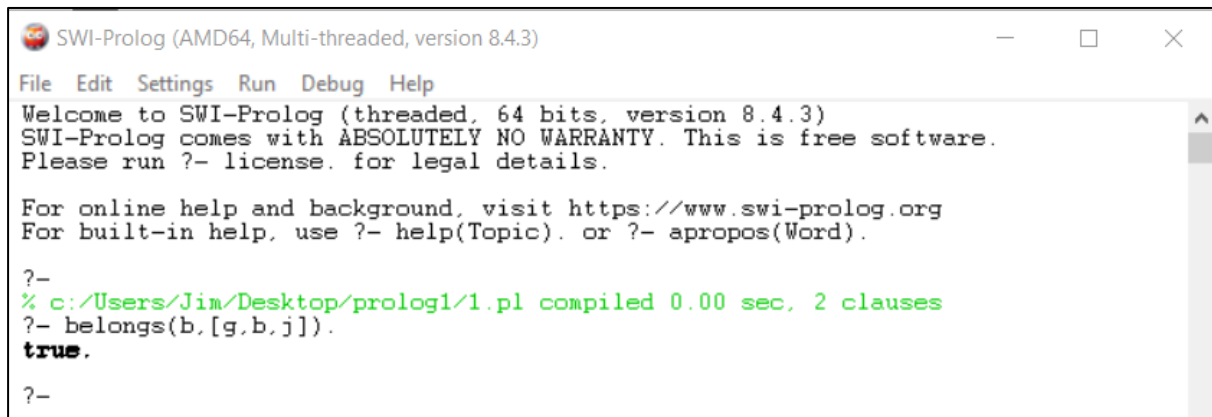


## 1. Πότε ανήκει ένα στοιχείο σε μία λίστα

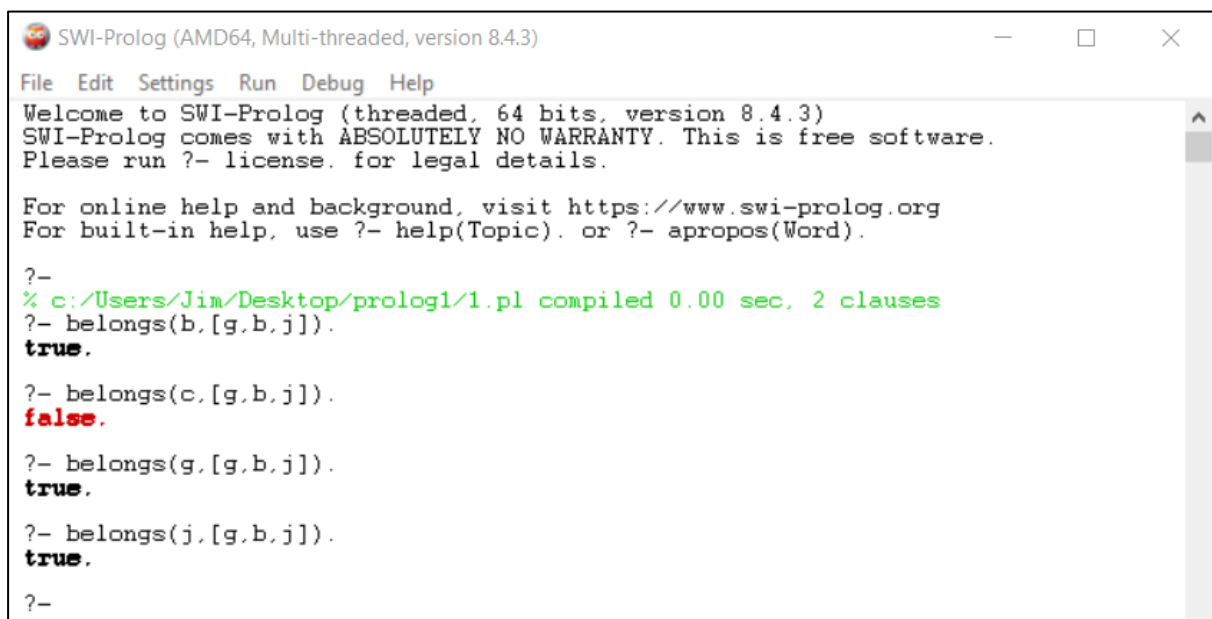
Πριν από το «,» γράφουμε τον χαρακτήρα που θέλουμε να ελέγξουμε αν ανήκει στην λίστα. Η λίστα είναι η [...] όπου εκεί μέσα γράφουμε τους χαρακτήρες. Αν ο χαρακτήρας που γράφουμε αριστερά από το «,» είναι και μέσα στην λίστα τότε εκτυπώνει true αλλιώς όταν δεν υπάρχει στην λίστα εκτυπώνει false.

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

Q: belongs(b,[g,b,j]).



```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?-  
% c:/Users/Jim/Desktop/prolog1/1.pl compiled 0.00 sec, 2 clauses  
?- belongs(b,[g,b,j]).  
true.  
?-
```



```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit https://www.swi-prolog.org  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?-  
% c:/Users/Jim/Desktop/prolog1/1.pl compiled 0.00 sec, 2 clauses  
?- belongs(b,[g,b,j]).  
true.  
?- belongs(c,[g,b,j]).  
false.  
?- belongs(g,[g,b,j]).  
true.  
?- belongs(j,[g,b,j]).  
true.  
?-
```

## 2. Γέμισμα και άδειασμα μίας λίστας

Γράφουμε τους χαρακτήρες που θέλουμε αρχικά να εκτυπώσει σταδιακά έναν-έναν (αρχικά η λίστα είναι κενή και μετά φορτώνονται ένας-ένας χαρακτήρες) ενώ μετά γράφουμε τους χαρακτήρες που θέλουμε να εκτυπώσει, αρχικά όλους μέχρι να αδειάσει η λίστα.

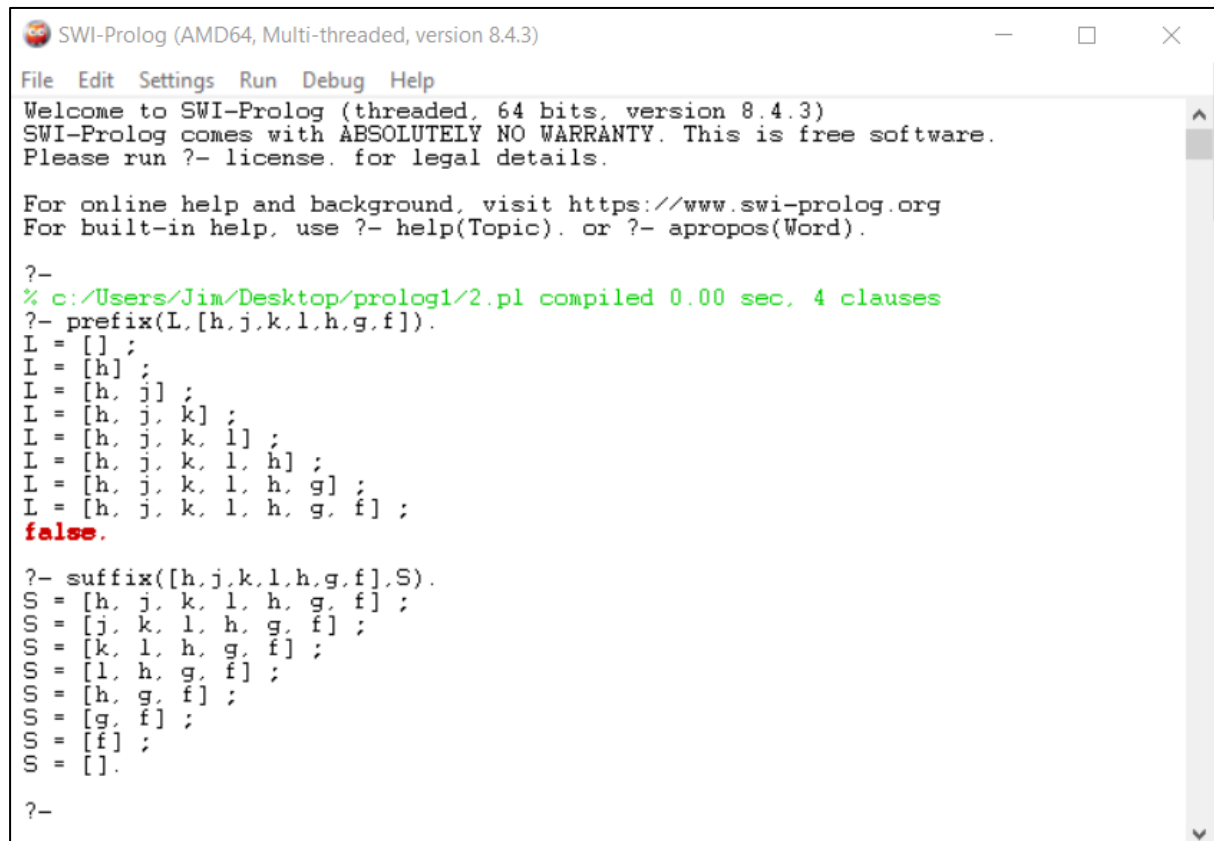
```
prefix([],_).  
prefix([H|T1],[H|T2]):-prefix(T1,T2).
```

```
suffix(S,S).
suffix([_ | T],L):-suffix(T,L).
```

Q:

```
prefix(L,[h,j,k,l,h,g,f]).
```

```
suffix([h,j,k,l,h,g,f],S).
```



```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/Jim/Desktop/prolog1/2.pl compiled 0.00 sec, 4 clauses
?- prefix(L,[h,j,k,l,h,g,f]).
L = [] ;
L = [h] ;
L = [h, j] ;
L = [h, j, k] ;
L = [h, j, k, l] ;
L = [h, j, k, l, h] ;
L = [h, j, k, l, h, g] ;
L = [h, j, k, l, h, g, f] ;
false.

?- suffix([h,j,k,l,h,g,f],S).
S = [h, j, k, l, h, g, f] ;
S = [j, k, l, h, g, f] ;
S = [k, l, h, g, f] ;
S = [l, h, g, f] ;
S = [h, g, f] ;
S = [g, f] ;
S = [f] ;
S = [].

?-
```

### 3. Ένωση 2 λιστών

Σύνδεση 2 διαφορετικών λιστών, με 3 διαφορετικά παραδείγματα.

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

Q:

```
syndese([9,7,g,a],[5,7,k],C).
```

```
syndese(L1,L2,[a,f,g]).
```

```
syndese(L3,[l,k],[h,k,l,k]).
```

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/Jim/Desktop/prolog1/3.pl compiled 0.00 sec, 2 clauses
?- syndese([9,7,g,a],[5,7,k],C).
C = [9, 7, g, a, 5, 7, k].

?- syndese(L1,L2,[a,f,g]).
L1 = [],
L2 = [a, f, g].

?- syndese(L3,[l,k],[h,k,l,k]).
L3 = [h, k].

?-
```

```
?- syndese(L3,[c,d],[a,b,c,d]).
L3 = [a, b].

?- syndese(L3,[a,b],[a,b,c,d]).
false.

?- syndese(L3,[b,c],[a,b,c,d]).
false.

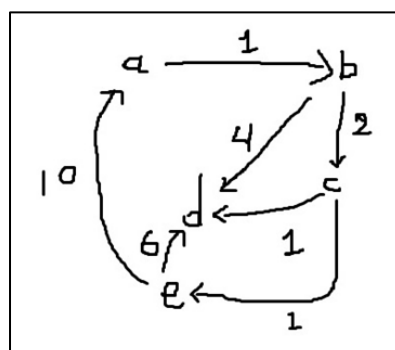
?- syndese(L3,[a,d],[a,b,c,d]).
false.

?- syndese(L3,[b,d],[a,b,c,d]).
false.

?-
```

#### 4. Άσκηση με γράφο και βάρη (Νεφρά)

Δίνεται ο ακόλουθος κατευθυντικός γράφος με βάρη που αφορά τα νεφρά ενός ανθρώπου και 5 περιοχές τους. Στις περιοχές αυτές κυκλοφορούν διάφορα υγρά. Η περιοχή d είναι η απόληξη δηλαδή η περιοχή όπου συγκεντρώνονται όλα τα υγρά και στην συνέχεια θα ακολουθήσει η διαδικασία αποβολής τους (δεν το εξετάζουμε εδώ). Δεν θα μπορούμε σε περαιτέρω ιατρικές λεπτομέρειες παρά μόνο θα πούμε ότι η κάθε περιοχή στέλνει, συγκεντρώνει υγρά διαφορετικής φύσεως τα οποία κυκλοφορούν με διάφορες ταχύτητες και μετατρέπονται σε κάθε κόμβο (κάτι όμως που δεν μας απασχολεί στον παρόν πρόβλημα).



Προκύπτει η ακόλουθη γνωσιακή βάση **Δε θα δοθεί στην εξέταση**

```
edge(a, b, 1).  
edge(b, c, 2).  
edge(c, d, 1).  
edge(b, d, 4).  
edge(c, e, 1).  
edge(e, a, 10).  
edge(e, d, 6).
```

Σκοπός μας είναι να βρούμε τρόπους κυκλοφορίας των υγρών και τις συνολικές ταχύτητες σε μη γειτονικούς κόμβους π.χ. από το a στο d. Έτσι πρέπει να γράψουμε κανόνες με τις ακόλουθες λογικές:

- Υπάρχει μονοπάτι (findapath) μεταξύ X και Y το οποίο έχει βάρος W, εάν υπάρχει ένα ακμή μεταξύ X και Y που έχει βάρος W.
- Το συνολικό βάρος μεταξύ X και Y που είναι το W, εάν μπορούμε να βρούμε μια διαδρομή μεταξύ X και Z του βάρους W1 και υπάρχει «εύρημα» μεταξύ Z και Y του βάρους W2 όπου το W είναι  $W1 + W2$ .

