s,g),s).

Ενοποιεί την ν-ιοστή παράμετρο με τη μεταβλητή Α.
---
name(filakia,S).
name(G,[102,105,108,97,107,105,97]).
name(H,”KeRaSi”).

Μας επιτρέπει να περάσουμε από ένα άτομο στη λίστα.
---
f(a,b,c)=.. L.
A=.. [father,alex,G].
[a,b,c,d]=.. L.
(a+b)=.. L.

Μπορούμε από γεγονός να δημιουργήσουμε λίστα, και από λίστα να δημιουργήσουμε γεγονός.

Γράψε στο notepad

payrate(secretary,65).
payrate(programmer,100).
payrate(boss,200).

Και σώσε το με το όνομα payrate.pl

Εκτελέστε τα ερωτήματα:
payrate(programmer,F).
payrate(analyti,J)=.. [payrate,G,234].
payrate(analyti,J)=.. [H,G,24].
G=.. [payrate,analyti,H].

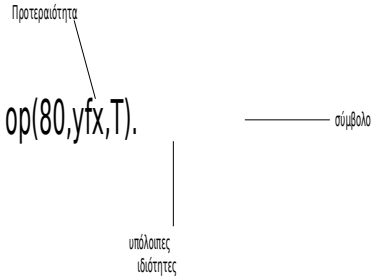
Εκτελέστε σε περιβάλλον Prolog
listing.
arg(2,[fred,jane,dick,peter],X).
arg(3,fruit(apple,orange,pear),Y).
functor(house(semi,1953,35000),X,Y).

Γράψτε σε notepad
friends(peter,jane).
friends(tom,dick,harry).

Τώρα εκτελέστε σε περιβάλλον Prolog
abolish(friends,2).

Εργασία 25η: Τελεστές

Ένας τελεστής χαρακτηρίζεται από:
1. Προτεραιότητα
2. Κατεύθυνση προτεραιότητας (αριστερά,δεξιά)
3. Αριθμός όρων (παραμέτρων).
4. Θέση του τελεστή σε σχέση με τους όρους.



	Αριθμός Αντικειμένων	Θέση Τελεστή	Κατεύθυνση προσεταιριστικότητας
xfy	2	Ενδιάμεση	Δεξιά
yfx	2	Ενδιάμεση	Αριστερά
xfx	2	Ενδιάμεση	Όχι
yfy	2	Ενδιάμεση	Αριστερά και Δεξιά
fx	1	Πριν	Όχι
fy	1	Πριν	Δεξιά
xf	1	Μετά	Όχι
yf	1	Μετά	Αριστερά

Γράψτε το πρόγραμμα

```
:- op(800,xfy,iz).
:- op(700,fx,the).
:- op(200,yfx,[father,brother,cousin,sister,mother,aunt,uncle]).
:- op(100,yfx,of).

john iz the father of peter.
thanasis iz the brother of xristos.
kostas iz the cousin of eleni.
lina iz the sister of efi.
eleni iz the mother of eleni.
nick iz the aunt of mitsos.
takis iz the uncle of theodoros.
```

Ρωτήστε :
Who iz the father of Someone.
Who iz the father of eleni.
Who iz the father of peter.
nick iz the X of Someone.
Someone iz the Something of eleni.

ΣΤ. ΑΣΚΗΣΕΙΣ ΣΤΑ ΠΡΟΗΓΟΥΜΕΝΑ

ΣΤ1. Ασκήσεις στους κανόνες

1η άσκηση

Ένας υπάλληλος προσπαθεί να δημιουργήσει μια λίστα με προσόντα για να προσληφθούν κάποιοι άλλοι στην εταιρία που δουλεύει. Τα κριτήρια που πρέπει να πληρούν κάποιοι ώστε να προσληφθούν είναι η γνώση Η/Υ, η γνώση οδήγησης και να ζούνε στο Λονδίνο.

Οι υποψήφιοι υπάλληλοι έχουν τα εξής χαρακτηριστικά:
Ο John Smith ζει στο Κέιμπριτζ, μπορεί να οδηγεί και δε ξέρει Η/Υ.
Ο Charles Brown ζει στο Λονδίνο, μπορεί να οδηγεί και ξέρει Η/Υ.
Ο Mary Jones ζει στο Λούτον, δε μπορεί να οδηγεί και ξέρει Η/Υ.
Ο Tony Evans ζει στο Λονδίνο, μπορεί να οδηγεί και ξέρει Η/Υ.
Ο Alice Greene ζει στο Λονδίνο, μπορεί να οδηγεί και ξέρει Η/Υ.

Ζητούμενα
Εκφράστε με γεγονότα τα παραπάνω.
Ορίστε τον κανόνα ώστε κάποιος υποψήφιος μπορεί να γίνει αποδεκτός. Ρωτήστε τελικά σε περιβάλλον Prolog και βρείτε ποιοι υποψήφιοι πληρούν τα κριτήρια για να προσληφθούν.

2η άσκηση

Δημιουργείστε ένα πρόγραμμα σε Prolog που να βρίσκει τους πιθανούς ενόχους σε κάποιο έγκλημα.

Τα βασικά γεγονότα είναι:
α. τα αισθήματα ενός ατόμου προς ένα άλλο: *αγαπά(Ατομο1, Ατομο2).*
Η Μαρία αγαπά τον Γιάννη
Η Μαρία αγαπά τον Πέτρο
Ο Πέτρος αγαπά τη Μαρία
Ο Πέτρος αγαπά τη Κλειώ
Η Κλειώ αγαπά τον Άλκη.

β. το επάγγελμα των ατόμων: *επάγγελμα(Ονομασία, Ατομο).*
Η Μαρία είναι μαγειρας
Ο Γιάννης είναι κηπουρός
Ο Γιάννης είναι κυνηγός
Ο Πέτρος είναι υδραυλικός
Ο Άλκης είναι κυνηγός
Η Κλειώ είναι φαρμακοποιός
Η Άννα είναι τραπεζικός.

Ισχύουν επίσης οι παρακάτω κανόνες που δημιουργούν νέα γεγονότα:
1. Κάποιος είναι φτωχός όταν είναι μάγειρας ή κηπουρός.
φτωχός(Όνομα)

2. Κάποιος είναι πλούσιος όταν είναι φαρμακοποιός ή τραπεζικός.
πλούσιος(Όνομα)

3. Κάποιος ανάλογα το επάγγελμα του έχει κάποιο πιθανό φονικό όπλο.
Ο μάγειρας έχει μαχαίρι.
Ο κυνηγός έχει όπλο.
Ο υδραυλικός έχει σωλήνα.
Ο φαρμακοποιός έχει δηλητήριο.
Όλοι μπορεί να έχουν περίστροφο.
έχει(Όπλο, Άτομο)

4.Για να είναι ύποπτος κάποιος πρέπει να ζηλεύει, ή να θέλει να κλέψει ή να έχει τάσεις αυτοκτονίας.
Ο ύποπτος ο οποίος ζηλεύει για να είναι ο δολοφόνος πρέπει να έχει όπλο, να αγαπά το θύμα και το θύμα να αγαπά κάποιον άλλο.

υποπτος(ζήλεια, Φονιάς, Θύμα, Όπλο)
Ο ύποπτος ο οποίος είναι κλέφτης για να είναι ο δολοφόνος πρέπει να έχει όπλο, να είναι φτωχός και το θύμα να είναι πλούσιος.
ύποπτος(κλοπή, Φονιάς, Θύμα, Όπλο)
Ο ύποπτος ο οποίος αυτοκτονεί για να είναι αυτόχειρας πρέπει να έχει όπλο, και να αγαπά αδιάφορο ποιον.
ύποπτος(αυτοκτονία, Αυτόχειρας, Αυτόχειρας, Όπλο)

Εκτελέσατε τα ερωτήματα,
1. Ποιος είναι δολοφόνος της Κλειούς που έχει μαχαίρι και τι κίνητρο είχε
2. Ποιες είναι οι πιθανές αυτοκτονίες με περίστροφο.
3. Βρείτε το άτομο που αγαπά τον Πέτρο και είναι φαρμακοποιός.
4. Ποιοι πιθανά θα δολοφονηθούν από κυνηγό
5. Ποιοι πλούσιοι κινδυνεύουν να σκοτωθούν από περίστροφο.
6. Ποιος είναι ο πλούσιος που μπορεί να κάνει φόνο με περίστροφο, για λόγους ζήλιας, σκοτώνοντας εκείνον τον υποψήφιο για αυτοκτονία με δηλητήριο που αγαπά το Γιάννη
7. Αγαπά η Μαρία κάποιον πιθανό δολοφόνο της.
8. Ποιο το πιθανό θύμα του Πέτρου.

ΣΤ2. Άσκησεις στις λίστες
1η άσκηση
Βρείτε πως μπορούμε να υπολογίσουμε το άθροισμα μιας λίστας με τερματική επιστροφή

2η άσκηση
Βρείτε πως μπορούμε να υπολογίσουμε το μέσο όρο μιας λίστας

3η άσκηση
Με τη χρήση της ένωσης πινάκων και των suffix, prefix δείξτε πως μια λίστα μπορεί να εμφανίζει ή να εξαφανίζει ένα - ένα τα στοιχεία της.

ΣΤ3. Άσκησεις στα δένδρα
1η άσκηση
Α. Δημιουργείστε πρόγραμμα σε Prolog που θα έχει ως γενικό γεγονός:
το βιβλίο και σε αυτό θα περιλαμβάνονται τέσσερα άλλα γεγονότα (μορφή δένδρου):
ο συγγραφέας με όνομα και επίθετο,
ο τίτλος,
ο εκδότης και
το έτος.
Β. Σε κανόνες να εκφράσετε ότι:
κάποιος έγραψε το τάδε βιβλίο
κάποιος εξέδωσε το τάδε βιβλίο
Γ. Να γραφτούν τρία ερωτήματα τα οποία αποδεικνύουν τη λειτουργία του προγράμματος.

2η άσκηση
Δημιουργείστε εφαρμογή για τη συγγένεια μεταξύ των μελών οικογένειας με τη χρήση δένδρων.

Τα γεγονότα θα προέλθουν από τα παρακάτω:
Πέτρος και Ελένη Θεοδώρου γέννησαν Κώστα, Βαγγέλη και Άννα Θεοδώρου
Γιάννης και Σοφία Θεοδώρου γέννησαν Πέτρο και Παναγιώτη Θεοδώρου
Κώστας και Καρολίνα Ιωαννίδου γέννησαν Ελένη Θεοδώρου και Νικολέτα Ιωαννίδου
Παναγιώτης Θεοδώρου και Νικολέτα Ιωαννίδου γέννησαν Σωτήρη Θεοδώρου

Δύο αδέρφια παντρεύτηκαν δύο αδελφές η οποία μία εξ αυτών διατηρεί το πατρικό της όνομα.

Ως γενικά γεγονότα θα ορίσετε:
το φύλλο του άνδρα με εσωτερικό γεγονός το ονοματεπώνυμο.
το φύλλο της γυναίκας με εσωτερικό γεγονός το ονοματεπώνυμο
το γεγονός μητέρα με εσωτερικά τα ονοματεπώνυμα παιδιού και μητέρας.

το γεγονός πατέρας με εσωτερικά τα ονοματεπώνυμα παιδιού και πατέρα.

Το γεγονός ονοματεπώνυμο θα έχει όρους το όνομα και επίθετο

Με κανόνες ορίστε τις σχέσεις:
γονέας, παιδί, θεία, θείος, γιος, κόρη, αγόρια αδέλφια, αδελφές, παππούς, γιαγιά,

ΣΤ4. Άσκηση στους τελεστές
1η άσκηση
Γράψτε ένα πρόγραμμα στο οποίο θα ορίσετε τελεστές που θα εξυπηρετούν την πρόταση
Κάποιος πίνει (πολύ ή λίγο ή καθόλου) Υγρό.

Συμπληρώστε με δομή όπως την παραπάνω κάποια δικά σας γεγονότα

Ορίστε κανόνες όπως
Κάποιος είναι αλκοολικός όταν Κάποιος πίνει πολύ ουίσκι

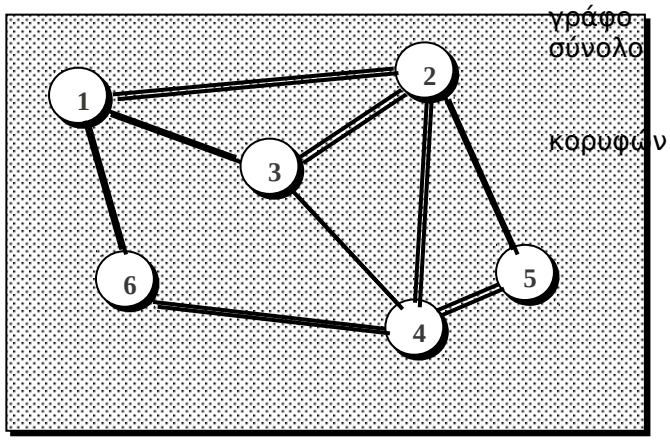
Τέλος εκτελέστε τα ερωτήματα
Ποιος πίνει πολύ Κάτι.
(ο τάδε) είναι Κάτι. (βάλτε στο ο τάδε συγκεκριμένο όνομα που χρησιμοποιήσατε)
Ο Τάδε πίνει Ρόφημα.