```
path(X, X, 0, []).  
path(X, Y, W, [Y]) :- edge(X, Y, W).  
path(X, Y, W, [Z|T]) :- edge(X, Z, W1), path(Z, Y, W2, T), W is W1 + W2.  
path(X, Y, W) :- path(X, Y, W, _).
```

Q:

path(a,d,K).

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)  
File Edit Settings Run Debug Help  
% c:/Users/Jim/Desktop/prolog1/4.pl compiled 0.00 sec, 11 clauses  
?- path(a,d,K).  
K = 5 ;  
K = 4 ;  
K = 4 ;  
K = 10 ;  
K = 19 ;  
K = 18 ;  
K = 18 ;  
K = 24 ;  
K = 33 ;  
K = 32 ;  
K = 32 ;  
K = 38 ;  
K = 47 ;  
K = 46 ;  
K = 46 ;  
K = 52 ;  
K = 61 ;  
K = 60 ;  
K = 60 ;
```

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help

?-
% c:/Users/Jim/Desktop/prolog1/4.pl compiled 0.00 sec, 11 clauses
?- path(a,c,K).
K = 3 ;
K = 3 ;
K = 17 ;
K = 17 ;
K = 31 ;
K = 31 ;
K = 45 ;
K = 45 ;
K = 59 ;
K = 59 ;
K = 73 ;
K = 73 ;
K = 87 ;
K = 87 ;
K = 101 ;
K = 101 ;
K = 115 ;
```

## 5. Άσκηση (Συνάντηση Φοιτητών)

Το παρακάτω project αφορά την παρακολούθηση μαθημάτων από φοιτητές. Υπάρχουν τα ακόλουθα γεγονότα:

- Το γεγονός `course(X)` δηλώνει τον κωδικό `X` ενός μαθήματος
- Το γεγονός `department(X,Y)` δηλώνει ότι το μάθημα με κωδικό `X` προσφέρεται από το Τμήμα `Y`
- Το γεγονός `student(X)` δηλώνει ότι ο `X` είναι φοιτητής
- Το γεγονός `enrolled(X,Y)` δηλώνει ότι ο φοιτητής `X` παρακολουθεί το μάθημα `Y`

Αναλυτικότερα:

Όπου τα γεγονότα `course(X)` δηλώνουν μάθημα με κωδικό `X`, τα γεγονότα `department(X,Y)` δηλώνουν ότι το μάθημα με κωδικό `X` προσφέρεται από το Τμήμα `Y`, τα γεγονότα `student(X)` δηλώνουν ότι ο `X` είναι φοιτητής και τέλος τα γεγονότα `enrolled(X,Y)` δηλώνουν ότι ο φοιτητής `X` παρακολουθεί το μάθημα `Y`.

Δίνονται μαθήματα με κωδικούς 312, 322, 315 και 371.

```
course(312).
course(322).
course(315).
course(371).
```

Τα μαθήματα με κωδικούς 312 και 322 προσφέρονται από Τμήμα Υπολογιστών.

```
department(312, computer_science).
department(322, computer_science).
```

Το μάθημα με κωδικό 315 προσφέρεται από το Τμήμα Μαθηματικών.

```
department(315, mathematics).
```

Το μάθημα με κωδικό 371 προσφέρεται από το Φυσικής.

```
department(371, physics).
```

Η Μαίρη, Ο Γιάννης, η Τζέιν και η Πέτρος είναι φοιτητές.

```
student(mary).  
student(jane).  
student(john).  
student(peter).
```

Η Μαίρη παρακολουθεί τα μαθήματα 322 και 312 και 315.

```
enrolled(mary, 322).  
enrolled(mary, 312).  
enrolled(mary, 315).
```

Ο Γιάννης παρακολουθεί τα μαθήματα 322 και 315.

```
enrolled(john, 322).  
enrolled(john, 315).
```

Η Τζέιν παρακολουθεί τα μαθήματα 312 και 322.

```
enrolled(jane, 312).  
enrolled(jane, 322).
```

Ο Πέτρος παρακολουθεί το μάθημα 371.

```
enrolled(peter, 371).
```

Στην συνέχεια πρέπει να γράψουμε κατηγορήμα όπου οι όλοι φοιτητές πηγαίνουν υποχρεωτικά τις παραδόσεις και ενδεχομένως να συναντιούνται κάποιοι με κάποιους. Το `meet(X,Y)` μας δίνει ποιοι από τους φοιτητές συναντούνται μεταξύ τους στις παραδόσεις.

```
meet(X, Y) :- student(X), student(Y), not(X = Y), enrolled(X, Z), enrolled(Y, Z).
```

Με ποιους συμφοιτητές της συναντιέται η Μαίρη;

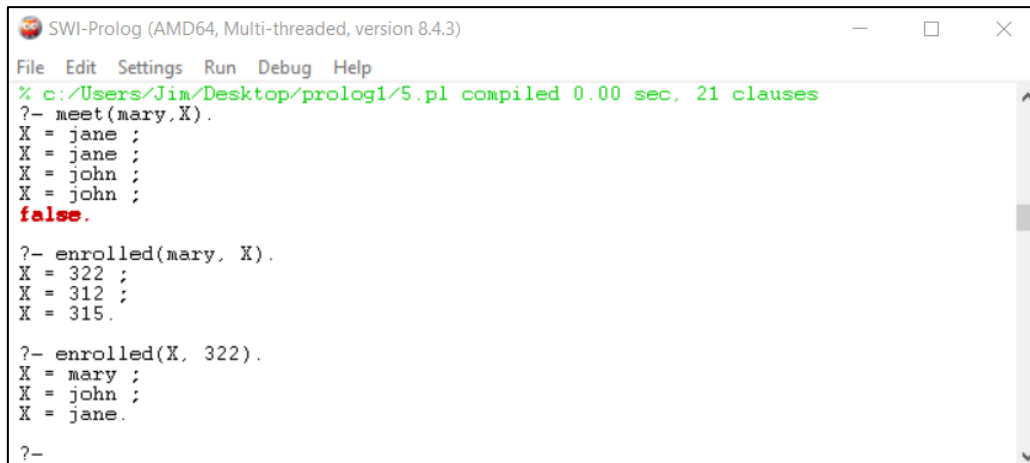
Q: `meet(mary, X)`.

Ποια μαθήματα παρακολουθεί η Μαίρη;

Q: enrolled(mary, X).

Ποιοι φοιτητές παρακολουθούν το μάθημα με κωδικό 322;

Q: enrolled(X, 322).



```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
% c:/Users/Jim/Desktop/prolog1/5.pl compiled 0.00 sec, 21 clauses
?- meet(mary,X).
X = jane ;
X = jane ;
X = john ;
X = john ;
false.

?- enrolled(mary, X).
X = 322 ;
X = 312 ;
X = 315.

?- enrolled(X, 322).
X = mary ;
X = john ;
X = jane.

?-
```

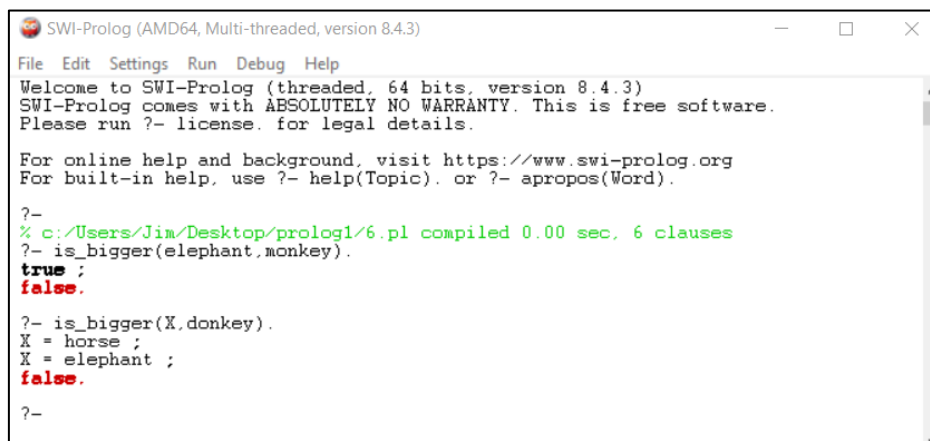
## 6. Απλή βάση γνώσης (μεγαλύτερα ζώα - σύγκριση)

bigger(elephant, horse).  
bigger(horse, donkey).  
bigger(donkey, dog).  
bigger(dog, monkey).

is\_bigger(X,Y):-bigger(X,Y).  
is\_bigger(X,Y):-bigger(X,Z), is\_bigger(Z,Y).

Τι αποτελέσματα θα εμφανίσουν τα ακόλουθα ερωτήματα;

1. is\_bigger(elephant, monkey).
2. is\_bigger(X,donkey).



```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/Jim/Desktop/prolog1/6.pl compiled 0.00 sec, 6 clauses
?- is_bigger(elephant,monkey).
true ;
false.

?- is_bigger(X,donkey).
X = horse ;
X = elephant ;
false.

?-
```

```
?- is_bigger(X,dog).  
X = donkey ;  
X = elephant ;  
X = horse ;  
false.
```

## 7. Άσκηση με είδη και χαρακτηριστικά οχημάτων

Μία εταιρεία μεταχειρισμένων αυτοκινήτων θέλει να δημιουργήσει μία βάση γνώσης για τα οχήματα που αγοράζει για να τα πουλήσει αργότερα. Όχημα είναι και το τρακτέρ και το αυτοκίνητο. Κάθε όχημα έχει χαρακτηριστικά όπως μάρκα, χιλιόμετρα, χρόνια κυκλοφορίας, χρώμα και τιμή. Τα παρακάτω δεδομένα είναι ένα μικρό απόσπασμα από μία βάση γνώσης μίας τέτοια επιχείρησης.

```
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).
```

Βρείτε:

1. Ποια οχήματα έχουν τιμή 25000 €;

vehicle(X,A,B,T,25000).

2. Εμφανίστε όλα τα datsun οχήματα

vehicle(datsun,X,A,B,G).

3. Εμφανίστε όλα τα red αυτοκίνητα

vehicle(A,B,C,red,Y).



```

SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
% c:\Users\Jim\Desktop\prolog1\7.pl compiled 0.00 sec, 7 clauses
?- vehicle(X,A,B,T,25000).
X = ford,
A = 90000,
B = 4,
T = gray ;
X = toyota,
A = 25000,
B = 2,
T = black.

?- vehicle(datsun,X,A,B,G).
X = 8000,
A = 1,
B = red,
G = 30000 ;
X = 50000,
A = 5,
B = orange,
G = 20000.

?- vehicle(A,B,C,red,Y).
A = chrysler,
B = 130000,
C = 3,
Y = 12000 ;
A = datsun,
B = 8000,
C = 1,
Y = 30000 ;
false.
?-

```

4. Εμφανίστε μόνο τα car ford

car(ford,B,C,D,T).

5. Εμφανίστε τα car ford και τα truck με χρώμα black

car(ford,B,C,D,T);truck(A,B,C,black,R).

6. Εμφανίστε τα οχήματα με τιμή μεγαλύτερη από 20000

vehicle(A,B,C,D,Price),Price>20000.

7. Εμφανίστε τα οχήματα με χιλιόμετρα λιγότερα από 50000

vehicle(A,Odometer,C,D,F),Odometer<20000.

```

SWI-Prolog (AMD64, Multi-threaded, version 8.4.3)
File Edit Settings Run Debug Help
?- car(ford,B,C,D,T).
B = 90000,
C = 4,
D = gray,
T = 25000.

?- car(ford,B,C,D,T); truck(A,B,C,black,R).
B = 90000,
C = 4,
D = gray,
T = 25000 ;
B = R, R = 25000,
C = 2,
A = toyota.

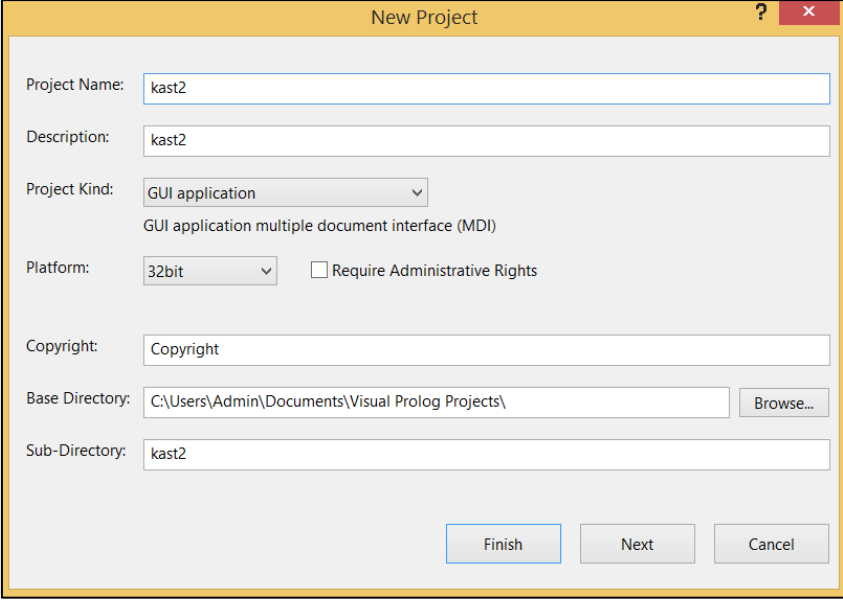
?- vehicle(A,B,C,D,Price),Price>20000.
A = ford,
B = 90000,
C = 4,
D = gray,
Price = 25000 ;
A = datsun,
B = 8000,
C = 1,
D = red,
Price = 30000 ;
A = toyota,
B = Price, Price = 25000,
C = 2,
D = black.

?- vehicle(A,Odometer,C,D,F),Odometer<20000.
A = datsun,
Odometer = 8000,
C = 1,
D = red,
F = 30000 ;
false.
?-

```

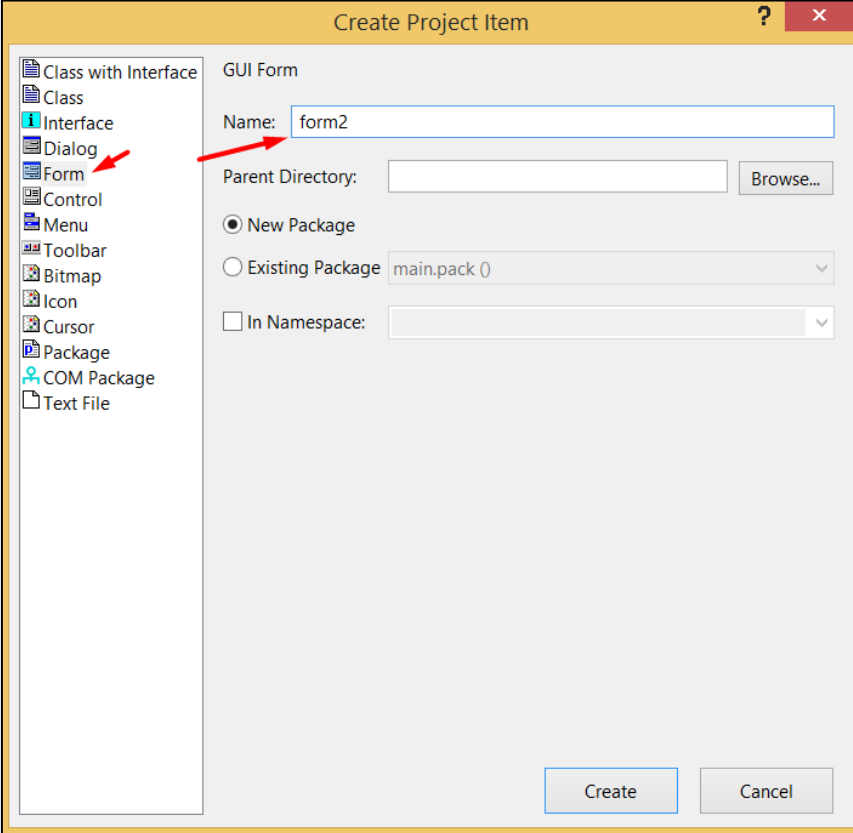
## 8. Δημιουργία φόρμας στην Visual Prolog

Project > New Project



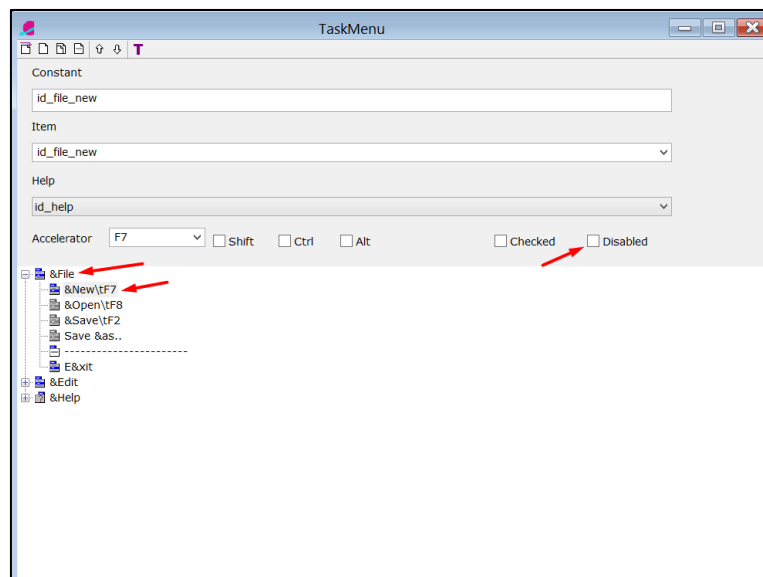
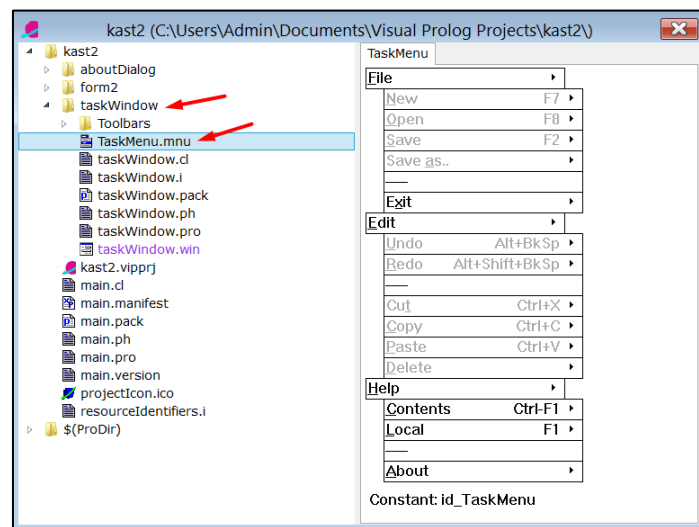
Με το που πατήσουμε Finish περιμένουμε να γίνει το compiling (θα αργήσει λίγο)

File > New in New Package

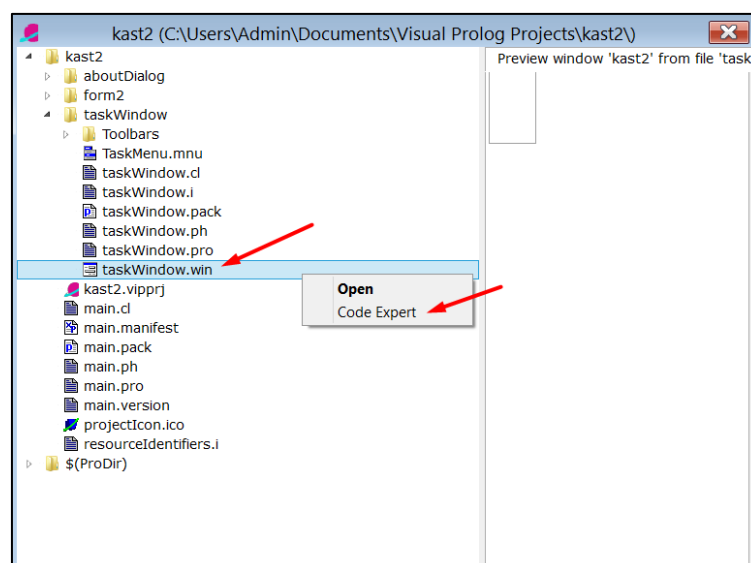


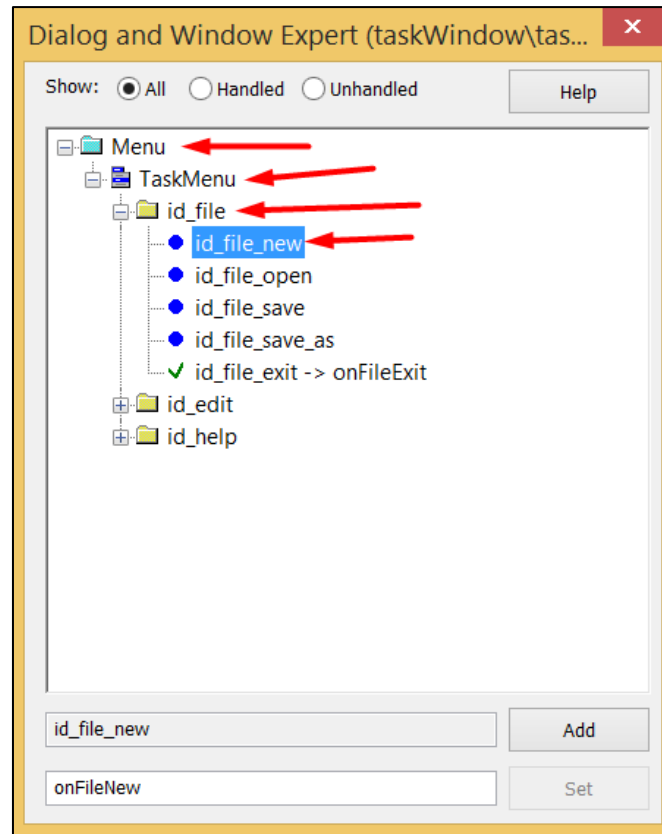
Κλείνουμε το παράθυρο που εμφανίζεται με τον κώδικα, θα το χρησιμοποιήσουμε αργότερα.

Διπλό κλικ:



Πατάμε X και μετά Save.





Στο παράθυρο που εμφανίζεται στο δεύτερο «onFileNew» γράφουμε τον παρακάτω κώδικα.

```

taskWindow.pro (taskWindow\)

onDestroy(_).

class predicates
  onHelpAbout : window::menuItemListener.
clauses
  onHelpAbout(TaskWin, _MenuTag) :-
    _AboutDialog = aboutDialog::display(TaskWin).

predicates
  onFileExit : window::menuItemListener.
clauses
  onFileExit(_, _MenuTag) :-
    close().

predicates
  onSizeChanged : window::sizeListener.
clauses
  onSizeChanged(_) :-
    vpiToolbar::resize(getVPIWindow()).

predicates
  onFileNew : window::menuItemListener.
clauses
  onFileNew(_Source, _MenuTag) :- _=form2::display(This).