Ζ. ΕΦΑΡΜΟΓΕΣ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

Ζ1. Γράφοι

Ζ1.1. Παράσταση συνόλων και ορισμός σχέσεων μεταξύ τους μέσω λιστών

Υποθέτουμε τον εικονιζόμενο που παρίσταται από το παρακάτω γεγονότων, καθένα από τα οποία αντιπροσωπεύει ένα ζεύγος <u>συνδεόμενων</u> μεταξύ τους (ο *1ος* κόμβος κάθε ζεύγους συνδέεται με τον *2ο*):



sindesi(1,2).
sindesi(1,3).
sindesi(1,6).
sindesi(2,3).
sindesi(2,4).
sindesi(2,5).
sindesi(4,5).
sindesi(3,4).
sindesi(6,4).

Αντίστοιχα (με τη χρήση λιστών), ο εικονιζόμενος γράφος μπορεί να παρασταθεί αν σε κάθε κόμβο του αντιστοιχίσουμε μία λίστα, τα περιεχόμενα της οποίας αντιπροσωπεύουν τους <u>διπλανούς</u> του κόμβους με τους οποίους συνδέεται. Έτσι ο συγκεκριμένος γράφος μπορεί ισοδύναμα να εκφρασθεί σε περιβάλλον prolog ως εξής:

diplanoi_komvoi(1,[2,3,6]).
diplanoi_komvoi(2,[1,3,4,5]).
diplanoi_komvoi(3,[1,2,4]).
diplanoi_komvoi(4,[3,2,5,6]).
diplanoi_komvoi(5,[2,4]).
diplanoi_komvoi(6,[1,4]).

- Ένα από τα προβλήματα που συνήθως τίθενται σε εφαρμογές γράφων είναι η ***<u>εύρεση της διαδρομής (μονοπατιού) που συνδέει δύο κόμβους του γράφου</u>***.

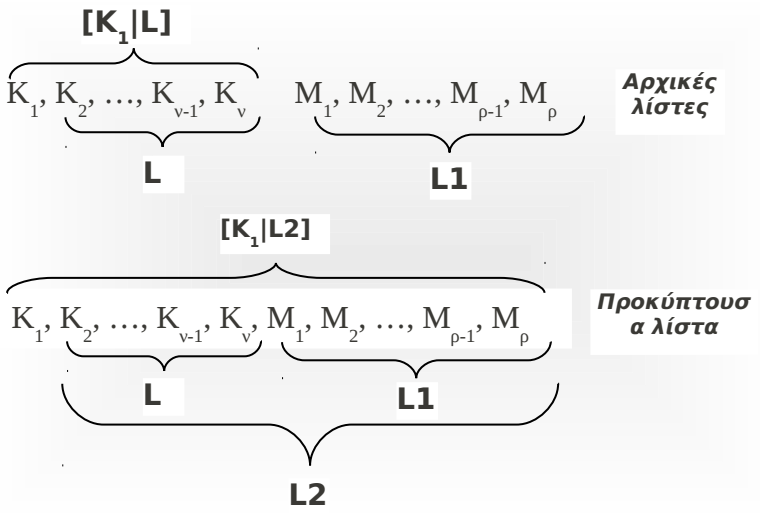
  Για τη λύση ενός τέτοιου προβλήματος θα μεταχειρισθούμε τις παρακάτω βασικές διαδικασίες διαχείρισης λιστών:

<u>Έλεγχος ύπαρξης στοιχείου σε λίστα</u>

Δοθείσης μιας λίστας *L1*, επιθυμούμε να γνωρίζουμε αν το στοιχείο *K* ανήκει στα στοιχεία της. Για να ανήκει το εν λόγω στοιχείο στη λίστα θα πρέπει να είναι ή το πρώτο στοιχείο (κεφαλή) της λίστας ή να είναι μέλος της ουράς (σώματος) της λίστας. Οι δύο αυτές διαπιστώσεις μπορούν να εκφρασθούν στην prolog ως εξής:

melos(K,[K|L1]). — *Το στοιχείο K είναι 1ο στοιχείο της λίστας L1.*
melos(K,[K1|L1]):-
melos(K,L1). — *Το στοιχείο K ανήκει στην ουρά της λίστας L1.*

Παράθεση δύο λιστών



Δοθέντων δύο λιστών *L1* και *L2* επιθυμούμε να προσαρτήσουμε τη λίστα *L2* στο τέλος της *L1* ώστε να αποκτήσουμε τη συγκεντρωτική λίστα *L1L2*. Για να υλοποιήσουμε αυτή τη διαδικασία αρκεί να παρατηρήσουμε ότι η προκύπτουσα λίστα πρέπει να έχει σαν *1ο* στοιχείο (*K1*), το *1ο* στοιχείο της *L1*, η δε ουρά της (*L2*) θα προκύπτει από την παράθεση της ουράς της 1ης λίστας (L) με την 2η λίστα (*L1*). Το σκεπτικό αυτό εκφράζεται στην prolog με τον παρακάτω κανόνα:

```
parathesi([K1|L],L1,[K1|L2]):-
        parathesi(L,L1,L2).
```

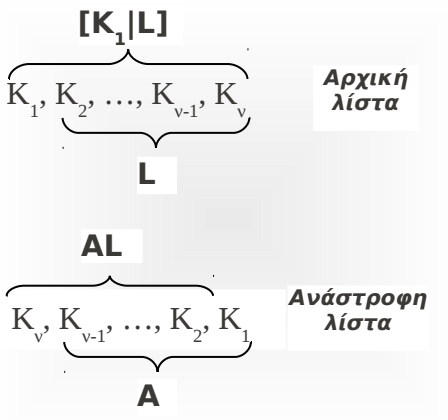
Πρέπει επίσης να εξετάσουμε το (οριακό) ενδεχόμενο να είναι κενή η 1η λίστα. Σε μια τέτοια περίπτωση η παράθεση μιας κενής λίστας με οποιαδήποτε άλλη λίστα *L* δίνει σαν προκύπτουσα λίστα την ίδια την *L*, διαπίστωση που εκφράζεται από το γεγονός:

```
parathesi([],L,L).
```

Συμπερασματικά ο ορισμός της παράθεσης δύο λιστών εκφράζεται από:

```
parathesi([],L,L).
parathesi([K1|L],L1,[K1|L2]):-
        parathesi(L,L1,L2).
```

Αναστροφή λίστας



Από το εικονιζόμενο σχήμα προκύπτει ότι η ανάστροφη μιας λίστας είναι μία άλλη λίστα *A* που προκύπτει από την παράθεση της ανάστροφης της ουράς της αρχικής με τη λίστα που έχει σαν στοιχείο την κεφαλή της αρχικής, δηλαδή:

```
anastrofi([K1|L],A):-
        anastrofi(L,AL),
        parathesi(AL,[K1],A).
```

Συμπληρώνοντας τον ορισμό αυτό με την οριακή περίπτωση της κενής λίστας παίρνουμε τελικά:

```
anastrofi([],[]).
anastrofi([K1|L],A):-
        anastrofi(L,AL),
        parathesi(AL,[K1],A).
```

Έκφραση διαδρομής σύνδεσης μεταξύ δύο κόμβων γράφου

Θεωρώντας σαν αφετηρία έναν αρχικό κόμβο (*A*) του γράφου και σαν προορισμό έναν τελικό κόμβο (*T*), η διαδρομή που τους συνδέει προκύπτει από την αναστροφή του αντίστοιχου μονοπατιού *κόμβος A → …→ κόμβος T* (του μονοπατιού *Amonopati*) που χαρακτηρίζεται από μία λίστα *A* επισκεφθέντων κόμβων. Η θεώρηση αυτή μπορεί να εκφρασθεί στην prolog από τον παρακάτω κανόνα:

```
diadromi(A,T,Diadromi):-
        monopati(A,T,[A],Amonopati),
          anastrofi(Amonopati,Diadromi).
```

Αναδρομικός ορισμός μονοπατιού σύνδεσης δύο κόμβων

Δοθέντων των κόμβων *A* και *T*, η διαδρομή (*Diadromi*) που τους συνδέει είναι το μονοπάτι (*monopati*) που σε κάθε βήμα της έρευνας αρχίζει από έναν διπλανό κόμβο (*A1*) του *A* και καταλήγει στον *T*, αρκεί ο επιλεγείς κόμβος *A1* να μην περιέχεται στη λίστα *Tmima_Monopatioy* των ήδη επισκεφθέντων μέχρι εκείνη τη στιγμή κόμβων. Ο αναδρομικός αυτός ορισμός του αναζητούμενου μονοπατιού μπορεί να εκφρασθεί ως εξής:

```
monopati(T,T,Diadromi,Diadromi).
monopati(A,T,Tmima_Monopatioy,Diadromi) :-
        diplana(A,A1),
        not(melos(A1,Tmima_Monopatioy)),
        monopati(A1,T,[A1|Tmima_Monopatioy],Diadromi).
```

Με βάση τα ανωτέρω, ο πλήρης κώδικας prolog για την αναζήτηση μιας διαδρομής στο εσωτερικό ενός γράφου είναι ο ακόλουθος:

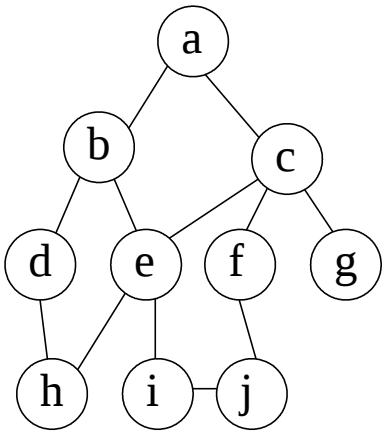
```
sindesi(1,2).
sindesi(1,3).
sindesi(1,6).
sindesi(2,3).
sindesi(2,4).
sindesi(2,5).
sindesi(4,5).
sindesi(3,4).
sindesi(6,4).

diplana(A,A1):-
        sindesi(A,A1).
diplana(A,A1):-
        sindesi(A1,A).

melos(K,[K|L1]).
melos(K,[K1|L1]):-
melos(K,L1).

parathesi([],L,L).
parathesi([K1|L],L1,[K1|L2]):-
        parathesi(L,L1,L2).

anastrofi([],[]).
anastrofi([K1|L],A):-
        anastrofi(L,AL),
        parathesi(AL,[K1],A).

diadromi(A,T,Diadromi):-
        monopati(A,T,[A],Amonopati),
          anastrofi(Amonopati,Diadromi).
```

```
monopati(T,T,Diadromi,Diadromi).
monopati(A,T,Tmima_Monopatioy,Diadromi) :-
        diplana(A,A1),
        not(melos(A1,Tmima_Monopatioy)),
        monopati(A1,T,[A1|Tmima_Monopatioy],Diadromi).
```

Μετά την ενεργοποίηση του προγράμματος, η υποβολή της ερώτησης *diadromi(1,4,M).* θα έχει σαν αποτέλεσμα τον εντοπισμό των παρακάτω διαδρομών:

M = [1, 2, 3, 4]
M = [1, 2, 4]
M = [1, 2, 5, 4]
M = [1, 3, 4]
M = [1, 3, 2, 4]
M = [1, 3, 2, 5, 4]
M = [1, 6, 4]

Ζ1.2. Αναζήτηση πρώτα κατά βάθος (με Depth First αλγόριθμο)



Υλοποίηση της σύνδεσης των κόμβων κατά το παραπάνω σχήμα

```
connected(a,b).
connected(b,d).
connected(b,e).
connected(d,h).
connected(e,i).
connected(e,h).
connected(a,c).
connected(c,f).
connected(c,e).
connected(c,g).
connected(f,j).
connected(i,j).
```

Υλοποίηση κανόνα επίλυσης αναζήτησης

```
solve(Start,End,Solution):-
write(Start),
df(Start,End,[],Solution).
```

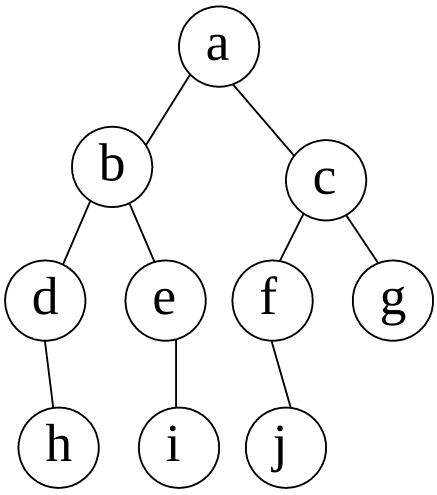
Ανάπτυξη κανόνα df

```
df(End,End,Path,[End|Path]):-!.
df(Node,End,Path,Solution):-
connected(Node,Node1),
not(member(Node1,Path)),
write(Node1),
df(Node1,End,[Node|Path],Solution),!.
```

Ερωτήματα: Εύρεση διαδρομής κατά depth first (χρήση γεγονότος solve)

1. από το a στο j
2. από το c στο j
3. από το e στο j
4. από το a στο g
5. από το c στο h

Z1.3. Αναζήτηση πρώτα κατά πλάτος (με Breadth First αλγόριθμο)




```
connected(a,b).
connected(b,d).
connected(b,e).
connected(d,h).
connected(e,i).
connected(a,c).
connected(c,f).
connected(c,g).
connected(f,j).
```

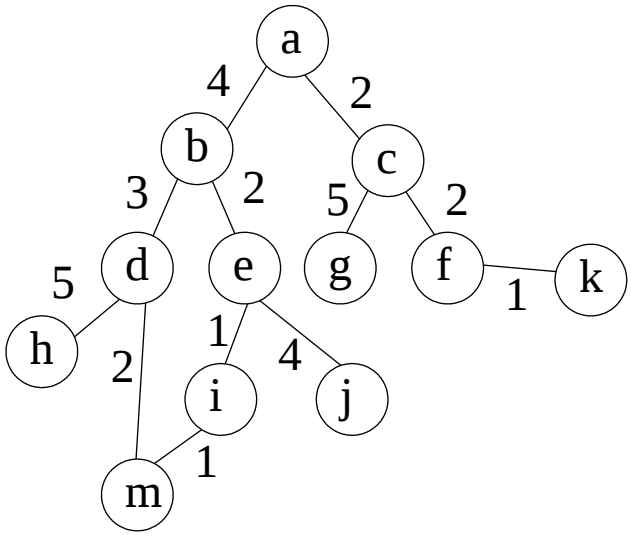
```
solve_bf(Arxikos,Telikos,Epilysi):-
```

```
write(Arxikos),
bf([[Arxikos]],Telikos,Epilysi).

bf([[Telikos|Monopati]|_],Telikos,[Telikos|Monopati]):-!.

bf([Monopati|Monopatia],Telikos,Epilysi):-
candidate_monopatia(Monopati,Neo_monopati),
display_monopati(Neo_monopati),
append(Monopatia,Neo_monopati,Monopatia1),
bf(Monopatia1,Telikos,Epilysi).

candidate_monopatia([Komvos|Monopati],Neo_monopati):-
bagof([X,Komvos|Monopati],
(connected(Komvos,X), \+ member(X,[Komvos|Monopati])),Neo_monopati),!.

candidate_monopatia(Monopati,[]).

display_monopati([]).
display_monopati([[H|T]|Rest]):-
write(H),
display_monopati(Rest).
```

Ερωτήματα: Εύρεση διαδρομής κατά bredth first (χρήση γεγονότος solve_bf)

1. από το a στο j
2. από το c στο j
3. από το a στο h

Z1.4. Αναζήτηση με αλγόριθμο εύρεσης ελαχίστου κατά μικρότερο κόστος

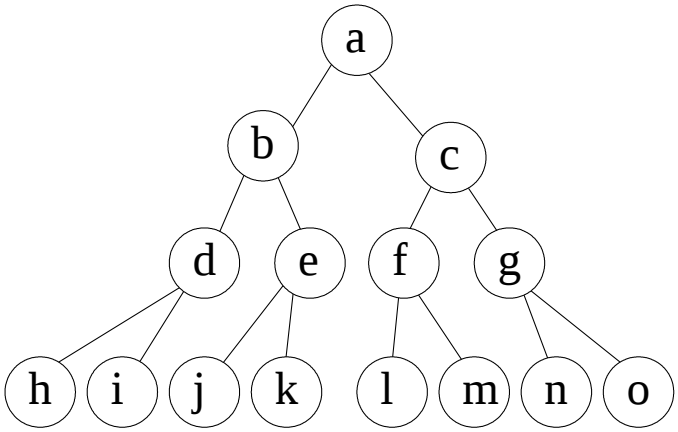



```
con(a,b,4).
con(a,c,2).
con(b,d,3).
con(b,e,2).
con(c,g,5).
con(c,f,2).
con(d,h,5).
con(d,m,2).
con(e,i,1).
con(e,j,4).
con(f,k,1).
con(i,m,1).
```

```
findmin(A,B,Min):-
findall(Path-Cost,path(A,B,Path,Cost),Results),
find(Results,Min).

path(A,B,Path,Cost):-
path2(A,[B],0,Path,Cost).
path2(A,[A|Path2],Cost2,[A|Path2],Cost2).
path2(A,[Y|Path2],Cost2,Path,Cost):-
con(X,Y,CostXY),
\+member(X,Path2),
Cost3 is Cost2 + CostXY,
path2(A,[X,Y|Path2],Cost3,Path,Cost).


find([L-X],[L-X]).
find([L2-X,L3-Y|T],Min):-
find([L3-Y|T],Min2),
min([L2-X],Min2,Min).
min([L2-X],[_-Y],[L2-X]):-
X=<Y,!.
min([_-_],[L3-Y],[L3-Y]).
```

Ερωτήματα:
Εύρεση διαδρομής (χρήση γεγονότος findmin)από το a στο m

Z1.5. Αναζήτηση με αλγόριθμο max-min-max




```
test(Start,Bestsucc):-
minimax(Start,Bestsucc,Val),
write('Best move from position '),
write(Start),
write(' is to move to position '),
write(Bestsucc),
nl,
write('Value is: '),
write(Val).

tree(a,[b,c]).
tree(b,[d,e]). tree(d,[h,i]).
tree(e,[j,k]). tree(c,[f,g]). tree(f,[l,m]). tree(g,[n,o]).
tree(h,[]). tree(i,[]). tree(j,[]). tree(k,[]).
tree(l,[]). tree(m,[]). tree(n,[]). tree(o,[]).
```

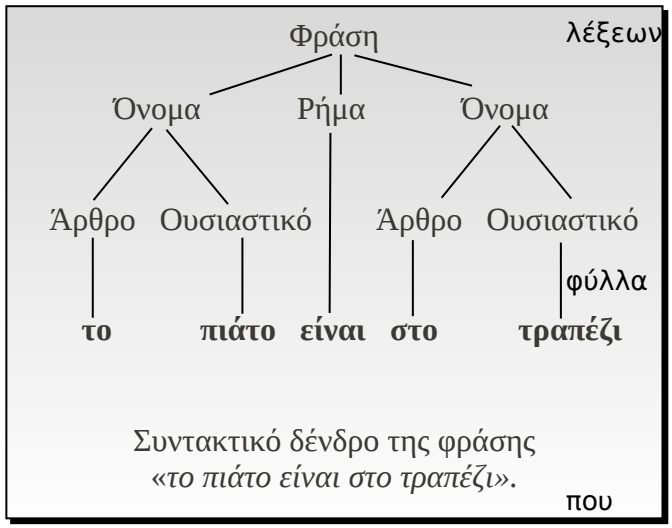
```
staticval(h,1). staticval(i,4). staticval(j,5). staticval(k,6).
staticval(l,2). staticval(m,1). staticval(n,1). staticval(o,1).

min_to_move(X):-
member(X,[b,c,h,i,j,k,l,m,n,o]).

max_to_move(X):-
member(X,[a,d,e,f,g]).

moves(Pos,Moves):-
tree(Pos,Moves).

minimax(Pos,Bestsucc,Val):-
moves(Pos,Poslist),!,
(
best(Poslist,Bestsucc,Val)
;
staticval(Pos,Val)
).

best([Pos],Pos,Val):-
minimax(Pos,_,Val),!.

best([Pos1|Poslist],Bestpos,Bestval):-
minimax(Pos1,_,Val1),
best(Poslist,Pos2,Val2),
betterof(Pos1,Val1,Pos2,Val2,Bestpos,Bestval).

betterof(Pos0,Val0,_Pos1,Val1,Pos0,Val0):-
min_to_move(Pos0),
Val0 > Val1,
!
;
max_to_move(Pos0),
Val0 < Val1,!.

betterof(_Pos0,_Val0,Pos1,Val1,Pos1,Val1).
```

Ερωτήματα: Εύρεση διαδρομής με επιλογή μεγαλύτερης ή μικρότερης τιμής (χρήση γεγονότος test)

- test(a,Max).
- test(b,Min).
- test(c,Min).
- test(d,Max).
- test(e,Max).
- test(f,Max).
- test(g,Max).

Z2. Αυτόματη κατανόηση γραπτού λόγου
1. ΕΙΣΑΓΩΓΗ

Δύο από τις σπουδαιότερες φάσεις κατά την αυτόματη μεταχείριση μιας γραπτής πρότασης είναι ***συντακτική*** και η ***σημασιολογική*** της ανάλυση.
Η πρώτη απ' αυτές στοχεύει στον εντοπισμό της *επιφανειακής της δομής* χρησιμοποιώντας συγκεκριμένους γραμματικούς (συντακτικούς) κανόνες που της έχουν γνωστοποιηθεί και με τη βοήθεια των οποίων επιδιώκει την παραγωγή της ακολουθίας των λέξεων της φράσης εισόδου μέσω ενός συντακτικού δένδρου που κατασκευάζει. Στα πλαίσια της συντακτικής ανάλυσης, το γεγονός ότι η αντίστοιχη εφαρμογή βρίσκει κάποιους κανόνες που της επιτρέπουν να κατασκευάσει ένα δένδρο, τα (τερματικοί κόμβοι) του οποίου συνθέτουν τη φράση εισόδου, αυτό σημαίνει ότι η εν λόγω φράση αναγνωρίστηκε και συνεπώς θεωρείται συντακτικά σωστή.



Συντακτικό δένδρο της φράσης
«το πιάτο είναι στο τραπέζι».

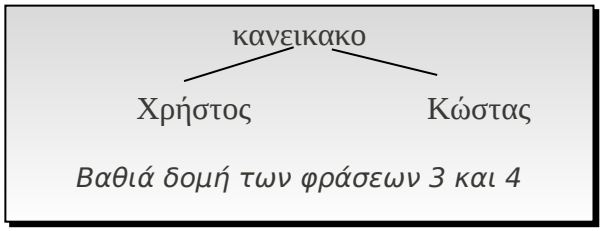
Το γενικό σχήμα των συντακτικών κανόνων είναι **A1 → A2 A3 | B** δηλώνει ότι από τον κόμβο του συντακτικού δένδρου *A1* προκύπτουν τα υποδένδρα *A2* και *A3* ή το υποδένδρο *B*. Ονομάζουμε ***γραμματική*** το σύνολο των χρησιμοποιούμενων συντακτικών κανόνων.
Για παράδειγμα η ενδεικτική γραμματική

Φράση → Όνομα Ρήμα Όνομα |
Όνομα Ρήμα το Όνομα |
Όνομα Ρήμα
Όνομα → Άρθρο Επίθετο Ουσιαστικό |
Άρθρο Ουσιαστικό
Ρήμα → χτύπησε | έκανε | χτυπήθηκε | κλαίει | πεινάει | έσπασε
Άρθρο → ο | η | το | τον | την | στο | τη
Επίθετο → καλό | όμορφο | γλυκιά | πικρό
Ουσιαστικό → Χρήστος | βάζο | φάρμακο | γλυκιά | Κώστας | Χρήστο | Κώστα

μπορεί να αναγνωρίσει φράσεις του τύπου

1. ***το όμορφο βάζο έσπασε***
2. ***το πικρό φάρμακο κλαίει το βάζο***
3. ***ο Χρήστος χτύπησε τον Κώστα***
4. ***ο Κώστας χτυπήθηκε από τον Χρήστο κλπ***

Επειδή ορισμένες φράσεις (αν και συντακτικά σωστές) δεν έχουν νόημα (π.χ. η $2^{η}$) καλείται στη συνέχεια η σημασιολογική ανάλυση να αναγνωρίσει όχι μόνο αν το προκύπτον συντακτικό δένδρο μιας φράσης έχει νόημα αλλά και να αποδώσει το ίδιο νόημα σε δύο φράσεις που ενδεχομένως χρησιμοποιούν διαφορετικά συντακτικά δένδρα για να το εκφράσουν (π.χ. η $3^{η}$ και η $4^{η}$). Το έργο αυτό θα γίνει με τη βοήθεια συντακτικο-σημασιολογικών πληροφοριών (ουσιαστικών και ρημάτων) που περιλαμβάνει το λεξιλόγιό μας. Με βάση αυτές τις πληροφορίες η σημασιολογική ανάλυση θα κατασκευάσει τη *βαθιά δομή* της φράσης, ένα δένδρο δηλαδή του οποίου οι κόμβοι εκφράζουν πλέον σημασιολογικές κατηγορίες. Το πόσο αξιόπιστο