% This code is maintained automatically, do not update it manually.
predicates
  generatedInitialize : ().
clauses
  generatedInitialize() :-

```

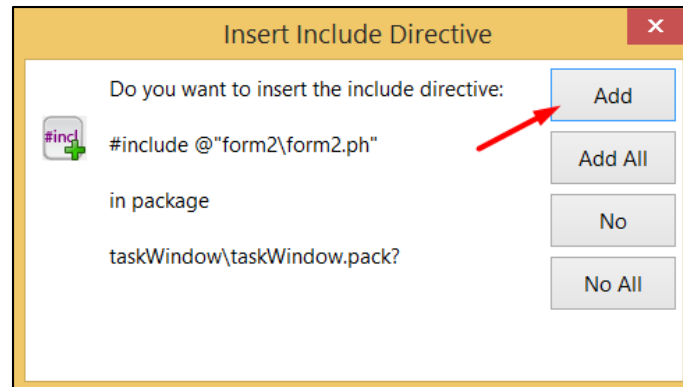
Πατάμε X και Ναι για την αποθήκευσή του.

Θα πρέπει να εμφανιστεί ένα παράθυρο, στην περίπτωση που δεν εμφανιστεί τότε κάνουμε τα εξής:

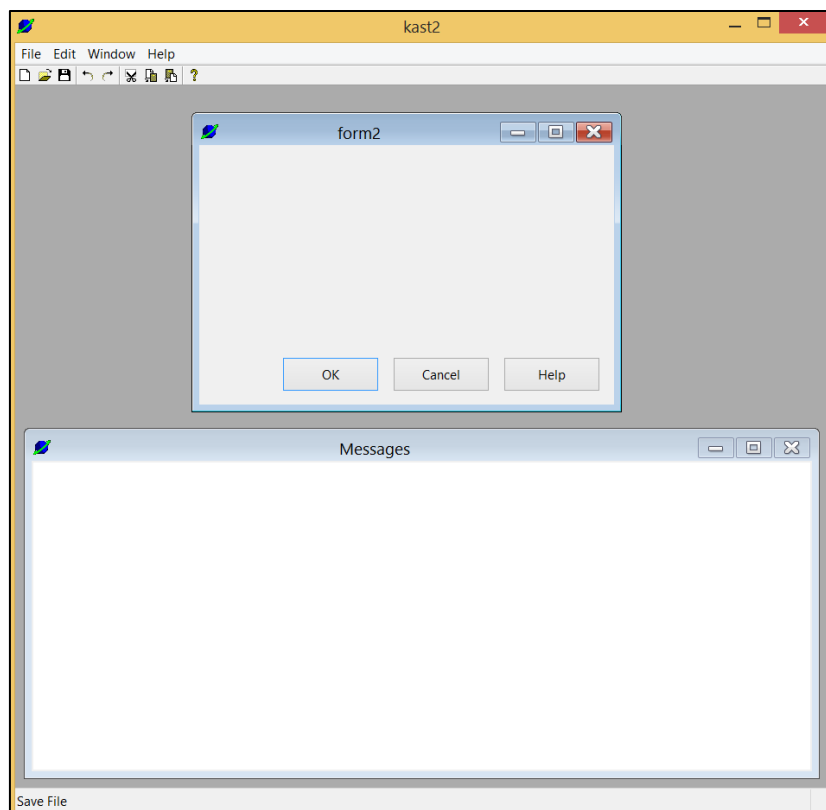
Build > Execute

Πατάμε Cancel στο License

Ξεκινάει ξανά το Compiling και μας εμφανίζει ένα παράθυρο για την προσθήκη του αρχείου με την φόρμα:

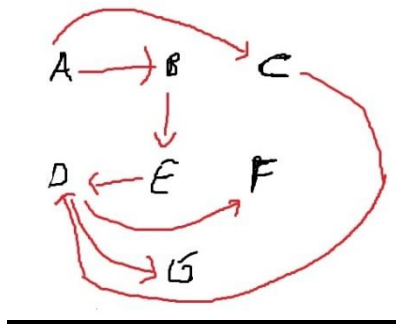


Μετά το Compiling θα εμφανιστεί ένα παράθυρο, σε αυτό πατάμε File > New και εμφανίζεται η φόρμα που δημιουργήσαμε.



## Άσκηση 1

Δίνεται ο παρακάτω κατευθυντικός γράφος χωρίς βάρη



Προκύπτει η ακόλουθη γνωσιακή βάση :

edge(a,b).

edge(b,e).

edge(a,c).

edge(c,d).

edge(e,d).

edge(d,f).

edge(d,g).

Στην συνέχεια γράψτε έναν απλό κανόνα όπου αν το X είναι συνδεδεμένο στο Z και το Z είναι συνδεδεμένο στο Y, τότε υπάρχει μια διαδρομή μεταξύ X και Y.

**Λύση:**

path(X,Y) :- edge(X,Y).

path(X,Y) :- edge(X,Z), path(Z,Y).

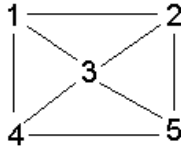
Στην συνέχεια βρείτε:

A. τις διαδρομές από το e προς έναν άλλο κόμβο path(e,A).

B. τις διαδρομές προς το g. path(A,g).

## Άσκηση 2

Δίνεται ο παρακάτω μη κατευθυντικός γράφος.



Με βάση τον γράφο αυτό προκύπτει η παρακάτω γνωσιακή βάση.

edge(1,2).

edge(1,4).

edge(1,3).

edge(2,3).

edge(2,5).

edge(3,4).

edge(3,5).

edge(4,5).

Κατασκευάστε κανόνες με βάση την παρακάτω λογική:

- Μια διαδρομή από το A στο B υφίσταται εάν τα A και B είναι συνδεδεμένα.
- Μια διαδρομή από το A στο B υφίσταται υπό την προϋπόθεση ότι το A είναι συνδεδεμένο σε έναν κόμβο C διαφορετικό από το B που δεν βρίσκεται στο τμήμα της διαδρομής που επισκεφτήκαμε προηγουμένως και συνεχίζει να βρίσκει μια διαδρομή από το C στο B.
- Λάβετε υπόψη ότι η αποφυγή επαναλαμβανόμενων κόμβων διασφαλίζει ότι το πρόγραμμα δεν θα κυκλοφορεί ατελείωτα.

## Λύση:

connected(X,Y) :- edge(X,Y).

connected(X,Y) :- edge(Y,X).

path(A,B,Path) :-travel(A,B,[A],Q), reverse(Q,Path).

travel(A,B,P,[B|P]) :- connected(A,B).

travel(A,B,Visited,Path) :-connected(A,C), C \==

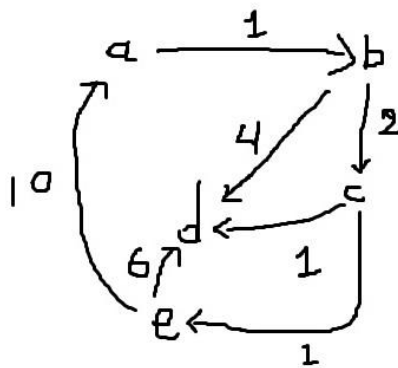
B,\+member(C,Visited),

travel(C,B,[C|Visited],Path).

**Δοκιμάστε το ?- path(1,5,P).**

### Άσκηση 3

Δίνεται ο ακόλουθος κατευθυντικός γράφος με βάρη που αφορά τα νεφρά ενός ανθρώπου και 5 περιοχές τους. Στις περιοχές αυτές κυκλοφορούν διάφορα υγρά. Η περιοχή d είναι η απόληξη δηλαδή η περιοχή όπου συγκεντρώνονται όλα τα υγρά και στην συνέχεια θα ακολουθήσει η διαδικασία αποβολής τους (δεν το εξετάζουμε εδώ). Δεν θα μπορούμε σε περαιτέρω ιατρικές λεπτομέρειες παρά μόνο θα πούμε ότι η κάθε περιοχή στέλνει, συγκεντρώνει υγρά διαφορετικής φύσεως τα οποία κυκλοφορούν με διάφορες ταχύτητες και μετατρέπονται σε κάθε κόμβο (κάτι όμως που δεν μας απασχολεί στον παρόν πρόβλημα)



Προκύπτει η ακόλουθη γνωσιακή βάση

edge(a, b, 1).  
edge(b, c, 2).  
edge(c, d, 1).  
edge(b, d, 4).  
edge(c, e, 1).  
edge(e, a, 10).  
edge(e, d, 6).

Σκοπός μας είναι να βρούμε τρόπους κυκλοφορίας των υγρών και τις συνολικές ταχύτητες σε μη γειτονικούς κόμβους π.χ. από το a στο d. Έτσι πρέπει να γράψουμε κανόνες με τις ακόλουθες λογικές:

- Υπάρχει μονοπάτι (findpath) μεταξύ X και Y το οποίο έχει βάρος W, εάν υπάρχει ένα ακμή μεταξύ X και Y που έχει βάρος W.
- Το συνολικό βάρος μεταξύ X και Y που είναι το W, εάν μπορούμε να βρούμε μια διαδρομή μεταξύ X και Z του βάρους W1 και υπάρχει «εύρημα» μεταξύ Z και Y του βάρους W2 όπου το W είναι W1 + W2.



### Λύση:

path(X, X, 0, []).

path(X, Y, W, [Y]) :- edge(X, Y, W).

path(X, Y, W, [Z|T]) :- edge(X, Z, W1), path(Z, Y, W2, T), W is W1 + W2.

path(X, Y, W) :- path(X, Y, W, \_).

Τρέξτε το πρόγραμμα με την εντολή

?- path(a,d,K).

Εντοπίστε αδυναμίες.....

### Άσκηση 4

Δημιουργήστε ένα απλό GPS σύστημα με διαδρομές από mavrohorι έως mikroakastro' με επιλογές κατεύθυνσης deksia, aristera, eythia.

### Λύση:

directTrain('mavrohorι','gkiolē','deksia').

directTrain('gkiolē','ampelokipoi','aristera').

directTrain('ampelokipoi','militsa','eythia').

directTrain('militsa','kostarazi','deksia').

directTrain('kostarazi','vogatsiko','eythia').

directTrain('vogatsiko','neapoli','aristera').

directTrain('neapoli','mikroakastro','deksia').

travelBetween(X,Y,M):-directTrain(X,Y,M),write(X),write('-->'),write(M),write('-->'),write(Y).

travelBetween(X,Y,A):-directTrain(X,Z,M),write(X),write('-->'),write(M),write('-->'),write(Z),write('-->'),travelBetween(Z,Y,A).

```

path(a,b).
path(b,e).
path(a,c).
path(c,d).
path(e,d).
path(d,f).
path(d,g).

path(X,Y):- path(Y,X).
connected(X,Y) :- path(X,Y) ; path(Y,X).
connected(X,Y) :- path(X,Y).
connected(X,Y) :- path(Y,X).
connected(X,Y) :- path(X,Y).
connected(X,Y) :- path(X,Z),connected(Z,Y).

```

```

edge(1,2).
edge(1,4).
edge(1,3).
edge(2,3).
edge(2,5).
edge(3,4).
edge(3,5).
edge(4,5).

connected(X,Y):-edge(X,Y) ; edge(Y,X).

path(A,B,Path):-travel(A,B,[A],Q),reverse(Q,Path).

travel(A,B,P,[B|P]):-connected(A,B).

travel(A,B,Visited,Path):-
connected(A,C),C\==B,\+member(C,Visited),travel(C,B,[C|Visited],Path).

```

```

edge(a,b,1).
edge(b,e,2).
edge(a,c,3).
edge(c,d,2).
edge(e,d,3).
edge(d,f,2).
edge(d,g,3).

```

```
findapath(X,Y,W,[X,Y],_-edge(X,Y,W).
```

```
findapath(X,Y,W,[X|P],V):-
```

```
\+member(X,V),edge(X,Z,W1),findapath(Z,Y,W2,P,[X|V]),W is  
W1+W2.
```

### Άσκηση 1

Ένας ηλεκτρονικός υπολογιστής έχει ένα πλήθος επικίνδυνων διασκορπισμένων αρχείων στον δίσκο του, που ολόκληρος είναι ένας ενιαίος φάκελος. Θεωρώντας ότι τα ονόματα αυτών των αρχείων είναι ίδια (trojan) και όλα μαζί τα αρχεία αποτελούν λίστα ονομάτων, να δημιουργήσετε πρόγραμμα σε prolog που να αφαιρεί από τη λίστα των ονομάτων, τα αρχεία με όνομα trojan.

Δοκιμάστε το πρόγραμμά σας με τη λίστα [f1, trojan, d, trojan, f2]).

### Άσκηση 2

Μία ηλεκτρική συσκευή αποτελείται από 5 διαφορετικά εξαρτήματα  $p_1, p_2, p_3, p_4, p_5$  τα οποία βρίσκονται στην κεντρική πλακέτα της και λειτουργούν το καθένα ξεχωριστά με βάση συγκεκριμένο λογισμικό  $s_1, s_2 \dots s_5$ . Το κάθε εξάρτημα δηλαδή έχει το δικό του λογισμικό το οποίο όμως περιοδικά χρειάζεται αντικατάσταση με αναβάθμισή του σε νεότερο  $n_i$  με  $i=1,2..5$ . Την αντικατάσταση αναλαμβάνει διαγνωστικό σύστημα που ελέγχει τα λογισμικά και αντικαθιστά το παρωχημένο  $s_i$  με νεότερο  $n_i$  και αυτό γίνεται με κατάλληλο πρόγραμμα της prolog όταν συνδέσουμε τη συσκευή στο διαγνωστικό σύστημα.

Γράψτε το πρόγραμμα του διαγνωστικού συστήματος που υλοποιεί την αντικατάσταση του λογισμικού, συγκεκριμένου εξαρτήματος, με το αναβαθμισμένο νεότερο λογισμικό. Κάντε τις σχετικές δοκιμές.

### Άσκηση 3

Υποθέτουμε ότι ο covid-19 αποτελείται από 5 διαφορετικά είδη μικροβίων, των οποίων ο πληθυσμός του κάθε είδους καταχωρείται ημερήσια σε κατάλληλη λίστα, με συγκεκριμένη καθορισμένη σειρά. Η συνάρτηση αξιολόγησης της διάγνωσης του ασθενούς είναι της μορφής  $2*\alpha_i - 3*\beta_i$ , όπου  $\alpha_i$  ο πληθυσμός του  $i$  είδους μικροβίου την πρώτη ημέρα και  $\beta_i$  ο αντίστοιχος πληθυσμός την δεύτερη ημέρα. Οι γιατροί βλέπουν τον πίνακα μικροβίων πώς διαμορφώνεται μετά τη δεύτερη ημέρα και αν ο πληθυσμός οποιουδήποτε από τα 5 μικρόβια είναι θετικός αποφαινόνται ότι εξεταζόμενος άνθρωπος είναι θετικός στον

covid-19. Στην περίπτωση όπου όλοι οι αριθμοί είναι αρνητικοί ή μηδέν τότε αποφαίνονται ότι είναι αρνητικός.

Διαμορφώστε κατάλληλο πρόγραμμα σε prolog που να κάνει τη σχετική διάγνωση. Δοκιμάστε το πρόγραμμά σας καταχωρώντας σε δυο λίστες διάφορους πληθυσμούς μικροβίων και ελέγξτε τα αποτελέσματα που προκύπτουν από τη συνάρτηση αξιολόγησης σε τρίτη λίστα.

#### Άσκηση 4

Όσοι ασχοληθείτε με την prolog σε επαγγελματικό επίπεδο θα συναντήσετε το παραθυρικό περιβάλλον της prolog όπου ένα από τα πιο χρήσιμα προγράμματα είναι η δημιουργία back-up αρχείων. Ο Στρατός η Αστυνομία και οι Δικαστικές αρχές έχουν προγράμματα που δημιουργούν κλώνους (δηλαδή ίδια αρχεία). Στα πλαίσια αυτού του μαθήματος ζητείται να δημιουργήσετε στο περιβάλλον SWI-PROLOG ένα project όπου δίπλα σε κάθε στοιχείο μίας λίστας να εμφανίζεται ένα όμοιο του, να δημιουργεί δηλαδή δεύτερη λίστα στην οποία θα εμφανίζονται δύο φορές τα στοιχεία της πρώτης.

#### Άσκηση 5

Η prolog δίνει την δυνατότητα να αντιστοιχίζονται κάποιες μεταβλητές ή κάποιες σταθερές με κάποιες άλλες. Μία πολύ καλή εφαρμογή θα μπορούσε να ήταν ένα λεξικό όρων – μεταφραστικό όπου μία ελληνική λέξη αντιστοιχίζεται σε μία αγγλική και αντίστροφα. Αυτό μπορεί να αποτελεί τη στοιχειώδη γνωσιακή βάση του συστήματος.

Να αναπτύξετε ένα απλό πρόγραμμα σε prolog στο οποίο να δίνεται σειρά λέξεων στα ελληνικά και το πρόγραμμα να αποδίδει τη σειρά των λέξεων στα αγγλικά και αντίστροφα.

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

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

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

```
change(E,NE,L,NL):-  
append(A,[E | B],L),append(A,[NE | B],NL).
```

```
addlist([],[],[]).  
addlist([H1 | T1],[H2 | T2],[H3 | T3]):-H3 is 2*H1-  
3*H2,addlist(T1,T2,T3).
```

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

```
a([],[]).  
a([H | T],[H1 | T1]):-b(H,H1),a(T | T1).  
a(L1,L2):-a(L2,L1).  
b('house','spiti').  
b('is','einai').
```

**5 ΜΑΙΟΥ 2020**

**ΑΣΚΗΣΕΙΣ ΜΕ MATCHING, ΛΙΣΤΕΣ, APPEND, REVERSE, LENGTH**

1. Δημιουργείστε μία δομή δεδομένων (όρισμα μέσα σε κατηγορημα) μέσα σε ένα fact - κατηγορημα όπου θα εμφανίζεται η ημερομηνία γέννησης ενός ανθρώπου.
2. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε μία γραμμή.
3. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε πολλές γραμμές.
4. Κατόπιν βρείτε αυτούς που γεννήθηκαν μία συγκεκριμένη χρονολογία.
5. Κατασκευάστε υπό μορφή λίστας ένα fact – κατηγορημα στο οποίο θα αποτυπώνονται ο πατέρας και οι τέσσερις του γιοι.
6. Δημιουργείστε ένα ερώτημα όπου θα γίνεται matching του πατέρα και των τεσσάρων γιών του που να εμφανίζεται η κεφαλή και η ουρά της λίστας. Να σημειωθεί ότι η συγκεκριμένη εμφάνιση είναι ΑΝΑΔΡΟΜΗ γιατί;
7. Δημιουργήστε μία ερώτηση σε prolog όπου θα εξετάσετε αν κάποιο στοιχείο είναι μέλος της λίστας.
8. Δημιουργήστε μία ερώτηση σε prolog όπου θα αποτυπώνονται όλα τα στοιχεία μίας λίστας .
9. Να ορισθεί αναδρομικά το member δηλαδή:
  - Ο πρώτος κανόνας (true) θα είναι ο κανόνας τερματισμού δηλαδή ότι αληθεύει το member όταν το πρώτο όρισμα είναι το X και το δεύτερο όρισμα είναι μία λίστα με κεφαλή X και τα υπόλοιπα στοιχεία οποιαδήποτε. (αληθεύει το member αν το X είναι το πρώτο στοιχείο της λίστας.

- Ο δεύτερος κανόνας θα είναι ο αναδρομικός (false) όπου αν το στοιχείο X ανήκει στην λίστα L τότε το member αληθεύει . ΑΝΑΔΡΟΜΗ.

10. Δημιουργήστε ένα fact – append όπου θα ενώνονται οι λίστες L1, L2 και θα σχηματίζεται η L.

11. Δημιουργήστε ένα fact-append στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την πρόσθεση μίας λίστας σε μία άλλη από αριστερά.

12. Δημιουργήστε ένα fact-append στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την συνένωση δύο λιστών με όλους τους δυνατούς τρόπους.

13. Να ορισθεί αναδρομικά το append δηλαδή:

- Θα κατασκευάσετε έναν κανόνα τερματισμού όπου θα συνενώνουμε μία κενή λίστα με μία άλλη L και το αποτέλεσμα θα είναι η L.
- Θα κατασκευάσετε ένα αναδρομικό κανόνα όπου θα συνενώνεται μία λίστα L1 με την L2 και θα προκύπτει η λίστα L3 με κεφαλή το X και ουρά μία λίστα L.

14. Δημιουργήστε ένα ερώτημα στο οποίο θα γίνεται η καταμέτρηση των στοιχείων της λίστας.

15. Δημιουργήστε ένα ερώτημα με το οποίο θα γίνεται η αντιστροφή μίας λίστας.



1) born(panos,date(23,8,1990)).  
born(maria,date(24,8,1990)).

2) born(panos,X).

3) born(maria,date(X,A,B)).

4) born(X,date(\_,\_,1990)).

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

6) children(a,[A|B]).

7) member(X,[X|\_]).  
member(X,[\_|L]):-member(X,L).

8) member(X,[1,2,3,4,5]).

9) 8,9

10) append([1,2,3],[4,5,6],L).

11) append(L,[2,3],[2,3,4]).

12) append(L1,L2,[3,6,8,9]).

13) append([],L,L).  
:-append(L1,L2,L3)



ΜΑΘΗΜΑ : ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ &  
ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΡΓΑΣΤΗΡΙΟ  
6 – 2 - 2017



Διδάσκοντες: Ιωάννης Σινάτκας, Παναγιώτης Μπάτος

Ονοματεπώνυμο φοιτητή:.....ΑΜ:.....

**Θέματα**

**ΘΕΜΑ 1<sup>ο</sup> (μονάδες 10)**

Γράψτε κώδικα σε γλώσσα prolog για μία εφαρμογή, με βάση τις ακόλουθες προδιαγραφές.

Προδιαγραφές

Η επιχείρηση «ΠΛΗΡΟΦΟΡΙΚΗ Ο.Ε.» δέχεται καθημερινά πελάτες που έχουν προβλήματα στον υπολογιστή τους. Επιθυμεί λοιπόν να δημιουργήσει ένα έμπειρο σύστημα το οποίο ανάλογα με αυτό που παρουσιάζει ένας υπολογιστής θα βρίσκει την βλάβη του. Ο παρακάτω πίνακας αποτελεί ένα μικρό απόσπασμα του συστήματος αυτού.

Παρουσιάζει	Βλάβη
Παραμορφωμένη Οθόνη ή οθόνη που τρέμει	Κάρτα γραφικών
Μαύρη οθόνη	Καμένη οθόνη
Ξαφνική επανεκκίνηση ή Κλείνει ο υπολογιστής	Ιός
Κλείνει ο υπολογιστής	Υπερθέρμανση
Κολλάει ο υπολογιστής	Σκληρός

Ο πίνακας αυτός θα χρησιμοποιηθεί για την κατασκευή εκφράσεων του τύπου:  
vlavi(X,skliros):- parousiazei(X,kollaei).

Την Δευτέρα 6-2-2017 ήρθαν στην επιχείρηση 5 πελάτες που ανέφεραν τι παρουσιάζει ο υπολογιστής τους. Ο τεχνικός κατέγραψε στις στήλες «Υπολογιστής» και «Παρουσιάζει» αυτά που ο κάθε πελάτης του ανέφερε όπως παρακάτω:

Υπολογιστής	Παρουσιάζει
Pc1	Παραμορφωμένη Οθόνη
Pc2	Μαύρη οθόνη
Pc3	Ξαφνική επανεκκίνηση
Pc4	Κλείνει ο υπολογιστής
Pc5	Κολλάει ο υπολογιστής

Ο πίνακας αυτός θα χρησιμοποιηθεί για την κατασκευή γεγονότων του τύπου:  
parousiazei(pc4,kleinei).

Στο σύστημα που δημιουργήσατε γράψτε τον κώδικα για 2 ερωτήματα της επιλογής σας και στην συνέχεια γράψτε απλά την έξοδο από το κάθε ερώτημα.

### ΘΕΜΑ 2<sup>ο</sup> (μονάδες 10)

Στο σχήμα, παρουσιάζεται ένα γενεαλογικό δένδρο. Όπως κάθε γενεαλογικό δένδρο, αποτελείται από κόμβους, καθένας από τους οποίους φέρει το όνομα ενός ατόμου. Κάθε χαμηλότερο επίπεδο δηλώνει τους απογόνους της προηγούμενης γενιάς, δηλαδή της γενιάς που βρίσκεται στο αμέσως υψηλότερο επίπεδο. Οι κόμβοι που αναπαριστούν συζύγους βρίσκονται στο ίδιο επίπεδο και ενώνονται με οριζόντιες γραμμές, έτσι ώστε να σχηματίζουν ζεύγος. Τα παιδιά ενός ζεύγους δηλώνονται με κόμβους που βρίσκονται στο αμέσως χαμηλότερο επίπεδο και συνδέονται με το ζεύγος κόμβων των γονιών του με κάθετες γραμμές.



Αντρέας  
Μαρία  
Δημήτρης  
Ουρανία  
Θανάσης

Ιωάννα  
Γλυκερία  
Βασιλική  
Ελένη  
Φώτης

Το παραπάνω γενεαλογικό δένδρο κωδικοποιεί τα ακόλουθα γεγονότα:

- Ο Αντρέας είναι άνδρας.
- Η Μαρία είναι γυναίκα.
- Ο Αντρέας νυμφεύτηκε την Μαρία.
- Ο Αντρέας και η Μαρία απέκτησαν δυο τέκνα τον αρσενικό Θανάση και την θηλυκή Ουρανία.
- Η Ουρανία παντρεύτηκε τον Δημήτρη
- Ο Θανάσης νυμφεύθηκε την Ιωάννα.
- Ο Φώτης είναι άνδρας
- Η Ελένη, η Γλυκερία και η Βασιλική είναι γυναίκες.

Για να εισαχθούν τα στοιχεία αυτά σε εντολές προγράμματος Prolog θα χρησιμοποιηθούν γεγονότα όπως **άνδρας** (male) με όρισμα το όνομα κατάλληλου προσώπου. Με τον ίδιο τρόπο το γεγονός **γυναίκα**(female). Στη συνέχεια να σχηματισθεί γεγονός **παντρεμένοι**(married) με δυο κατάλληλα ορίσματα, ένα για

κάθε σύζυγο. Το γεγονός αυτό να αληθεύει με όποια σειρά και να δοθούν τα ορίσματά του.

Στη συνέχεια να ορισθούν τα γεγονότα που αφορούν στο ποιος είναι **γονέας**(parent) ποιου με δυο επίσης ορίσματα. Το πρώτο όρισμα δηλώνει ποιος είναι ο γονέας και το δεύτερο ποιος ή ποια το τέκνο. Αυτό να ισχύει, ο γονέας δηλαδή, τόσο για τον πατέρα όσο και την μητέρα.

Στη συνέχεια να ορισθούν οι έννοιες **πατέρας**(father), **μητέρα**(mother), **παππούς**(grandfather), **γιαγιά**(grandmother) και αδέρφια ως τέκνα του ίδιου γονέα χωρίς διάκριση φύλου **αδέρφια**(siblings)

Οι εκφράσεις αυτές έχουν ως ακολούθως:

- ο Ο X είναι πατέρας του/της Y, αν ο X είναι άνδρας και επιπλέον γονέας του/της Y.
- ο Η X είναι μητέρα του/της Y, αν η X είναι γυναίκα και επιπλέον γονέας του/της Y.
- ο Ο X είναι παππούς του/της Y αν ο X είναι άνδρας, είναι γονέας του/της Z και με τη σειρά του ο/η Z είναι γονέας του/της Y.
- ο Η X είναι γιαγιά του/της Y, αν η Y είναι γυναίκα, είναι γονέας του/της Z και με τη σειρά του ο/η Z είναι γονέας του/της Y.
- ο Ο/Η X είναι παιδί του Y, αν ο Y είναι γονέας του X.
- ο Οι X και Y είναι αδέρφια (χωρίς να γίνεται προσδιορισμός φύλου) αν έχουν ένα κοινό πρόγονο Z, αν δηλαδή ο/η Z είναι γονέας του/της X, ταυτόχρονα ο/η Z είναι γονέας του/της Y.

**Απαντήστε στα 1 από τα 2 θέματα**

**Καλή επιτυχία !!!**

**%males:**

**male(αντρεας).**  
**male(δημήτρης).**  
**male(φώτης).**  
**male(θανάσης).**

**%females:**

**female(ιωάννα).**  
**female(γλυκερία).**  
**female(βασιλική).**  
**female(ελένη).**  
**female(ουρανία).**  
**female(μαρία).**

**%married:**

**married(αντρεας, μαρία).**  
**married(μαρία, αντρεας).**  
**married(δημήτρης, ουρανία).**  
**married(ουρανία, δημήτρης).**  
**married(θανάσης, ιωάννα).**  
**married(ιωάννα, θανάσης).**

**%parents:**

**parent(αντρεας, ουρανία).**  
**parent(αντρεας, θανάσης).**  
**parent(μαρία, ουρανία).**  
**parent(μαρία, θανάσης).**  
**parent(ουρανία, βασιλική).**  
**parent(ουρανία, γλυκερία).**  
**parent(δημήτρης, γλυκερία).**  
**parent(δημήτρης, βασιλική).**  
**parent(θανάσης, ελένη).**  
**parent(θανάσης, φώτης).**  
**parent(ιωάννα, ελένη).**  
**parent(ιωάννα, φώτης).**

**%fathers and mothers:**

**father(Fa,Ch) :- parent(Fa,Ch), male(Fa).**  
**mother(Mo,Ch) :- parent(Mo,Ch), female(Mo).**

**%grandfathers and grandmothers:**

**grandfather(GFa,GCh) :- parent(GFa,Pa), parent(Pa,GCh), male(GFa).**  
**grandmother(GMo,GCh) :- parent(GMo,Pa), parent(Pa,GCh), female(GMo).**

**%siblings:**

**siblings(Ch1, Ch2) :- father(Fa, Ch1), father(Fa, Ch2), mother(Ma,Ch1),**  
**mother(Ma,Ch2), Ch1 \= Ch2.**

**%cousin:**

**cousin(P1,P2) :- parent(Pp1,P1), parent(Pp2,P2), siblings(Pp1, Pp2), P1 \= P2.**

man(andreas).  
 man(thanasis).  
 man(dimitris).  
 man(fwtis).  
 woman(marry).  
 woman(ourania).  
 woman(iwanna).  
 woman(hellen).  
 woman(vasiliki).  
 woman(glikeria).  
 marrie(andreas,marry).  
 marrie(dimitris,ourania).  
 marrie(thanasis,iwanna).

married(X,Y) :- marrie(X,Y) ; marrie(Y,X).

parent(andreas,ourania).  
 parent(marry,ourania).  
 parent(andreas,thanasis).  
 parent(marry,thanasis).  
 parent(dimitris,glikeria).  
 parent(dimitris,vasiliki).  
 parent(ourania,glikeria).  
 parent(ourania,vasiliki).  
 parent(thanasis,fwtis).  
 parent(thanasis,hellen).  
 parent(iwanna,fwtis).  
 parent(iwanna,hellen).

father(X,Y):-parent(X,Y),man(X).  
 mother(X,Y):-parent(X,Y),woman(X).  
 granpa(X,Y):- parent(Z,X),parent(Y,Z),man(Y).  
 granma(X,Y):- parent(Z,X),parent(Y,Z),woman(Y).  
 siblings(X,Y):- parent(Z,X),parent(Z,Y).  
 uncle(X,Y):-man(Y),parent(Z,X),siblings(Z,Y).  
 aunt(X,Y):-woman(Y),parent(X,Z),siblings(Z,Y).  
 nephew(X,Y):-man(Y),siblings(X,Z),parent(Z,Y).  
 neice(X,Y):-woman(Y),siblings(X,Z),parent(Z,Y).  
 cousin(X,Y):-parent(Z,X),siblings(Z,K),parent(K,Y).



**ΘΕΜΑ 1<sup>ο</sup> (μονάδες 6)**

Δίνεται ο ακόλουθος κατευθυντικός γράφος με βάρη που αφορά την καρδιά ενός ανθρώπου και 5 περιοχές της. Στις περιοχές αυτές κυκλοφορεί αίμα. Η περιοχή d είναι η απόληξη δηλαδή η περιοχή όπου συγκεντρώνονται όλο το αίμα.

Προκύπτει η ακόλουθη γνωσιακή βάση

edge(a, b, 1).

edge(b, c, 2).

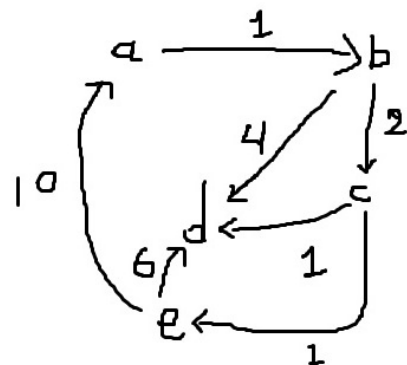
edge(c, d, 1).

edge(b, d, 4).

edge(c, e, 1).

edge(e, a, 10).

edge(e, d, 6).



Σκοπός μας είναι να βρούμε τις συνολικές ταχύτητες σε μη γειτονικούς κόμβους π.χ. από το a στο d. Έτσι πρέπει να γράψουμε κανόνες με τις ακόλουθες λογικές:

- Υπάρχει μονοπάτι (findapath) μεταξύ X και Y το οποίο έχει βάρος W, εάν υπάρχει ένα ακμή μεταξύ X και Y που έχει βάρος W.
- Το συνολικό βάρος μεταξύ X και Y που είναι το W, εάν μπορούμε να βρούμε μια διαδρομή μεταξύ X και Z του βάρους W1 και υπάρχει «εύρημα» μεταξύ Z και Y του βάρους W2 όπου το W είναι W1 + W2.