Βαθιά δομή των φράσεων 3 και 4

και νοήμον θα είναι το πρόγραμμα ανάλυσης εξαρτάται από τον αριθμό, το είδος και το βαθμό λεπτομέρειας των εμφανιζόμενων στη βαθιά δομή σημασιολογικών κατηγοριών. Βεβαίως, μία ουσιαστική κατανόηση προϋποθέτει εκτός από την κατασκευή του αντίστοιχου σημασιολογικού δένδρου για μία φράση εισόδου και τη δυνατότητα εξαγωγής συμπερασμάτων που αφορούν μη εμφανιζόμενες στη φράση πληροφορίες (υπονοούμενες). Για παράδειγμα από το γεγονός «*ο Κώστας χτυπήθηκε από το Χρήστο*» συμπεραίνουμε ότι «*ο Κώστας μάλωσε με το Χρήστο*», ότι «*ο Κώστας πονάει*» κλπ. Η σημασιολογική ανάλυση λοιπόν πρέπει να ενεργοποιεί συγκεκριμένους (προκαθορισμένους από τον κατασκευαστή του προγράμματος και εξαρτώμενους από την εφαρμογή) συμπερασματικούς κανόνες που συνήθως υλοποιούνται με τα λεγόμενα ***πλαίσια*** (frames), δηλαδή από σύνολα σημασιολογικών πληροφοριών που εκφράζουν τυποποιημένα καθημερινά σενάρια. Με τον τρόπο αυτό επιδιώκει να συγκρίνει τις σημασιολογικές πληροφορίες που εντοπίζει στη φράση εισόδου με τα προκαθορισμένα πλαίσια που διαθέτει, εφαρμόζοντας (με τη βοήθεια της ενοποίησης της Prolog) μεθόδους σύγκρισης (***pattern matching***) ώστε να αποδώσει συγκεκριμένες τιμές στις μεταβλητές των πλαισίων. Για λόγους απλούστευσης της τρέχουσας εργαστηριακής εφαρμογής, θα αποφύγουμε την ενσωμάτωση πλαισίων στον κώδικα και θα αρκεσθούμε στην κατασκευή της βαθιάς δομής μιας φράσης εισόδου.

2. ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΣΕ ΠΕΡΙΒΑΛΛΟΝ PROLOG

Θα θεωρήσουμε ότι η κατασκευαζόμενη βαθιά δομή συντίθεται από τις παρακάτω πρωταρχικές σημασιολογικές πράξεις (<u>*semantic primitives*</u>):

- ***katastasi(X,Y)***: Η κατάσταση του *X* είναι *Y*.
- ***kanikako(X,Y)***: Ο *X* ενεργεί με αρνητικά αποτελέσματα για τον *Y*.
- ***katoxi(X,Y,A)***: Το αντικείμενο (ή η πληροφορία) *A* μεταβιβάζεται από τον *X* στον *Y*. Αν οι *X* και *Y* ταυτίζονται, αυτό σημαίνει ότι ο *X* κατέχει το αντικείμενο *A* (ή γνωρίζει την πληροφορία *A*).
- ***analosi(X,Y)***: Ο *X* τρώει το Y.

Παραδείγματα σημασιολογικών δομών φράσεων

το σκυλάκι πεινάει
analosi(skylaki,kaki)

η μαρία έδωσε ένα μήλο στο σκυλάκι
katoxi(maria,skylaki,milo)

η μαρία νομίζει ότι το σκυλάκι έφαγε το μήλο
katoxi(maria,maria,analosi(skylaki,milo))

Το κατηγόρημα ***frasi*** θα καλύψει ταυτόχρονα τη συντακτική ανάλυση και την κατασκευή της βαθιάς δομής της φράσης εισόδου. Η *1η* από τις δύο παραμέτρους του θα εκφράζει τη βαθιά δομή που προκύπτει από την ανάλυση της *2ης* παραμέτρου που αντιστοιχεί στη φράση εισόδου.

κατηγόρημα frasi(βαθιά δομή, φράση εισόδου)

δένδρο με κλαδιά τις μεταβλητές που θα ενοποιηθούν αντίστοιχα με το όνομα της πρωταρχικής σημασιολογικής πράξης και τα ονόματα των πρωταγωνιστών της

λίστα με στοιχεία τις λέξεις που περιέχει

Η υλοποίησή του θα γίνει από ισάριθμους κανόνες με τους τύπους των φράσεων που θα αναγνωρίζει (υποκείμενο και αντικείμενο, φράση χωρίς αντικείμενο, φράση με δευτερεύουσες φράσεις κλπ). Το δεξιό σκέλος κάθε κανόνα θα περιλαμβάνει τα κατηγορήματα <u>*onoma*</u> και <u>*praxi*</u> με ***τρεις*** παραμέτρους το καθένα. Η χρήση τους συνίσταται στο να αφαιρούν από τη λίστα (*2η παράμετρος*) τις λέξεις που αναγνωρίζουν εξάγοντας τη χρήσιμη πληροφορία (πρωταγωνιστής ή πρωταρχική πράξη) η

οποία ενοποιείται με την *1η παράμετρο*, αποδίδοντας στο επόμενο κατηγόρημα το κομμάτι της φράσης (*3η παράμετρος*) που απομένει να αναγνωρισθεί.
Η σύνταξη των κανόνων συντακτικής ανάλυσης ακολουθεί την προκαθορισμένη γραμματική. Κάθε φορά που μία σχέση του δεξιού σκέλους αποτυγχάνει να αναγνωρίσει το κομμάτι της φράσης που της αναλογεί, ο κανόνας αποτυγχάνει και η prolog επιχειρεί (με τον επόμενο κανόνα) να αποδείξει ότι η φράση εισόδου ακολουθεί μία άλλη (προβλεπόμενη από τη γραμματική) μορφή, ενώ σε περίπτωση επιτυχίας το *δένδρο εκτέλεσης* της prolog ταυτίζεται με το συντακτικό δένδρο της φράσης εισόδου. Η αποτυχία του κατηγορήματος *frasi* θα σημαίνει ότι η φράση εισόδου δεν έχει μία από τις προβλεπόμενες μορφές της γραμματικής ή ότι περιλαμβάνει άγνωστη (για το υπάρχον λεξιλόγιο) λέξη.
Το προκαθορισμένο λεξιλόγιο γνωστοποιείται στο πρόγραμμα υπό μορφή γεγονότων, ενώ σε κάθε λέξη έχουν προστεθεί οι απαραίτητες για την κατασκευή της βαθιάς δομής συντακτικο-σημασιολογικές πληροφορίες (π.χ. *f*, *m* και *n* για τον προσδιορισμό του γένους, *energitiko* και *pathitiko* για τον προσδιορισμό της φωνής του ρήματος κλπ).
Τα στοιχεία των δένδρων της βαθιάς δομής παρίστανται (για λόγους εύκολης διαχείρισης του ονόματος των πρωταρχικών πράξεων) σαν όροι ενός γενικού δένδρου με ρίζα το ***ph*** (π.χ. η πράξη *katoxi(X,X,A)* θα παρίσταται με *ph(katoxi,X,X,A)* στον κώδικα).
Με το συνδυασμό των εννοιών της *ενοποίησης* και της *μη προσδιοριστικότητας* η prolog είναι σε θέση να μας παρέχει εκτός από τη βαθιά δομή μιας φράσης εισόδου και *όλες τις φράσεις*

3. ΠΛΗΡΗΣ ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
epitheto(omorfi,f).
epitheto(oraio,n).
epitheto(kali,f).
arthro(o,m).
arthro(i,f).
arthro(ena,n).
arthro(mia,f).
arthro(to,n).
arthro(ton,m).
arthro(ti,f).
arthro(tin,f).
arthro(sti,f).
arthro(stin,f).
arthro(sto,n).
arthro(ston,m).
rima(espase,kanikako,energitiko).
rima(xtypise,kanikako,energitiko).
rima(xtypithike,kanikako,pathitiko).
rima(efage,analosi,energitiko).
rima(fagothike,analosi,pathitiko).
rima(edose,katoxi,energitiko).
rima(pire,katoxi,pathitiko).
rima(peinaei,katastasi,pathitiko).
rima(ponaei,katastasi,pathitiko).
rima(gelaei,katastasi,energitiko).
rima(klaiei,katastasi,pathitiko).
rima(eipe,katoxi,energitiko).
rima(nomizei,katoxi,energitiko).
rima(malose,kanikako,energitiko).
ousiastiko(maria,f,empsyxo).
ousiastiko(anna,f,empsyxo).
ousiastiko(ivan,m,empsyxo).
ousiastiko(skylaki,n,empsyxo).
ousiastiko(milo,n,kali).
ousiastiko(xastouki,n,kaki).
ousiastiko(aspirini,f,kali).
ousiastiko(vazo,n,apsyxo).
```

```
/* o ivan xtypise tin maria */
frasi(ph(A,Yp,Ap),L):-
onoma(Yp,L,L1),
praxi(A,energitiko,L1,L2),
onoma(Ap,L2,[]).

/* o ivan xtypithike apo tin maria */
frasi(ph(A,Yp,Ap),L):-
onoma(Ap,L,L1),
praxi(A,pathitiko,L1,[apo|L2]),
onoma(Yp,L2,[]).

/* o ivan ponaei */
frasi(ph(A,Ap,kaki),L):-
onoma(Ap,L,L1),
praxi(A,pathitiko,L1,[]).

/* o ivan gelaei */
frasi(ph(A,X,kali),L):-
onoma(X,L,L1),
praxi(A,energitiko,L1,[]).

/* o ivan edose ena milo stin maria  */
frasi(ph(A,Yp,Ap,Ant),L):-
onoma(Yp,L,L1),
praxi(A,energitiko,L1,L2),
onoma(Ant,L2,L3),
onoma(Ap,L3,[]).

/* o ivan pire ena milo apo tin maria  */
frasi(ph(A,Yp,Ap,Ant),L):-
onoma(Ap,L,L1),
praxi(A,pathitiko,L1,L2),
onoma(Ant,L2,[apo|L3]),
onoma(Yp,L3,[]).

/* o ivan pire mia aspirini */
frasi(ph(A,_,Ap,Ant),L):-
onoma(Ap,L,L1),
praxi(A,pathitiko,L1,L2),
onoma(Ant,L2,[]).

/* o ivan nomizei oti i maria gelaei */
frasi(ph(A,Yp,Ap,Ant),L):-
onoma(Yp,L,L1),
praxi(A,energitiko,L1,L2),
onoma(Ap,L2,[oti|L3]),
frasi(Ant,L3).

/* o ivan eipe stin maria oti to skylaki espase to vazo */
frasi(ph(A,Yp,Yp,Ant),L):-
onoma(Yp,L,L1),
praxi(A,energitiko,L1,[oti|L2]),
frasi(Ant,L2).

praxi(P,Phoni,[V|Y],Y):-
rima(V,P,Phoni).
```

```
onoma(X,[A,X|Y],Y):-
arthro(A,Genos),
ousiastiko(X,Genos,_).

onoma(X,[A,E,X|Y],Y):-
arthro(A,Genos),
epitheto(E,Genos),
ousiastiko(X,Genos,_).

onoma(X,[X|Y],Y):-
ousiastiko(X,_,_).
```

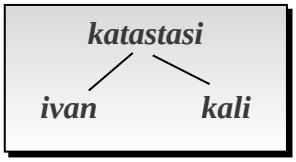
4. ΕΝΔΕΙΚΤΙΚΟΙ ΔΙΑΛΟΓΟΙ

```
?- frasi(Bathia_Domi,[o,ivan,edose,ena,milo,stin,maria]).
   Bathia_Domi = ph(katoxi, ivan, maria, milo)

?- frasi(Bathia_Domi,[i,maria,pire,ena,milo,apo,ton,ivan]).
   Bathia_Domi = ph(katoxi, ivan, maria, milo)

?- frasi(ph(katastasi,maria,kaki),Frasi).
   Frasi = [i, maria, peinaei] ;
   Frasi = [i, maria, ponaei] ;
   Frasi = [i, maria, klaiei] ;
   Frasi = [mia, maria, peinaei] ;
   Frasi = [mia, maria, ponaei] ;
   Frasi = [mia, maria, klaiei] ;
   Frasi = [ti, maria, peinaei] ;
   Frasi = [ti, maria, ponaei] ;
   Frasi = [ti, maria, klaiei] ;
   Frasi = [tin, maria, peinaei] ;
   Frasi = [tin, maria, ponaei] ;
   .....
```

Η μη ορθή σύνταξη κάποιων από τις φράσεις-αποτελέσματα (π.χ. ti maria peinaei) οφείλεται στην απουσία από το πρόγραμμα κατάλληλων συντακτικών πληροφοριών (άγνοια πτώσεων)

5. ΑΣΚΗΣΕΙΣ

1. Ζητείστε από την prolog να σας δώσει όσες φράσεις με την εικονιζόμενη σημασιολογία (βαθιά δομή) γνωρίζει το πρόγραμμα.



2. Ποιες φράσεις αντιστοιχούν στη βαθιά δομή *ph(kanikako,ivan,vazo)*;

3. Ποια σημασιολογία (από τις διαθέσιμες) διέπει τη φράση *to milo fagothike*;

Z3. Προσομοίωση Φυσικής Γλώσσας

Η παρούσα εφαρμογή προσομοιάζει τη φυσική γλώσσα και δίδει ένα παράδειγμα απάντησης του Η/Υ σε ερωτήσεις του χρήστη. Αρχικά διαβάζεται από την εφαρμογή μια πρόταση που εισάγει ο χρήστης. Το response μετατρέπει την πρόταση του χρήστη σε απάντηση από τον Η/Υ χρησιμοποιώντας δομή λίστας και το transform. Δηλαδή οι λέξεις της πρότασης του χρήστη γίνονται λίστα και από αυτές δημιουργείται η απάντηση του Η/Υ ως άλλη λίστα. Η όλη διαδικασία τελειώνει όταν τελειώσουν οι λέξεις της πρότασης του χρήστη δηλαδή, όταν δημιουργηθεί κενή λίστα. Έπειτα με το print_list από τη λίστα των λέξεων της απάντησης του Η/Υ, εμφανίζεται ως πρόταση αυτή.

Κώδικας του προγράμματος

```
english:-
nl,
write('User> '),
text(Inlist),
nl,
continue(Inlist).
```

Το αρχικό γεγονός είναι το english το οποίο δίδει το πλαίσιο επικοινωνίας. Αρχικά έχει το λόγο ο χρήστης και γράφει τη πρότασή του. Μέσω του text αυτή αποθηκεύεται σε μια λίστα λέξεων και έπειτα εκτελείται το continue.

```
continue([finish,.]):-!.
```

Το continue λήγει την εφαρμογή όταν πληκτρολογηθεί από τον χρήστη η λέξη finish

```
continue(Inlist):-
response(Inlist,Outlist),
write('Computer> '),
print_list(Outlist),
english.
```

Εάν ο χρήστης δε γράψει τη λέξη finish εκτελούνται με σειρά το response, print_list και ξανά το english.
- με το response δημιουργείται μια απάντηση από τον Η/Υ που σχετίζεται με το τι είπε ο χρήστης
- το print_list την δείχνει στην οθόνη σαν να απαντά ο Η/Υ
- έπειτα ξαναενεργοποιείται το english για νέα συζήτηση μεταξύ χρήστη και Η/Υ

```
text([Word|Words]):-
get0(Char),
word(Char,Word,Char1),
rest(Word,Char1,Words).
```

Το text
- παίρνει το πρώτο χαρακτήρα της πρότασης του χρήστη
- χρησιμοποιεί το word για να διαβάσει τη πρώτη λέξη από την πρόταση του χρήστη
- και χρησιμοποιεί το rest για να διαβάσει την υπόλοιπη πρόταση την οποία και θα τη μετατρέψει σε μια λίστα

```
rest('.',_,[]):-!.
```

Το rest τελειώνει μόλις βρει στην πρόταση τελεία

```
rest(Word,Char,[Word1|Words]):-
word(Char,Word1,Char1),
rest(Word1,Char1,Words).
```

Το δεύτερο rest:
- διαβάζει μια λέξη μέσω του word
- και διαβάζει τις εναπομείναντες λέξεις της πρότασης

```
word(46,'.',_):-!.
```

Η τελεία αποτελεί και η ίδια λέξη

```
word(Char,Word,Char2):-
check(Char),!,
get0(Char1),
restw(Char1,Chars,Char2),
name(Word,[Char|Chars]).
```

Η πλειονότητα των λέξεων θα αντιμετωπίζεται από αυτή τη μορφή του word η οποία και:
- ελέγχει την κανονικότητα των χαρακτήρων
- με το get0 παίρνει τον επόμενο χαρακτήρα
- χρησιμοποιεί το restw για να διαβάσει το υπόλοιπο της λέξεως
- χρησιμοποιεί το name για να μετατρέψει τη λέξη ως τμήμα λίστας

```
word(Char,Word,Char2):-
get0(Char1),
word(Char1,Word,Char2).
```

Αυτή η μορφή του word χρησιμοποιείται για τη χρήση κενών μεταξύ των λέξεων

```
restw(Char,[Char|Chars],Char2):-
check(Char),!,
get0(Char1),
restw(Char1,Chars,Char2).
```

Το restw χρησιμοποιείται για να διαβαστεί το υπόλοιπο της λέξεως

```
restw(Char,[],Char).
```

Εδώ τελειώνει το restw μόλις βρεθεί κενό και έτσι αναγνωρίζονται και διαχωρίζονται μεταξύ τους οι λέξεις

```
check(Char):-
Char>96,
Char<123.
```

Το check ελέγχει την κανονικότητα των χαρακτήρων που φτιάχνουν μια λέξη – προσοχή επιτρέπονται μόνο μικρά γράμματα όχι κεφαλαία

```
response([],[]).
response([H|T],[L|M]):-
transform(H,L),
response(T,M).
```

Με το response ενεργοποιείται και παράγεται μια απάντηση στην ερώτηση του χρήστη. Η απάντηση αυτή χρησιμοποιεί την transform, και τις λέξεις που έρχονται ως απάντηση στις λέξεις της λίστας της ερώτησης του χρήστη, τις ομαδοποιεί δημιουργώντας μια νέα λίστα

```
transform(are,no).
transform(you,'I am').
transform(computer,person).
transform(man,woman).
transform(tired,awake).
transform(foolish,intelligent).
transform(intelligent,stupid).
transform(how,'').
transform(is,'').
transform(weather,'weather is fine').
transform(what,'').
transform(your,my).
transform(name,'name is joujoukos').
transform(i,'').
transform(am,hello).
transform(X,X).
```

Το transform χρησιμοποιείται για να αλλάξει μια σειρά λέξεων του χρήστη σε αντίστοιχες με τις οποίες θα απαντά ο Η/Υ
- με το τελευταίο transform κάθε λέξη που θα χρησιμοποιεί ο χρήστης μπορεί να χρησιμοποιηθεί από τον Η/Υ στην απάντησή του

```
print_list([]):-
nl.
print_list([H|T]):-
write(H),
tab(1),
print_list(T).
```

Το print_list χρησιμοποιείται για την εμφάνιση κατά την εκτέλεση της εφαρμογής, της απάντησης του Η/Υ, η οποία όπως προείπαμε μέσω response υφίσταται ως λίστα λέξεων.
- Γράφει την κεφαλή της λίστας κάθε φορά και από τις υπόλοιπες λέξεις δημιουργεί μια καινούργια λίστα εως ότου αυτή καταλήξει κενή

Ερώτηματα:
Τώρα ρωτήστε:

A

```
?- english.
User> hello there.
Computer> hello there .

User> |: i am chris.
Computer>  hello chris .

User> |: what is your name.
Computer>   my name is joujoukos .

User> |: are you a computer.
Computer> no I am a person .

User> |: are you intelligent.
Computer> no I am stupid .

User> |: how is the weather today.
Computer>   the weather is fine today .

User> |: are you a man.
Computer> no I am a woman .
```

User> |: will you talk to me.
Computer> will I am talk to me .

User> |: are you foolish.
Computer> no I am intelligent .

User> |: i do not think so.
Computer> do not think so .

User> |: bey now.
Computer> bey now .

User> |: finish.
Yes

B
?- english.
User> jo bo yo.
Computer> jo bo yo .

User> |: good boy.
Computer> good boy .

User> |: good man.
Computer> good woman .

User> |: try hard.
Computer> try hard .

User> |: you are faoul.
Computer> I am no faoul .

User> |: finish.
Yes

Z4. Πρόγραμμα Έμπειρου Συστήματος για την διάγνωση ασθενιών

1. Θεωρητική ανάλυση παραδείγματος

Στη συνέχεια ακολουθεί ένα ιατρικό Ε.Σ το οποίο ρωτάει το χρήστη αν παρατηρούνται κάποια συμπτώματα και ανάλογα δίνει τις πιθανές διαγνώσεις. Αυτό αποτελείται από τρία κατηγορήματα :

συμπέρασμα, symperasma(X,Y)
σύμπτωμα, symptom(C,B).
αρρώστια, arrostia(G).

Το ***κατηγόρημα συμπέρασμα*** βοηθάει στο ν' ανακαλύπτει το πρόγραμμα πληροφορίες οι οποίες είναι γνωστές ή υπονοούνται από το χρήστη.

Π.χ. ο *πολύς πυρετός* φέρνει *υπνηλία*
symperasma(ipnilia,arketa):- symptom(pyreto,poly).

και ο *πολύς βήχας* σημαίνει και *αρκετό βήχα*
symperasma(X,arketa):- symptom(X,poly).

Το κατηγόρημα *σύμπτωμα προσφέρει* την επικοινωνία με το χρήστη. Έτσι, όταν το πρόγραμμα χρειάζεται να βρει το σύμπτωμα του πυρετού, το κατηγόρημα *σύμπτωμα* θα ελέγξει κατ' αρχήν αν ο χρήστης έχε *ήδη* απαντήσει σχετικά με αυτό (είτε αρνητικά είτε θετικά).

```
symptom(X,Y):-
arnitiki(X,Y),
!,
fail.

symptom(X,Y):-
gnosto(X,Y),
!.

symptom(X,Y):-
symperasma(X,Y),
!.
```

Αν καμιά ερώτηση σχετικά με αυτό το σύμπτωμα, δεν έχει υποβληθεί στο χρήστη, το κατηγόρημα *σύμπτωμα* θα προσπαθήσει να τη συμπεράνει. Αν πάλι δεν τα καταφέρει, θα υποβάλλει (μόνον τότε) την αντίστοιχη ερώτηση.

```
symptom(X,W):-
write('Exoume '),
write(W),
write(' '),
write(X),
write(' (y/n); '),
read(y),
!,
assert(gnosto(X,W)).
```

Το πρόγραμμα θα διαβάσει την απάντηση με την εντολή *read* και θα προσπαθήσει να την ενοποιήσει με τη **y.** που εμφανίζεται σαν παράμετρος.

Αν η ενοποίηση είναι επιτυχής (αν δηλαδή ο χρήστης πληκτρολόγησε το ***y.***, που σημαίνει *ναι)* τότε θα δημιουργηθεί ένα δυναμικό γεγονός *γνωστό* (το οποίο θα χρειαστεί ενδεχομένως το πρόγραμμα για ν' αποδείξει άλλες αρρώστιες) και το σύμπτωμα θα θεωρηθεί ως γνωστό.

Αν η απάντηση του χρήστη ήταν αρνητική τότε θα δηλωθεί το δυναμικό γεγονός *αρνητική* και το κατηγόρημα *σύμπτωμα* θ' αποτύχει. Με αυτό τον τρόπο το πρόγραμμα δεν θα ξαναρωτήσει το χρήστη σχετικά μ' αυτό το σύμπτωμα.

```
symptom(X,Y):-
assert(arnitiki(X,Y)),
fail.
```

Το κατηγόρημα *αρρώστια* υλοποιείται από τους ιατρικούς κανόνες του Ε.Σ.
Π.χ. :

```
arrostia(monopirinosi):-
symptom(adenes,prismeni),
symptom(kourasi,arketa),
symptom(ipnilia,arketa).
```

Δηλαδή υπάρχει περίπτωση *λοφώδους μονοπυρήνωσης* όταν εντοπισθούν *πρησμένοι αδένες* και παρατηρηθεί αρκετή *υπνηλία* και *κούραση* στον άρρωστο.
ή

```
arrostia(vronxitida):-
arrostia(kriologima),
symptom(bixa,poly).
```

Δηλαδή ο ασθενής παθαίνει *βρογχίτιδα* όταν έχει ήδη υποστεί *κρυολογήμα* που ακολουθείται από πολύ *βήχα.*

<u>2. Ο κώδικας του προγράμματος</u>

```
symperasma(X,arketa):-
symptom(X,poly),
!.
symperasma(ponolaimos,arketa):-
symptom(bixa,arketa),
!.

symperasma(kourasi,arketa):-
symptom(pyreto,poly),
!.

symperasma(ipnilia,arketa):-
symptom(pyreto,poly).

/*-------------------------*/
symptom(X,Y):-
arnitiki(X,Y),
!,
fail.

symptom(X,Y):-
gnosto(X,Y),
!.

symptom(X,Y):-
symperasma(X,Y),
!.

symptom(X,W):-
write('Exoume '),
write(W),
write(' '),
write(X),
write(' (y/n); '),
read(y),
!,
assert(gnosto(X,W)).
```

```
symptom(X,Y):-
assert(arnitiki(X,Y)),
fail.