Στην συνέχεια γράψτε έναν απλό κανόνα όπου αν το X είναι συνδεδεμένο στο Z και το Z είναι συνδεδεμένο στο Y, τότε υπάρχει μια διαδρομή μεταξύ X και Y στην περίπτωση όπου δεν υπάρχουν βάρη.

**Καλή επιτυχία !!!**

edge(a, b, 1).

edge(b, c, 2).

edge(c, d, 1).

edge(b, d, 4).

edge(c, e, 1).

edge(e, a, 10).

edge(e, d, 6).

path(X, X, 0, []).

path(X, Y, W, [Y]) :- edge(X, Y, W).

path(X, Y, W, [Z|T]) :- edge(X, Z, W1), path(Z, Y, W2, T), W is W1 + W2.

path(X, Y, W) :- path(X, Y, W, \_).





ΜΑΘΗΜΑ : ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ &  
ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ  
23 – 6 - 2020

Διδάσκοντες: Ιωάννης Σινάτκας, Παναγιώτης Μπάτος



**ΘΕΜΑ 2<sup>ο</sup> (μονάδες 6)**

Δίνεται ο παρακάτω πίνακας που αφορά την πρόσληψη ενός υπαλλήλου με βάση συγκεκριμένα κριτήρια.

Όνομα	Ηλικία age	Προϋπηρεσία years	Πόλη town
mary	18	5	kastoria
katerina	20	5	kastoria
nikoleta	22	6	kastoria
christina	24	7	kastoria
thomai	39	7	kastoria
georgia	40	8	kastoria
giota	39	8	kastoria

Οι υπάλληλοι που πληρούν τις παρακάτω προϋποθέσεις δικαιούνται σύνταξης (syntaxi) και η εταιρεία μπορεί να τους ενημερώσει για αυτό.

Προϋποθέσεις:

1. Ικανός για πρόσληψη είναι ο υπάλληλος που έχει προϋπηρεσία πάνω από 5 έτη (Years).
2. Δεν προσλαμβάνεται υπάλληλος που η ηλικία του είναι πάνω από 40 έτη
3. Δεν δικαιούται πρόσληψης ένας κάτοικος άλλης πόλης εκτός από την Καστόρια.

Από τον παρακάτω πίνακα προκύπτουν τα ακόλουθα facts

age(mary,18).	years (mary, 5).	kastoria (mary).
age(katerina,20).	years(katerina,5).	kastoria(katerina).
age(nikoleta,22).	years(nikoleta, 6).	kastoria(nikoleta).
age(christina,24).	years (christina,7).	kastoria (christina).
age(thomai,39).	years(thomai,7).	kastoria(thomai).
age(georgia,40).	years(georgia, 8).	kastoria(georgia).
age(giota,39).	years(giota, 8).	kastoria(giota)

Δημιουργήστε τους κατάλληλους κανόνες για ένα project το οποίο θα αποθηκεύσετε ως **asep.pl** και στην συνέχεια βρείτε ποιοι υπάλληλοι με βάση τον πίνακα είναι ικανοί για πρόσληψη

**Καλή επιτυχία !!!**

age(mary,18).  
age(katerina,20).  
age(nikoleta,22).  
age(christina,24).  
age(thomai,39).  
age(georgia,40).  
age(giota,39).

years(mary,5).  
years(katerina,5).  
years(nikoleta,6).  
years(christina,7).  
years(thomai,7).  
years(georgia,8).  
years(giota,8).

kastoria(mary).  
kastoria(katerina).  
kastoria(nikoleta).  
kastoria(christina).  
kastoria(thomai).  
kastoria(georgia).  
kastoria(giota).

proslipsi(Employee):-years(Employee,Years),Years>5, ikanos(Employee).  
ikanos(Employee):-age(Employee,Years) , Years>40,! ,fail.  
ikanos(Employee):-not(kastoria(Employee)),! ,fail.  
ikanos(Employee):-kastoria(Employee).

1. Δημιουργείστε μία δομή δεδομένων (όρισμα μέσα σε κατηγορήμα) μέσα σε ένα fact - κατηγορήμα όπου θα εμφανίζεται η ημερομηνία γέννησης ενός ανθρώπου. Στην συνέχεια κάντε matching.

born(maria, date(13,11,1995)).

born(john, date(7,11,1994)).

#### ΕΡΩΤΗΣΗ

born(panos, date(23,7,1974)).     TRUE

2. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε μία γραμμή

#### ΕΡΩΤΗΣΗ

born(panos, A).

3. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε πολλές γραμμές

#### ΕΡΩΤΗΣΗ

born(panos, date(A,B,C)).

4. Κατόπιν βρείτε αυτούς που γεννήθηκαν μία συγκεκριμένη χρονολογία

#### ΕΡΩΤΗΣΗ

born(X, date(\_,\_,A)).

5. Κατασκευάστε υπό μορφή λίστας ένα fact – κατηγορήμα στο οποίο θα αποτυπώνονται ο πατέρας και οι τέσσερις του γιοι.

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

6. Δημιουργείστε ένα ερώτημα όπου θα γίνεται matching του πατέρα και των τεσσάρων γιών του που να εμφανίζεται η κεφαλή και η ουρά της λίστας. Να σημειωθεί ότι η συγκεκριμένη εμφάνιση είναι ΑΝΑΔΡΟΜΗ. γιατί;

**ΕΡΩΤΗΣΗ**

children(a,[X|Y]).

ΔΙΟΤΙ Η ΕΡΓΑΣΙΑ ΠΟΥ ΓΙΝΕΤΑΙ ΓΙΑ ΤΗΝ ΚΕΦΑΛΗ ΤΗΣ ΛΙΣΤΑΣ ΓΙΝΕΤΑΙ ΣΤΗΝ ΣΥΝΕΧΕΙΑ ΚΑΙ ΓΙΑ ΤΗΝ ΟΥΡΑ

**MEMBER**

7. Δημιουργήστε μία ερώτηση σε prolog όπου θα εξετάσετε αν κάποιο στοιχείο είναι μέλος της λίστας.

**ΕΡΩΤΗΣΗ**

member(1, [1,2,3,4]).

TRUE

8. Δημιουργήστε μία ερώτηση σε prolog όπου θα αποτυπώνονται όλα τα στοιχεία μίας λίστας.

**ΕΡΩΤΗΣΗ**

member(X, [1,2,3,4]).

9. Να ορισθεί αναδρομικά το member δηλαδή:

- Ο πρώτος κανόνας (true) θα είναι ο κανόνας τερματισμού δηλαδή ότι αληθεύει το member όταν το πρώτο όρισμα είναι το X και το δεύτερο όρισμα είναι μία λίστα με κεφαλή X και τα υπόλοιπα στοιχεία οποιαδήποτε. (αληθεύει το member αν το X είναι το πρώτο στοιχείο της λίστας).

- Ο δεύτερος κανόνας θα είναι ο αναδρομικός (false) όπου αν το στοιχείο  $X$  ανήκει στην λίστα  $L$  τότε το `member` αληθεύει .

### ΑΝΑΔΡΟΜΗ

`member(X, [X|_]) .`

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

ΚΑΝΟΥΜΕ ΚΑΤΙ ΣΤΗΝ ΚΕΦΑΛΗ ΚΑΙ ΜΕΤΑ ΣΤΗΝ ΥΠΟΛΟΙΠΗ ΛΙΣΤΑ

### APPEND ΣΥΝΕΝΩΣΗ

10. Δημιουργήστε ένα `fact – append` όπου θα ενώνονται οι λίστες  $L1$ ,  $L2$  και θα σχηματίζεται η  $L$ .

`append([1,2,3],[4,6,7], L).`

11. Δημιουργήστε ένα `fact-append` στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την πρόσθεση μίας λίστας σε μία άλλη από αριστερά.

`append(L, [6,7],[4,6,7]).`

`append( [1,2],L,[1,2,3,4,5,6,7]).` (από δεξιά)

12. Δημιουργήστε ένα `fact-append` στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την συνένωση δύο λιστών με όλους τους δυνατούς τρόπους

`append(L1, L2,[4,6,7]).`

13. Να ορισθεί αναδρομικά το `append` δηλαδή:

- Θα κατασκευάσετε έναν κανόνα τερματισμού όπου θα συνενώνουμε μία κενή λίστα με μία άλλη  $L$  και το αποτέλεσμα θα είναι το  $L$ .

- Θα κατασκευάσετε ένα αναδρομικό κανόνα όπου θα συνενώνεται μία λίστα L1 με την L2 και θα προκύπτει η λίστα L3 με κεφαλή το X και ουρά μία λίστα L.

append([], L,L).

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

ΣΥΝΕΝΩΣΗ ΚΕΝΗΣ ΛΙΣΤΑ ΜΕ ΛΙΣΤΑ L ΚΑΙ ΠΡΟΚΥΠΤΕΙ Η L.

ΚΑΝΩ APPEND ΜΙΑ ΛΙΣΤΑ ΜΕ ΚΕΦΑΛΗ Χ ΚΑΙ ΟΥΡΑ ΜΙΑ ΛΙΣΤΑ L1 ΜΕ ΜΙΑ ΑΛΛΗ ΛΙΣΤΑ L2 ΚΑΙ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΠΡΕΠΕΙ ΝΑ ΕΙΝΑΙ ΜΙΑ ΛΙΣΤΑ ΜΕ ΚΕΦΑΛΗ ΤΟ Χ ΚΑΙ ΟΥΡΑ ΤΗΝ L3. ΑΥΤΟ ΑΛΗΘΕΥΕΙ ΟΤΑΝ ΕΝΩΝΩ L1 ΚΑΙ L2 ΚΑΙ ΠΑΙΡΝΩ L3. ΣΤΗΝ ΟΥΣΙΑ ΑΡΧΙΚΑ ΓΙΝΕΤΑΙ ΕΝΩΣΗ ΤΩΝ ΟΥΡΩΝ L1 ΚΑΙ L2 ΚΑΙ ΣΧΗΜΑΤΙΖΕΤΑΙ Η ΟΥΡΑ L3 ΚΑΙ ΜΕΤΑ ΕΠΙΣΥΝΑΠΤΕΤΑΙ ΤΟ Χ ΣΤΗΝ L3.

**ΕΡΩΤΗΣΗ**→ append([1,2], [3], L).

**LENGTH**

14. Δημιουργήστε ένα ερώτημα στο οποίο θα γίνεται η καταμέτρηση των στοιχείων της λίστας

**ΕΡΩΤΗΣΗ**→ length([a,k,B,j,2],X).

**REVERSE**

15. Δημιουργήστε ένα ερώτημα με το οποίο θα γίνεται η αντιστροφή μίας λίστας.

**ΕΡΩΤΗΣΗ**→ reverse([1,4,6,7], L).

## Άσκηση 1

### **ΑΦΑΙΡΕΣΗ**

Ένας ηλεκτρονικός υπολογιστής έχει ένα πλήθος επικίνδυνων διασκορπισμένων αρχείων στον δίσκο του, που ολόκληρος είναι ένας ενιαίος φάκελος. Θεωρώντας ότι τα ονόματα αυτών των αρχείων είναι ίδια (trojan) και όλα μαζί τα αρχεία αποτελούν λίστα ονομάτων, να δημιουργήσετε πρόγραμμα σε prolog που να αφαιρεί από τη λίστα των ονομάτων, τα αρχεία με όνομα trojan.

Δοκιμάστε το πρόγραμμά σας με τη λίστα [f1, trojan, d, trojan, f2]).

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

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

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

`afairw(trojan,L,[f, trojan,d, trojan,f]).`

## Άσκηση 2

### **ΑΝΤΙΚΑΤΑΣΤΑΣΗ**

Μία ηλεκτρική συσκευή αποτελείται από 5 διαφορετικά εξαρτήματα  $p_1, p_2, p_3, p_4, p_5$  τα οποία βρίσκονται στην κεντρική πλακέτα της και λειτουργούν το καθένα ξεχωριστά με βάση συγκεκριμένο λογισμικό  $s_1, s_2 \dots s_5$ . Το κάθε εξάρτημα δηλαδή έχει το δικό του λογισμικό το οποίο όμως περιοδικά χρειάζεται αντικατάσταση με αναβάθμισή του σε νεότερο  $n_i$  με  $i=1,2..5$ . Την αντικατάσταση αναλαμβάνει διαγνωστικό σύστημα που ελέγχει τα λογισμικά και αντικαθιστά το παρωχημένο  $s_i$  με νεότερο  $n_i$  και αυτό γίνεται με κατάλληλο πρόγραμμα της prolog όταν συνδέσουμε τη συσκευή στο διαγνωστικό σύστημα.

Γράψτε το πρόγραμμα του διαγνωστικού συστήματος που υλοποιεί την αντικατάσταση του λογισμικού, συγκεκριμένου εξαρτήματος, με το αναβαθμισμένο νεότερο λογισμικό. Κάντε τις σχετικές δοκιμές.

change(E,NE,L,NL):-

append(A,[E | B],L),append(A,[NE | B],NL).

change(s3, n3, [s1, s2, s3, s4, s5], NL).

### Άσκηση 3

#### **ΑΘΡΟΙΣΜΑ**

Υποθέτουμε ότι ο covid-19 αποτελείται από 5 διαφορετικά είδη μικροβίων, των οποίων ο πληθυσμός του κάθε είδους καταχωρείται ημερήσια σε κατάλληλη λίστα, με συγκεκριμένη καθορισμένη σειρά. Η συνάρτηση αξιολόγησης της διάγνωσης του ασθενούς είναι της μορφής  $2 * \alpha_i - 3 * \beta_i$ , όπου  $\alpha_i$  ο πληθυσμός του  $i$  είδους μικροβίου την πρώτη ημέρα και  $\beta_i$  ο αντίστοιχος πληθυσμός την δεύτερη ημέρα. Οι γιατροί βλέπουν τον πίνακα μικροβίων πώς διαμορφώνεται μετά τη δεύτερη ημέρα και αν ο πληθυσμός οποιουδήποτε από τα 5 μικρόβια είναι θετικός αποφαίνονται ότι εξεταζόμενος άνθρωπος είναι θετικός στον covid-19. Στην περίπτωση όπου όλοι οι αριθμοί είναι αρνητικοί ή μηδέν τότε αποφαίνονται ότι είναι αρνητικός.

Διαμορφώστε κατάλληλο πρόγραμμα σε prolog που να κάνει τη σχετική διάγνωση. Δοκιμάστε το πρόγραμμά σας καταχωρώντας σε δυο λίστες διάφορους πληθυσμούς μικροβίων και ελέγξτε τα αποτελέσματα που προκύπτουν από τη συνάρτηση αξιολόγησης σε τρίτη λίστα.

addlist([],[],[]).

addlist([H1 | T1],[H2 | T2],[H3 | T3]):-

H3 is 2\*H1-3\*H2, addlist(T1,T2,T3).

addlist([4,1,3,2, 4],[5,6,1,0,7],X).



#### Άσκηση 4

##### ΚΛΩΝΟΙ

Όσοι ασχοληθείτε με την prolog σε επαγγελματικό επίπεδο θα συναντήσετε το παραθυρικό περιβάλλον της prolog όπου ένα από τα πιο χρήσιμα προγράμματα είναι η δημιουργία back-up αρχείων. Ο Στρατός η Αστυνομία και οι Δικαστικές αρχές έχουν προγράμματα που δημιουργούν κλώνους (δηλαδή ίδια αρχεία). Στα πλαίσια αυτού του μαθήματος ζητείται να δημιουργήσετε στο περιβάλλον SWI-PROLOG ένα project όπου δίπλα σε κάθε στοιχείο μίας λίστας να εμφανίζεται ένα όμοιο του, να δημιουργεί δηλαδή δεύτερη λίστα στην οποία θα εμφανίζονται δύο φορές τα στοιχεία της πρώτης.

`double([],[]):-!.`

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

`double([x,-9,[7],g],C).`

#### Άσκηση 5

##### ΛΕΞΙΚΟ

Η prolog δίνει την δυνατότητα να αντιστοιχίζονται κάποιες μεταβλητές ή κάποιες σταθερές με κάποιες άλλες. Μία πολύ καλή εφαρμογή θα μπορούσε να ήταν ένα λεξικό όρων – μεταφραστικό όπου μία ελληνική λέξη αντιστοιχίζεται σε μία αγγλική και αντίστροφα. Αυτό μπορεί να αποτελεί τη στοιχειώδη γνωσιακή βάση του συστήματος.

Να αναπτύξετε ένα απλό πρόγραμμα σε prolog στο οποίο να δίνεται σειρά λέξεων στα ελληνικά και το πρόγραμμα να αποδίδει τη σειρά των λέξεων στα αγγλικά και αντίστροφα.

`a([],[]).`

`a([H|T],[H1|T1]) :-b(H,H1),a(T,T1).`

`a(L1,L2):-a(L2,L1).`

b('φοιτητής', 'student').

b('πληροφορική', 'informatics').

b('κωρονοϊός', 'covid19').

b('εξετάσεις', 'exams').

b('σπίτι', 'home').

b('θέλω', 'want').

b(X, M).

b('θέλω', M).

Εδώ μπορούμε να χρησιμοποιήσουμε και φράσεις  
αντί λέξεις.

a([θέλω, σπίτι], M).

M = [want, home].

ή

a([want, home], M).

M = [θέλω, σπίτι] .

## **Άσκηση 1**

Δίνεται η ακόλουθη βάση γνώσης:

```
likes(john,mary).
```

```
likes(john,trains).
```

```
likes(peter,fast_cars).
```

```
likes(Person1,Person2):- hobby(Person1,Hobby),  
hobby(Person2,Hobby).
```

```
hobby(john,trainspotting).
```

```
hobby(tim,sailing).
```

```
hobby(helen,trainspotting).
```

```
hobby(simon,sailing).
```

Τι αποτελέσματα θα εμφανίσουν τα ακόλουθα ερωτήματα;

```
1.likes(trains,john).
```

```
2.likes(helen,john).
```

```
3.likes(tim,helen).
```

```
4.likes(john,X).
```

## **Άσκηση 2**

Δίνεται η ακόλουθη βάση γνώσης:

```
hold_party(X):-birthday(X), happy(X).
```

```
birthday(tom).
```

```
birthday(fred).
```

```
birthday(helen).
```

happy(mary).

happy(jane).

happy(helen).

Τι αποτελέσματα θα εμφανίσουν τα ακόλουθα ερωτήματα;

1.birthday(jane).

2.hold\_party(X).

### **Άσκηση 3**

Δίνεται η ακόλουθη βάση γνώσης:

woman(helen).

woman(maria).

woman(sofia).

loves(nikos, helen).

loves(panos, helen).

loves(petros, viki).

loves(viki, petros).

jealous(X,Y):-loves(X,Z), loves(Y,Z).

Τι αποτελέσματα θα εμφανίσουν τα ακόλουθα ερωτήματα;

1.woman(X).

2.loves(panos,X), woman(X).

3.loves(petros,X), woman(X).

4.jealous(panos,X).

#### **Άσκηση 4**

Δίνεται η ακόλουθη βάση γνώσης:

```
bigger(elephant, horse).
```

```
bigger(horse, donkey).
```

```
bigger(donkey, dog).
```

```
bigger(dog,
```

```
monkey).
```

```
is_bigger(X,Y):-bigger(X,Y).
```

```
is_bigger(X,Y):-bigger(X,Z), is_bigger(Z,Y).
```

Τι αποτελέσματα θα εμφανίσουν τα ακόλουθα ερωτήματα;

1. `is_bigger(elephant, monkey).`

2. `is_bigger(X,donkey).`

## ΑΣΚΗΣΕΙΣ ΜΕ ΛΙΣΤΕΣ

### 1. Αναγραφή όλων των στοιχείων μίας λίστας

`write_list([]).`

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

Ερώτηση: `write_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]).`

### 3.Εμφάνιση στοιχείου λίστας

`emfanisi(1,[M|_],M).`

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

Ερώτηση: `emfanisi(3,[g,b,j,u,k,o],K).`

### 4. Πρώτο και τελευταίο στοιχείο λίστας

`prwto(B,[B|_]).`

`teleytaio(L,[L]):-!.`

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

Ερωτήματα:

teleytaio(K,[h,j,k,l,h,g,f]).

prwto(K,[h,j,k,l,h,g,f]).

## 5. Γέμισμα και άδειασμα 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).

## 6. Μετρητής στοιχείων λίστας.

*1ος τρόπος: τερματική επιστροφή*

length([],Result,Result):-!.

length([\_|T],Result,Counter):-Ncounter is Counter+1,

length(T,Result,Ncounter).

Ερώτηση: length([a,g,n],X,0).

*2ος τρόπος: αρχική επιστροφή*

length([],0):-!.

length([\_|T],L):-

length(T,TailLength),

L is TailLength + 1.

Ερώτηση: length([a,g,n],X).

#### 7. Άθροισμα στοιχείων λίστας - αρχική επιστροφή

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,\_).

#### 8. 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).

#### 9. Γέμισε με a μια λίστα.

list([],0):-!.

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

Ερωτήματα: list(H,4).

#### 10. Αφαίρεσε κάποιο γράμμα 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]).`

11. Αφαίρεση από λίστα όλων των 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).`

12. Πρόσθεση των ίδιων στοιχείων σε μια λίστα

`double([],[]):-!.`

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

Ερωτήματα: `double([x,-9,7,g,a],C).`

13. 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]).

3. syndese(L3,[l,k],[h,k,l,k]).

14. Αντιστροφή λίστας

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

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

Ερωτήματα:

antistrofi([a,b,c],[],L).

15. Πιθανοί συνδυασμοί στοιχείων λίστας

perm([],[]).

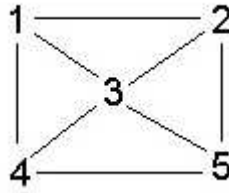
perm([H|T],Perm):-perm(T,SP),insert(H,SP,Perm).

insert(X,T,[X|T]).

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

Ερωτήματα: perm([1,2,3],P).

## ΑΣΚΗΣΗ 1



edge(1,4).

edge(1,3).

edge(2,3).

edge(2,5).

edge(3,4).

edge(3,5).

edge(4,5).

connected(X,Y) :- edge(X,Y) ; edge(Y,X).

path(A,B,Path) :-travel(A,B,[A],Q), reverse(Q,Path).

travel(A,B,P,[B|P]) :- connected(A,B).

travel(A,B,Visited,Path) :-connected(A,C), C \==  
B,\+member(C,Visited),

travel(C,B,[C|Visited],Path).

## ΑΣΚΗΣΗ 2