/*-------------------*/

arrostia(kriologima):-
symptom(pyreto,arketa),
symptom(ponolaimos,arketa),
symptom(bixa,arketa).

arrostia(kriologima):-
symptom(kourasi,arketa),
symptom(miti,treχei).

arrostia(kriologima):-
symptom(diarroia,arketa),
symptom(pyreto,arketa).

/*-------------------*/

arrostia(ilara):-
symptom(spirakia,poly),
symptom(pyreto,_).

/*-------------------*/

arrostia(ipatitida):-
symptom(derma,kitrino),
symptom(oura,kokkina).

/*-------------------*/

arrostia(lariggitida):-
not(symptom(pyreto,_)),
symptom(biχa,poly),
symptom(ponolaimos,poly).

/*-------------------*/

arrostia(monopirinosi):-
symptom(adenes,prismeni),
symptom(kourasi,arketa),
symptom(ipnilia,arketa).

/*-------------------*/

arrostia(kokkitis):-
symptom(biχa,poly),
symptom(ponolaimos,poly),
symptom(pyreto,arketa).

/*-------------------*/

arrostia(elonosia):-
symptom(ipnilia,poly),
symptom(pyreto,poly).

/*-------------------*/
```

```
arrostia(anafilaxia):-
not(symptom(pyreto,_)),
symptom(spirakia,poly).

/*-------------------*/

arrostia(anemia):-
symptom(kourasi,poly),
symptom(adenes,prismeni),
symptom(derma,kitrino).

/*-------------------*/

arrostia(vronxitida):-
arrostia(kriologima),
symptom(bixa,poly).

/*-------------------*/

arrostia(dilitiriasi):-
not(arrostia(kriologima)),
symptom(diarroia,poly).

/*-------------------*/

go(X):-
assert(arnitiki(_,_)),
assert(gnosto(_,_)),
retractall(gnosto(_,_)),
retractall(arnitiki(_,_)),
arrostia(X).
```

3. Ερωτήματα

Οι ερωτήσεις που μπορούμε να κάνουμε είναι της παρακάτω μορφής. Επίσης με έντονα πλάγια γράμματα είναι οι απαντήσεις του χρήστη.

A.

```
?- go(X).
Exoume poly pyreto (y/n); n.
Exoume arketa pyreto (y/n); n.
Exoume poly kourasi (y/n); y.
Exoume trexei miti (y/n); n.
Exoume poly diarroia (y/n); y.
Exoume poly spirakia (y/n); n.
Exoume kitrino derma (y/n); n.
Exoume poly bixa (y/n); n.
Exoume prismeni adenes (y/n); n.
Exoume poly ipnilia (y/n); n.
X = dilitiriasi
No
```

B.

```
?- go(X).
Exoume poly pyreto (y/n); y.
Exoume poly ponolaimos (y/n); n.
Exoume poly bixa (y/n); n.
Exoume arketa bixa (y/n); n.
Exoume arketa ponolaimos (y/n); n.
Exoume poly kourasi (y/n); y.
Exoume trexei miti (y/n); n.
```

Exoume poly diarroia (y/n); ***y***.
X = kriologima;
Exoume poly spirakia (y/n); ***n***.
Exoume kitrino derma (y/n); ***n***.
Exoume prismeni adenes (y/n); ***y***.
Exoume poly ipnilia (y/n); ***y***.
X = monopirinosi;
Exoume poly biχa (y/n); **n**.
X = elonosia ;
Exoume kitrino derma (y/n); **n.**
No

Z5. Εφαρμογή Έμπειρου Συστήματος για την επιλογή αυτοκινήτου

Το παρακάτω πρόγραμμα είναι ένα μικρό έμπειρο σύστημα για επιλογή αυτοκινήτου ανάλογα των χαρακτηριστικών που επιθυμεί ο πιθανός αγοραστής να έχει το αυτοκίνητο. Αποτελείται από το γεγονός χωρίς κατηγόρημα car_select μέσω του οποίου γίνεται η διάγνωση των επιθυμιών του πιθανού αγοραστή.

Η χρήση της write_list και η λειτουργία της όπως φαίνεται στο τέλος του κώδικα έχει ως στόχο την επιλογή λέξεων από λίστα οι οποίες όμως στην prolog όταν θα τρέχει η εφαρμογή, θα σχηματίζουν τις ερωτήσεις προς τον αγοραστή.

Το γεγονός car περιγράφει τα αυτοκίνητα με χαρακτηριστικά αντίστοιχα αυτών που ερωτάται ο πιθανός αγοραστής. Το συγκεκριμένο γεγονός εισάγει γεγονότα στη βάση γνώσεων της εφαρμογής.

Τέλος, το γεγονός find_car είναι αυτό που τεστάρει και συγκρίνει τις επιλογές του αγοραστή με τα στοιχεία της βάσης γνώσης της εφαρμογής και τελικά είναι αυτό το γεγονός που θα φέρει τη λύση - πρόταση προς τον αγοραστή.

```
car_select:-
write_list(['Answer',the,following,questions]),
nl,
nl,
write_list(['How',much,do,you,wish,to,'spend?']),
nl,
read(Amount),
write_list(['Do',you,want,a,saloon,car,or,an,estate,'car?']),
nl,
read(Type),
write_list(['Do',you,want,a,british,',',american,or,japanese,'car?']),
nl,
read(Country),
find_car(Amount,Make,Type,Country,Cost),
write_list(['The',most,suitable,car,is,a]),
write(Make),
tab(1),
write_list([which,costs]),
write('Euro'),
write(Cost),
nl.

find_car(Amount,Make,Type,Country,Cost):-
car(Make,Type,Country,Cost),
Cost=<Amount.

car('Toysun',saloon,japanese,4000).
car('Toysun',estate,japanese,4500).
car('Ford',saloon,american,4000).
car('Ford',estate,american,4500).
car('Roller',saloon,british,50000).
car('Uaston',saloon,british,4500).
car('Uaston',estate,british,5000) .

write_list([]).
write_list([H|T]):-
write(H),
tab(1),
write_list(T).
```

Ερώτηματα:
Εκτελεστέ

Α)
?- car_select.
Answer the following questions

How much do you wish to spend?
|: 4000.
Do you want a saloon car or an estate car?
|: saloon.
Do you want a british , american or japanese car?
|: american.
The most suitable car is a Ford which costs Euro4000
Yes

Β)
?- car_select.
Answer the following questions

How much do you wish to spend?
|: 3000.
Do you want a saloon car or an estate car?
|: saloon.
Do you want a british , american or japanese car?
|: japanese.
No

Γ)
?- car_select.
Answer the following questions

How much do you wish to spend?
|: 40000.
Do you want a saloon car or an estate car?
|: saloon.
Do you want a british , american or japanese car?
|: british.
The most suitable car is a Uaston which costs Euro4500
Yes

Δ)
?- car_select.
Answer the following questions

How much do you wish to spend?
|: 5000.
Do you want a saloon car or an estate car?
|: estate.
Do you want a british , american or japanese car?
|: american.
The most suitable car is a Ford which costs Euro4500
Yes

Z6. Έμπειρο σύστημα εύρεσης είδους ζώου ανάλογα με τα χαρακτηριστικά που δηλώνει ο χρήστης ότι είδε.

Το παρόν πρόγραμμα προσπαθεί να αναγνωρίσει ποιο ζώο είδε ο χρήστης, ανάλογα τα χαρακτηριστικά που δίνει αυτός κατά την επικοινωνία με το έμπειρο σύστημα.

Κώδικας του προγράμματος (ταυτόχρονος σχολιασμός).

```
positive(X,Y):-
        xnegative(X,Y),!,fail.
positive(X,Y):-
        xpositive(X,Y),!.
```

κάτι δεν είναι θετικό ως γεγονός, δηλαδή δεν υπάρχει ως χαρακτηριστικό στο ζώο, όταν βρίσκεται εισαγμένο ως αρνητικό στο xnegative. Επίσης, μόλις βρεθεί αρνητικό θα πρέπει να σταματά η αναζήτηση.

κάτι είναι θετικό όταν χρεώνεται ως θετικό στο xpositive, άρα υπάρχει ως χαρακτηριστικό στο ζώο. Κάποιο χαρακτηριστικό ανήκει στο ζώο που είδε ο χρήστης όταν δηλώνεται κατά την παρακάτω επικοινωνία του με το έμπειρο.

```
positive(E,W):-
write('The animal '),
write(E),
write(' '),
write(W),
write(' (y/n); '),
read(y),
!,
assert(xpositive(E,W)).
```

παραπάνω φαίνεται πως είναι ο τρόπος επικοινωνίας του χρήστη με τον Η/Υ, στα Ε και W θα μπαίνουν τα ρήματα και τα χαρακτηριστικά που υπάρχουν στους κανόνες ορισμού των ζώων.

```
positive(X,W):-
assert(xnegative(X,W)),
fail.
```

οτι δεν εισάγεται ως θετικό στο xpositive τότε να χρεώνεται ως xnegative

```
 animal_is(cheetah):-
        it_is(mammal),
        it_is(carnivore),
        positive(has,tawny_color),
        positive(has,dark_spots).

 animal_is(tiger):-
        it_is(mammal),
        it_is(carnivore),
        positive(has,tawny_color),
        positive(has,black_stripes).

 animal_is(giraffe):-
        it_is(ungulate),
        positive(has,long_neck),
        positive(has,long_legs),
        positive(has,dark_spots).

 animal_is(zebra):-
```

```
        it_is(ungulate),
        positive(has,black_stripes).

 animal_is(ostrich):-
        it_is(bird),
        not(positive(does,fly)),
        positive(has,long_neck),
        positive(has,long_legs),
        positive(has,black_and_white_color).

 animal_is(penguin):-
        it_is(bird),
        not(positive(does,fly)),
        positive(does,swim),
        positive(has,black_and_white_color).

 animal_is(albatross):-
        it_is(bird),
        positive(does,fly_well).

 it_is(mammal):-
        positive(has,hair).
 it_is(mammal):-
        positive(does,give_milk).

 it_is(bird):-
        positive(has,feathers).
 it_is(bird):-
        positive(does,fly),
        positive(does,lay_eggs).

 it_is(carnivore):-
        positive(does,eat_meat).

 it_is(carnivore):-
        positive(has,pointed_teeth),
        positive(has,claws),
      positive(has,forward_eyes).

 it_is(ungulate):-
        it_is(mammal),
        positive(has,hooves).

 it_is(ungulate):-
        it_is(mammal),
        positive(does,chew_cud).
```

Τόσο το animal_is όσο και το it_is εξαρτώνται από τα positive δηλαδή από τι τελικά αποδέχεται ο χρήστης κατά τη επικοινωνία του με τον Η/Υ εκεί που εμφανίζονται οι read, και write.

```
run(V):-
assert(xnegative(_,_)),
assert(xpositive(_,_)),
retractall(xpositive(_,_)),
retractall(xnegative(_,_)),
animal_is(V).
```

η run βρίσκει το ζώο αφού εμπεριέχει την animal_is, βάσει των χαρακτηριστικών που δέχεται ο χρήστης κατά την αναζήτηση στα positive. Ταυτόχρονα αφού τελειώσει η εύρεση του ζώου αναιρεί από τη βάση γνώσης τις xnegative και χpositive

Πως λειτουργεί το πρόγραμμα.

Αφού τρέξει το run(N) τότε πάει ο Η/Υ να εισάγει ως γνώση κάποιο xnegative, μετά πάει να εισάγει κάποιο xpositive, αρχικά όμως αφού δεν υπάρχει τίποτα δεν εισάγει τίποτα στη βάση γνώσης. Μετά πάει να αναιρέσει τις xpositive και xnegative για ελευθέρωση του buffer. Έπειτα τρέχει την animal_is για να ενοποιήσει το N της run με κάποια από τις σταθερές που η animal_is έχει. Δηλαδή ξεκινά με το cheetah και πάει να ζητήσει από το χρήστη αν ισχύει ο πρώτος κανόνας αυτού που είναι το it_is. Επειδή, οι it_is βασίζονται στις positive τελικά ικανοποιούναι αυτές ή όχι αναλόγως με το τι απαντά ο χρήστης για τα αντικείμενα των positive. Αυτό ισχύει επίσης όταν η animal_is εξαρτάται απευθείας από τις positive.
Έτσι με την αναδρόμηση που υπάρχει ως χαρακτηριστικό στη PROLOG, αναλόγως τι απαντά ο Χρήστης, βλέπει ο Η/Υ αν ικανοποιούνται τα αντικείμενα των positive. Αν δεν ισχύουν όλα για το cheetah θα πάει το έμπειρο να δει αν ικανοποιούνται αυτά για το tiger κ.ο.κ.

Ερώτηματα:
Τώρα ρωτήστε:
Α.
?-run(N).
The animal has hair (y/n); **n.**
The animal does give_milk (y/n); **n.**
The animal has feathers (y/n); **y.**
The animal does fly (y/n); **y.**
The animal does lay_eggs (y/n); **y.**
The animal does fly_well (y/n); **y.**

N = albatross **;**

Β.
?- run(O).
The animal has hair (y/n); **y.**
The animal does eat_meat (y/n); **y.**
The animal has tawny_color (y/n); **y.**
The animal has dark_spots (y/n); **y.**

O = cheetah **;**
The animal has pointed_teeth (y/n); **n.**
The animal does give_milk (y/n); **n.**
The animal has black_stripes (y/n); **y.**

O = tiger **;**
The animal has hooves (y/n); **n.**
The animal does chew_cud (y/n); **n.**
The animal has feathers (y/n); **n.**
The animal does fly (y/n); **n.**
No

Z7. Έμπειρο σύστημα τουριστικού οδηγού εύρεσης Pub

Το παρόν πρόγραμμα είναι ένα έμπειρό σύστημα το οποίο μέσα από τη χρήση μενού επιλογών δίνει τη δυνατότητα στον τελικό χρήστη να επιλέγει pub βάσει συγκεκριμένων επιθυμιών του σχετικά με τη περιοχή που βρίσκεται και των υπηρεσιών που προσφέρει η κάθε pub.

Τρία είδη ερωτήσεων μπορεί να κάνει ο τελικός χρήστης – τουρίστας

1. εύρεση μιας Pub ανάλογα με την περιοχή που βρίσκεται, οπότε το σύστημα δείχνει τις Pub που βρίσκονται στην περιοχή που επιλέγεται.
2. εύρεση μιας Pub ανάλογα με τις υπηρεσίες που προσφέρει η pub, οπότε όταν επιλεγεί συγκεκριμένη υπηρεσία το σύστημα θα δείξει τις Pub που διαθέτουν αυτή την υπηρεσία
3. εύρεση των υπηρεσιών μιας Pub δίνοντας το όνομά της

Κώδικας του προγράμματος (ταυτόχρονος σχολιασμός).

```
pubguide:-nl,nl,
write('The following options are available:'),nl,nl,nl,nl,
write(' To find which pubs are in each area of Anytown'),
tab(10),
write('1'),nl,
write('To find which pubs have certain amenities' ),
tab(15),
write('2'),nl,
write('To find the amenities available at a given pub'),
tab(10),
write('3'),
nl,nl,nl,nl,
write('Enter option 1 , 2 or 3 '),
nl,nl,
read(Opt),
option(Opt).
```

Το pubguide είναι το αρχικό γεγονός που εκτελεί την όλη εφαρμογή και δίνει το αρχικό μενού με τις τρεις επιλογές – είδη ερωτήσεων που μπορεί να ρωτήσει ο τελικός χρήστης το σύστημα.

```
option(1):-nl,
write('Areas of Anytown'),nl,nl,nl,
areas(Arealist),
write_menu(Arealist,1),
nl,nl,
write('Type in a number to choose a particular area '),nl,nl,
read(Areano),
option1(Areano).
```

Εδώ φαίνεται τι θα γίνει (τι θα εμφανιστεί ως δεύτερο μενού) όταν ο χρήστης επιλέξει την επιλογή 1 από το αρχικό μενού της pubguide.

```
option1(Areano):-nl,
write('List of pubs in '),
areanumb(Area,Areano),
write(Area),
write(' area'),nl,nl,nl,
pub(Area,Pub,Amenity),
write(Pub),nl,
fail.
```

Η option1 πάει στη βάση γνώσης που ακολουθεί και ψάχνει να επιστρέψει τις Pub που ικανοποιούν την επιλογή που έκανε ο χρήστης κατά την εκτέλεση της option(1), βάσει της περιοχής που επιλέγεται.

```
option(2):-nl,
write('List of Amenities'),
nl,nl,nl,
amenities(Amenlist),
write_menu(Amenlist,1),
nl,nl,
write('Type a number to choose an amenity'),nl,nl,
read(Ameno),
option2(Ameno).
```

Εδώ φαίνεται τι θα γίνει (τι θα εμφανιστεί ως δεύτερο μενού) όταν ο χρήστης επιλέξει την επιλογή 2 από το αρχικό μενού της pubguide.

```
option2(Ameno):-nl,
write('List of pubs with '),
amenumb(Amenity,Ameno),
write(Amenity),
nl,nl,nl,
pub(Area,Pub,Amenlist),
member(Amenity,Amenlist),
write(Pub),nl,
fail.
```

Η option2 πάει στη βάση γνώσης που ακολουθεί και ψάχνει να επιστρέψει τις Pub που ικανοποιούν την επιλογή που έκανε ο χρήστης κατά την εκτέλεση της option(2), βάσει των υπηρεσιών που προσφέρουν οι Pub

```
option(3):-nl,
write('Amenities available at a given pub'),nl,nl,nl,
write('Type in name of pub'),
nl,nl,
read(Pub),
nl,nl,
pub(Area,Pub,Amenlist),
write_list(Amenlist).
```

Εδώ φαίνεται τι θα γίνει (τι θα εμφανιστεί ως δεύτερο μενού) όταν ο χρήστης επιλέξει την επιλογή 3 από το αρχικό μενού της pubguide.

```
write_menu([],_).
write_menu([H|T],I):-
write(H),
tab(10),
write(I),
nl,
J is I+1,
write_menu(T,J).
```

Το γεγονός write_menu δείχνει πως μπορεί να γράφεται ένα μενού επιλογών ανάλογα τι αριθμός έχει επιλεγεί από τα μενού κατά την εκτέλεση της pubguide.

```
write_list([]).
write_list([H|T]):-
write(H),
nl,
write_list(T).
```

Το γεγονός write_list δείχνει πως μπορεί να γράφεται ένα στοιχείο του μενού κατά την εκτέλεση της pubguide.

```
member(X,[X|_]).
member(X,[_|Y]):-member(X,Y).
```

Το member ψάχνει αν κάποιο στοιχείο που θα επιλεγεί από το χρήστη είναι μέρος λίστας. Αναφέρεται κυρίως στην εύρεση υπηρεσιών μιας pub.

```
areas([town_centre,warwick_road,paddock_lane]).
```

Στο areas υπάρχει η λίστα με τις περιοχές της πόλης που υπάρχουν pub.

```
amenities([restaurant,music,accommodation]).
```

Στο amenities υπάρχει η λίστα με τα είδη υπηρεσιών των pub που υπάρχουν.

```
pub(town_centre,red_lion,[restaurant,music]).
pub(town_centre,black_horse,[music]).
pub(warwick_road,kings_head,[accommodation,restaurant]).
pub(warwick_road,sailors_rest,[music]).
pub(warwick_road,gyradiko,[]).
pub(paddock_lane,pennies,[]).
pub(paddock_lane,castles,[restaurant]).
pub(paddock_lane,legionnaire,[music]).
```

Στο pub υπάρχει η λίστα με το τι χαρακτηριστικά έχουν οι Pub, σε ποια περιοχή βρίσκονται, ποιο όνομα εχουν, και τελικά τι υπηρεσίες προσφέρουν.

```
areanumb(town_centre,1).
areanumb(warwick_road,2).
areanumb(paddock_lane,3).
amenumb(restaurant,1).
amenumb(music,2).
amenumb(accommodation,3).
```

Η areanumb καλείται τόσο στην option1 όσο και για στην option2. Έτσι ανάλογα ποιο νούμερο θα επιλέξουμε θα αναζητειται το χαρακτηριστικό που ο χρήστης θέλει να έχει η Pub που επιθυμεί να πάει.

Το μυστικό σε αυτό το έμπειρο σύστημα είναι μέσα από μενού επιλογών και με την εισαγωγή αριθμών να επιτυγχάνεται η ικανοποίηση της επιλογής Pub από το χρήστη. Σημαντικό γεγονός είναι το write_menu το οποίο ανάλογα με το τι αριθμός επιλέγεται εμφανίζει τα στοιχεία του μενού επιλογών που θα ακολουθήσει.

Ερώτηματα:
Τώρα ρωτήστε:

Α.

```
?- pubguide.
The following options are available:
To find which pubs are in each area of Anytown          1
To find which pubs have certain amenities               2
To find the amenities available at a given pub          3

Enter option 1 , 2 or 3
|: 1.

Areas of Anytown
town_centre          1
warwick_road          2
paddock_lane          3

Type in a number to choose a particular area
|: 2.

List of pubs in warwick_road area
kings_head
sailors_rest
gyradiko
No
```

Β.

```
?- pubguide.
The following options are available:
To find which pubs are in each area of Anytown          1
To find which pubs have certain amenities               2
To find the amenities available at a given pub          3

Enter option 1 , 2 or 3
|: 2.

List of Amenities
restaurant          1
music          2
accommodation          3

Type a number to choose an amenity
|: 3.

List of pubs with accommodation
kings_head
No
```

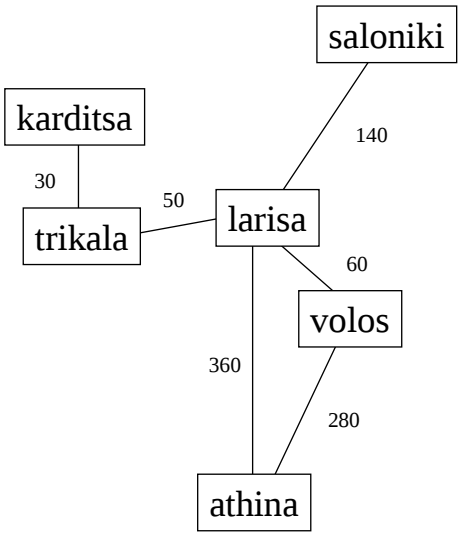
Γ.

```
?- pubguide.
The following options are available:
To find which pubs are in each area of Anytown          1
To find which pubs have certain amenities               2
To find the amenities available at a given pub          3
```

```
Enter option 1 , 2 or 3
|: 3.

Amenities available at a given pub
Type in name of pub
|: kings_head.

accommodation
restaurant
Yes
```

Z8. Έμπειρο σύστημα διαδρομών πωλητή

Προς συμπλήρωση των αλγορίθμων αναζήτησης που παρατέθηκαν στην παράγραφο Z.1. το παρόν έμπειρο σύστημα καλύπτει την επιλογή διαδρομών μεταξύ διαφόρων πόλεων. Έτσι ο πωλητής μέσω του συγκεκριμένου έμπειρού συστήματος μπορεί να δεί τις υπάρχουσες συνδέσεις μεταξύ των πόλεων, να δεί την ακριβή διαδρομή πηγαίνοντας από τη μία στην άλλη και να επιλέξει διαδρομή ανάλογα με την απόστασή.



Κώδικας του προγράμματος (ταυτόχρονος σχολιασμός).

```
run:-nl,nl,
write(' Travelling salesman program'),nl,nl,nl,
write('Type in first town'),
nl,
read(Town1),
nl,nl,
write('Type in second town'),
nl,
read(Town2),
route(Town1,Town2,[Town1],Way,Dist),nl,nl,
write('One route from '),
write(Town1),
write(' to '),
write(Town2),
write(' is : '),nl,
write(Way),nl,nl,
write('The distance is '),
write(Dist),nl,nl,
write('Do you wish to find another route ?'),nl,
write('Answer yes or no '),nl,
read(Response),
act_on(Response).
```

Το run είναι το κεντρικό γεγονός και παρέχει τις ερωτήσεις για να μπορέσει ο πωλητής να δώσει τη πόλη αναχώρησης και την πόλη άφιξης για το ταξίδι που πρόκειται να ξεκινήσιε.

act_on(yes):-nl,fail.
act_on(no).

Ανάλογα τι θα απαντηθεί από τον τελικό χρήστη στη τελευταία ερώτηση της run το γεγονός act_on θα οδηγήσει σε επαναεκτέλεση της route για την εύρεση και άλλων διαδρομών μεταξύ των δύο πόλεων ή θα τερματίσει το σύστημα.

route(Town,Town,_,[Town],0):-!.

route(Town1,Town2,Covered,[Town1|Way],Dist):-
go(Town1,Town3,Dist1),
\+member(Town3,Covered),
route(Town3,Town2,[Town3|Covered],Way,Dist2),
Dist is Dist1 + Dist2.

Το route(Town1,Town2,Covered,Way,Dist) επιστρέφει διαδρομή και απόσταση όταν υπάρχει διαδρομή μεταξύ των δύο πόλεων (Town1 προς Town2). Η Covered είναι η λίστα με τις πόλεις που μέσω του route **ως μια συγκεκριμένη στιγμή** έχουν ενταχθεί στη διαδρομή κατά την εύρεση, ξεκινώντας από την Town1, της Town2 και της απόστασή μεταξύ των δύο. Η Way είναι η λίστα με τις πόλεις βρέθηκαν τελικώς για την ολοκλήρωση της διαδρομής μεταξύ των Town1 και Town2. Η Dist είναι η τελική απόσταση μεταξύ Town1 και Town2.

member(X,[X|_]).
member(X,[_|Y]):-member(X,Y).

Το member ψάχνει αν κάποια πόλη που θα βρεθεί κατά την ολοκλήρωση της διαδρομής μεταξύ των πόλεων άναχώρησης και άφιξης είναι ήδη τμήμα αυτής της διαδρομής ή όχι για να μπει.

go(larisa,volos,60).
go(volos,larisa,60).
go(larisa,trikala,50).
go(trikala,larisa,50).
go(karditsa,trikala,30).
go(trikala,karditsa,30).
go(larisa,a8ina,360).
go(a8ina,larisa,360).
go(volos,a8ina,280).
go(a8ina,volos,280).
go(larisa,8essaloniki,140).
go(8essaloniki,larisa,140).

Στη go βρίσκεται η βάση γνώσης για το ποιες πόλεις συνδέονται με άλλες και ποιες είναι αποστάσεις μεταξύ τους.

Ερώτηματα:
Τώρα ρωτήστε:
A.
?- run.
Travelling salesman program
Type in first town
|: larisa.

```
Type in second town
|: saloniki.

One route from larisa to saloniki is :
[larisa, saloniki]
The distance is 140

Do you wish to find another route ?
Answer yes or no
|: yes.
No
```

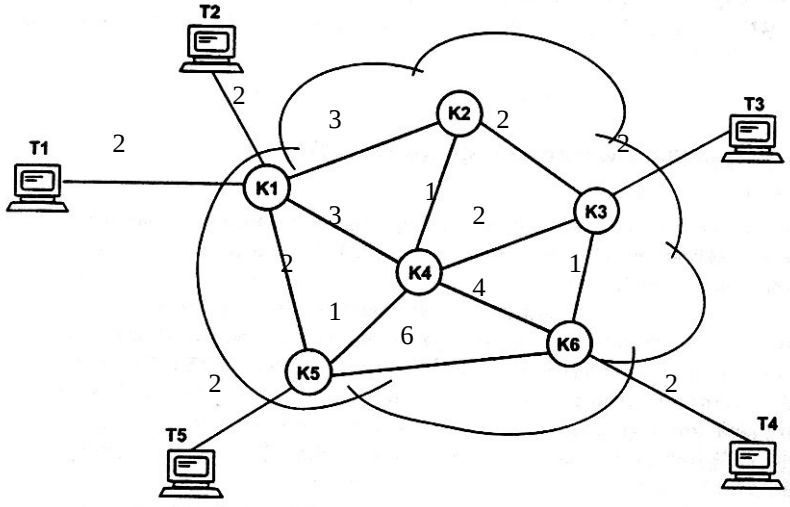
B.

```
?- run.

Travelling salesman program
Type in first town
|: karditsa.

Type in second town
|: athina.

One route from karditsa to athina is :
[karditsa, trikala, larisa, volos, athina]
The distance is 420

Do you wish to find another route ?
Answer yes or no
|: yes.

One route from karditsa to athina is :
[karditsa, trikala, larisa, athina]
The distance is 440

Do you wish to find another route ?
Answer yes or no
|: yes.
No
```

Η. ΑΣΚΗΣΕΙΣ ΣΤΗΝ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Άσκηση 1

Δημιουργείστε έμπειρο σύστημα το οποίο να υλοποιεί τις συνδέσεις μεταξύ τερματικών σταθμών και κόμβων όπως αυτοί φαίνονται στο σχήμα και να αναφέρει όταν ερωτάται για κάθε σύνδεση μεταξύ δύο μερών ποιες επιλογές υπάρχουν και το συνολικό μήκος καλωδίου που απαιτείται.



Άσκηση 2

Αναπτύξατε εφαρμογή προσομοίωσης της φυσικής γλώσσας με συσχετισμούς λέξεων τους παρακάτω:

- are - no
- you - 'I am'
- computer - person
- man - woman
- tired - awake
- foolish - intelligent
- intelligent – stupid
- how - '' δηλ. τίποτα
- is - '' δηλ. τίποτα
- weather - 'weather is fine'
- what - '' δηλ. τίποτα
- your - my
- name - 'name is jim'
- I - ''
- am - hello
- Κάτι – Κατι δηλ. Χ με Χ

Θα πρέπει να χρησιμοποιήσετε οπωσδήποτε τα προκαθορισμένα κατηγορήματα read, assert, write και να υλοποιήσετε ένα παρόμοιο μηχανισμό με την άσκηση Z3, ο οποίος:

1. θα λαμβάνει πρόταση από τον χρήστη θα την κάνει λίστα
2. το κάθε στοιχείο της λίστας θα το συσεχτίζει με κάποιο αντίστοιχο όπως φαίνεται στους παραπάνω συσχετισμούς
3. και τελικά θα απαντά ο Η/Υ στην αρχική πρόταση του χρήστη

Η συγκεκριμένη εργασία είναι συνδυασμός της εργασίας Z3 και της εργασίας 15 σελ 14.

Η επικοινωνία θα παρουσιάζει ένα πρόβλημα μπορείτε να σκεφτείτε ποιο θα είναι αυτό;

Άσκηση 3
Υποθέτουμε ότι θέλουμε να εμπλουτίσουμε το υπάρχον Ε.Σ. ώστε να είναι σε θέση να κάνει τη διάγνωση της αρρώστιας «μηνιγγίτιδα». Δοθέντος ότι αυτή η αρρώστια χαρακτηρίζεται από:
Η μύτη τρέχει.
Δεν υπάρχει βήχας και πονόλαιμος.
Υπάρχει πονοκέφαλος αρκετά.
Συμπληρώστε τον κώδικα του προγράμματος και εκτελέστε το ερώτημα

```
go(X).
Exoume poly pyreto (y/n); y.
Exoume poly ponolaimos (y/n); n.
Exoume poly bixa (y/n); n.
Exoume arketa bixa (y/n); n.
Exoume arketa ponolaimos (y/n); n.
Exoume poly kourasi (y/n); n.
Exoume treχei miti (y/n); y.

X = kriologima ;
Exoume poly diarroia (y/n); n.
Exoume arketa diarroia (y/n); n.
Exoume poly spirakia (y/n); n.
Exoume kitrino derma (y/n); n.
Exoume prismeni adenes (y/n); n.
Exoume poly biχa (y/n); n.
Exoume poly ipnilia (y/n); n.
Exoume trexei miti (y/n); y.
Exoume poly ponokefalos (y/n); y.

X = miniggitida ;
```

Άσκηση 4
Αναπτύξτε έμπειρο σύστημα που θα υποστηρίζει ένα κατάστημα ρούχων.

Πιο συγκεκριμένα η βάση γνώσης θα περιλαμβάνει ρούχα (το λιγότερο 7 γεγονότα), με

- το είδος τους (παντελόνι, μπλουζάκι, φούστα),
- το χρώμα (μαύρο, άσπρο),
- το μέγεθος (small, large, xl),
- τη χώρα προέλευσης (ιταλία, αμερική, γαλλία)
- και την τιμή σε Ευρώ.

Ορίστε τον τρόπο επικοινωνίας με τη χρήση λέξεων από λίστα, ώστε ο αγοραστής να ερωτάται:

- τι χρώμα το θέλει, το ρουχο
- σε ποιο μέγεθος,
- που να είναι ραμμένο
- και πόσα διαθέτει για να το αγοράσει.

Οι ερωτήσεις θα ξεκινούν με κεφαλαίο γράμμα.

Ορίστε τον τρόπο εύρεσης του ρούχου σύμφωνα με τις απαιτήσεις του πελάτη και εμφανίστε την απάντηση του Η/Υ με τρόπο:

Το κατάλληλο ρούχο είναι το οποίο κοστίζει Ευρώ.

Προτείνετε, σύνολο απαιτήσεων του πελάτη ώστε να ασκήσετε ένα ερώτημα στο έμπειρο και να λειτουργήσει αυτό.

Άσκηση 5
Αναπτύξτε ένα έμπειρο σύστημα για τη διάγνωση προβλημάτων σε ένα αυτοκίνητο. Το έμπειρο σύστημα που θα υλοποιηθεί, αντιμετωπίζει ως ενδεχόμενο αρχικά την ύπαρξη βλάβης στο *σύστημα εκκίνησης του αυτοκινήτου*.

Έτσι το σύστημα εκκίνησης έχει **βλάβη** όταν:
1. Το αυτοκίνητο **δεν** παίρνει μπροστά και η μίζα δεν γυρνάει. (ενδείξεις)
2. Το αυτοκίνητο **δεν** παίρνει μπροστά και η μίζα γυρνάει και το ρεζερβουαρ είναι γεμάτο. (ενδείξεις)
3. Το ρεζερβουαρ είναι γεμάτο και το αυτοκίνητο μουγκρίζει. (ενδείξεις)
4. Το αυτοκίνητο παίρνει μπροστά και το ρεζερβουαρ είναι γεμάτο και το αυτοκίνητο ξεκινά αλλά σβήνει η μηχανή μετά από λίγο. (ενδείξεις)

Αφού διαπιστωθεί βλάβη στο σύστημα εκκίνησης το έμπειρο σύστημα θα πρέπει να διαγνώσει το πρόβλημα σε κάποιο από τα συγκεκριμένα κομμάτια από τα οποία αποτελείται το σύστημα εκκίνησης.
Τα κομμάτια από τα οποία αποτελείται είναι τα εξής:
- Μίζα
- Μπαταρία
- Χρονισμός Μηχανής
- Μπουζί
- Καλώδια εκκίνησης

Κάθε μηχανικό μέρος από τα παραπάνω έχει **προβλήμα** βάση των παρακάτω μηχανολογικών **ενδείξεων**:
- Η μίζα είναι χαλασμένη όταν το σύστημα εκκίνησης είναι χαλασμένο και τα φώτα ανάβουν.
- Η μπαταρία είναι χαλασμένη όταν το σύστημα εκκίνησης είναι χαλασμένο και τα φώτα **δεν** ανάβουν.
- Υπάρχει πρόβλημα χρονισμού όταν το σύστημα εκκίνησης είναι χαλασμένο και το αυτοκίνητο **δεν** έχει πάει συνεργείο.
- Υπάρχει πρόβλημα στα μπουζί όταν το σύστημα εκκίνησης είναι χαλασμένο και τα μπουζί **δεν** είναι καινούρια.
- Υπάρχει πρόβλημα στα καλώδια εκκίνησης όταν το σύστημα εκκίνησης είναι χαλασμένο και τα μπουζί είναι καινούρια και το αυτοκίνητο έχει πάει συνεργείο.

Προτείνετε, σύνολο απαντήσεων ως ενδείξεις ώστε να ασκήσετε ένα ερώτημα στο έμπειρο και να λειτουργήσει αυτό.

Άσκηση 6
Για την καλύτερη κατανόηση του εμπείρου με τη χρήση μενού επικοινωνίας δημιουργείστε έμπειρο σύστημα το οποίο να επιλέγει κινηματογράφο βάση της περιοχής που βρίσκεται και των ειδών ταινιών τα οποία προσφέρει.
Θα μπορούν να γίνονται στο σύστημα τρία είδη ερωτήσεων

1. εύρεση κινηματογράφου ανάλογα με την περιοχή που βρίσκεται
2. εύρεση κινηματογράφου ανάλογα με τα είδη ταινιών που προβάλει
3. εύρεση των είδη ταινιών που προβάλει δίνοντας το όνομα του κινηματογράφου

Ισχύουν τα εξής:

- Τα Kops cinemas προβάλουν περιπέτειες και κοινωνικά, ενώ βρίσκονται στην συνοικία ΑΒΕΡΩΦ
- Το Lari videocenter προβάλλει ταινίες αισθησιακές, θρησκευτικές και περιπέτειες, ενώ βλίσκεται στον Αγιο Χαράλαμπο
- Το VideoZook προβάλλει ταινίες αισθησιακές και περιπέτειες, ενώ βλίσκεται στην συνοικία ΑΒΕΡΩΦ
- Το agapoulini προβάλλει αισθησιακές, ενώ βλίσκεται στα Ταμπάκικα
- Το B-romylos προβάλλει ταινίες θρησκευτικές, ενώ βλίσκεται στην συνοικία ΑΒΕΡΩΦ
- Το koukoulas SA προβάλλει ταινίες περιπέτειες, ενώ βλίσκεται στον Αγιο Χαράλαμπο
- Το tzampatzis Odeon προβάλλει αισθησιακές και περιπέτειες, ενώ βλίσκεται στα Ταμπάκικα