```
directTrain('mavrochori','gkirole','deksia').  
directTrain('gkirole','ampelokipoi','aristera').  
directTrain('ampelokipoi','militsa','eythia').  
directTrain('militsa','kostarazi','deksia').  
directTrain('kostarazi','vogatsiko','eythia').  
directTrain('vogatsiko','neapoli','aristera').  
directTrain('neapoli','mikrokastro','deksia').
```

travelBetween(X,Y,M):-

```
directTrain(X,Y,M),write(X),write('-->'),write(M),write('-->'),write(Y).
```

travelBetween(X,Y,A):-

```
directTrain(X,Z,M),write(X),write('-->'),write(M),write('-->'),write(Z),write('-->'),travelBetween(Z,Y,A).
```

## ΑΣΚΗΣΗ - ΔΙΑΓΝΩΣΤΙΚΟ

Σχεδιάστε μία διαγνωστική εφαρμογή για τον εντοπισμό βλαβών σε έναν Η/Υ όπου:

1. Όταν η οθόνη παρουσιάζει παραμορφώσεις και μαυρίζει τότε υπάρχει βλάβη στην κάρτα γραφικών.
2. Όταν η οθόνη μαυρίζει τότε η βλάβη είναι καμένη οθόνη.
3. Όταν ο υπολογιστής κάνει επανεκκινήσεις ή κλείνει απότομα τότε υπάρχει ιός.
4. Όταν κλείνει τότε υπάρχει υπερθέρμανση.
5. Όταν κολλάει τότε υπάρχει βλάβη στον σκληρό δίσκο

Στις 11 Απριλίου 2022 η εταιρεία επισκευής παρέλαβε τους εξής υπολογιστές με τα ακόλουθα προβλήματα:

simptoma(pc1,paramorfosi).

simptoma(pc1,mavro).

simptoma(pc2,mavro).

simptoma(pc3,epanekinei).

simptoma(pc4,kleinei).

simptoma(pc5,kollaei).

simptoma(pc6,paramorfosi).

simptoma(pc7,kollaei).

simptoma(pc7,kleinei).

simptoma(pc8,epanekinei).

simptoma(pc8,mavro).

### Ερωτήσεις:

Γράψτε τους αντίστοιχους κανόνες για την παραπάνω λεκτική περιγραφή.

1. Εντοπίστε αν υπάρχουν pc που έχουν πρόβλημα με την κάρτα γραφικών.
2. Τι βλάβη έχει το pc2;
3. Εντοπίστε αν υπάρχουν pc που έχουν ιό.

simptoma(pc1,paramorfosi).

simptoma(pc1,mavro).

simptoma(pc2,mavro).

simptoma(pc3,epanekinei).

simptoma(pc4,kleinei).

simptoma(pc5,kollaei).

simptoma(pc6,paramorfosi).

simptoma(pc7,kollaei).

simptoma(pc7,kleinei).

simptoma(pc8,epanekinei).

simptoma(pc8,mavro).

vlavi(X,karta\_grafikwn):-simptoma(X, paramorfosi), simptoma(X, mavro).

vlavi(X,kameni\_othoni):-simptoma(X,mavro).

vlavi(X,ios):- simptoma(X, epanekinei); simptoma(X,kleinei).

vlavi(X,yperthemansi):- simptoma(X,kleinei).

vlavi(X,skliros):- simptoma(X,kollaei).

**ΠΟΙΟ PC ΕΧΕΙ ΘΕΜΑ ΜΕ ΚΑΡΤΑ ΓΡΑΦΙΚΩΝ**

vlavi(X,karta\_grafikwn).

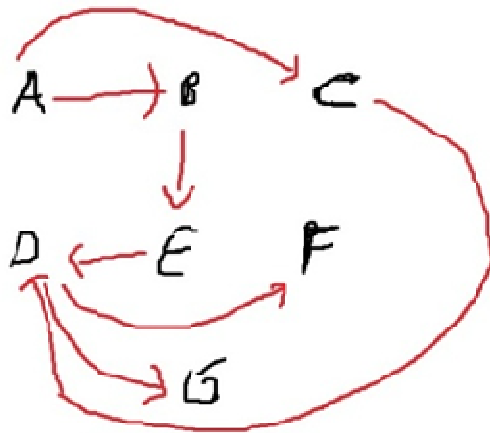
**ΤΙ ΕΧΕΙ ΤΟ pc2**

vlavi(pc2,X).

**ΠΟΙΑ ΕΧΟΥΝ ΙΟ**

vlavi(X, ios).

### Άσκηση 1



Δίνεται ο παρακάτω κατευθυντικός γράφος χωρίς βάρη

Προκύπτει η ακόλουθη γνωσιακή βάση :

edge(a,b).

edge(b,e).

edge(a,c).

edge(c,d).

edge(e,d).

edge(d,f).

edge(d,g).

Στην συνέχεια γράψτε έναν απλό κανόνα όπου αν το X είναι συνδεδεμένο στο Z και το Z είναι συνδεδεμένο στο Y, τότε υπάρχει μια διαδρομή μεταξύ X και Y.

**Λύση:**

path(X,Y) :- edge(X,Y).

path(X,Y) :- edge(X,Z), path(Z,Y).



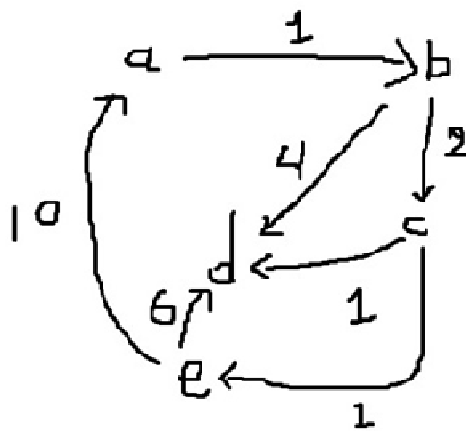
Στην συνέχεια βρείτε:

A. τις διαδρομές από το  $e$  προς έναν άλλο κόμβο  $path(e,A)$ .

B. τις διαδρομές προς το  $g$ .  $path(A,g)$ .

### ΑΣΚΗΣΗ κατευθυντικός γράφος με βάρη

Δίνεται ο ακόλουθος κατευθυντικός γράφος με βάρη που αφορά τα νεφρά ενός ανθρώπου και 5 περιοχές τους. Στις περιοχές αυτές κυκλοφορούν διάφορα υγρά. Η περιοχή d είναι η απόληξη δηλαδή η περιοχή όπου συγκεντρώνονται όλα τα υγρά και στην συνέχεια θα ακολουθήσει η διαδικασία αποβολής τους (δεν το εξετάζουμε εδώ). Δεν θα μπούμε σε περαιτέρω ιατρικές λεπτομέρειες παρά μόνο θα πούμε ότι η κάθε περιοχή στέλνει, συγκεντρώνει υγρά διαφορετικής φύσεως τα οποία κυκλοφορούν με διάφορες ταχύτητες και μετατρέπονται σε κάθε κόμβο (κάτι όμως που δεν μας απασχολεί στον παρόν πρόβλημα)



Προκύπτει η ακόλουθη γνωσιακή βάση

edge(a, b, 1).

edge(b, c, 2).

edge(c, d, 1).

edge(b, d, 4).

edge(c, e, 1).

edge(e, a, 10).

edge(e, d, 6).

Σκοπός μας είναι να βρούμε τρόπους κυκλοφορίας των υγρών και τις συνολικές ταχύτητες σε μη γειτονικούς κόμβους π.χ. από το a στο d. Έτσι πρέπει να γράψουμε κανόνες με τις ακόλουθες λογικές:

- Υπάρχει μονοπάτι (findapath) μεταξύ  $X$  και  $Y$  το οποίο έχει βάρος  $W$ , εάν υπάρχει ένα ακμή μεταξύ  $X$  και  $Y$  που έχει βάρος  $W$ .
- Το συνολικό βάρος μεταξύ  $X$  και  $Y$  που είναι το  $W$ , εάν μπορούμε να βρούμε μια διαδρομή μεταξύ  $X$  και  $Z$  του βάρους  $W1$  και υπάρχει «εύρημα» μεταξύ  $Z$  και  $Y$  του βάρους  $W2$  όπου το  $W$  είναι  $W1 + W2$ .

### Λύση:

path( $X, X, 0, []$ ).

path( $X, Y, W, [Y]$ ) :- edge( $X, Y, W$ ).

path( $X, Y, W, [Z|T]$ ) :- edge( $X, Z, W1$ ), path( $Z, Y, W2, T$ ),  $W$  is  $W1 + W2$ .

path( $X, Y, W$ ) :- path( $X, Y, W, \_$ ).

Τρέξτε το πρόγραμμα με την εντολή

?- path(a,d,K).

### ΑΣΚΗΣΗ ΜΕ ΒΙΤΑΜΙΝΕΣ

Δίνονται 3 ομάδες δεδομένων για τις οποίες πρέπει να γράψετε συγκεκριμένα γεγονότα και κανόνες.

**Η πρώτη ομάδα δεδομένων αφορά διατροφικούς κανόνες και μπορεί να κωδικοποιηθεί με την εισαγωγή της σχέσης της γενικής μορφής  $needs(X, Y)$ , όπου το άτομο  $X$  σύμφωνα με τους διατροφικούς κανόνες πρέπει να λάβει την τροφή  $Y$ .**

Μια έγκυος χρειάζεται φολικό οξύ.

Κάποιος που είναι κρυωμένος, πρέπει να λάβει επιπλέον βιταμίνη C.

Άνθρωποι που πάσχουν από αναιμία χρειάζονται σίδηρο.

Η βιταμίνη D απαιτείται για την καλύτερη απορρόφηση του ασβεστίου.

Όταν ένας οργανισμός παρουσιάσει έλλειψη σε κάποιο στοιχείο τότε χρειάζεται να λάβει το στοιχείο αυτό.

Άνθρωποι που πάσχουν από ραχιτισμό χρειάζονται ασβέστιο.

**Η δεύτερη είναι προσωπικά δεδομένα των ατόμων, δηλαδή σε ποια κατάσταση βρίσκεται ο καθένας τους:**

Η Μαίρη (Mary) είναι έγκυος.

Η Τζένη (Jane) είναι έγκυος.

Ο Γιάννης (John) είναι κρυωμένος.

Ο Πέτρος (Peter) είναι αναιμικός.

Ο Δημήτρης (Jim) πάσχει από ραχιτισμό (rickets).

Ο Γιώργος (George) έχει έλλειψη βιταμίνης A.

**Η τρίτη ομάδα δεδομένων συσχετίζει συγκεκριμένα τρόφιμα και κατηγορίες τροφίμων με συγκεκριμένες ουσίες. Αυτό γίνεται με την εισαγωγή της σχέσης  $found\_in(X,Y)$  όπου  $X$  είναι η ουσία (Nutrient) που βρίσκεται στην τροφή ή την κατηγορία τροφής  $Y$ . Είναι:**

Τα καρότα (carrots) περιέχουν βιταμίνη A.

Η βιταμίνη A και η βιταμίνη D βρίσκεται στα λιπαρά ψάρια (oily fish).

Το ασβέστιο (calcium) υπάρχει στα όσπρια (pulse).

Τα πράσινα λαχανικά (green veg) περιέχουν φολικό οξύ, βιταμίνη Α και βιταμίνη Ε.

Τα πορτοκάλια (orange) περιέχουν φολικό οξύ (folic acid) και βιταμίνη C.

Τα όσπρια περιέχουν φολικό οξύ.

Τα αυγά (eggs) περιέχουν βιταμίνη D.

Το σπανάκι (spinach) περιέχει σίδηρο (iron).

Το μαρούλι (lettuce), το λάχανο (cabbage) και ο μαϊντανός (parsley) είναι πράσινα λαχανικά (green\_veg).

Οι φακές (lentils), τα μπιζέλια (peas) και τα φασόλια (beans) είναι όσπρια (pulse).

Το σκουμπρί (mackerel), η σαρδέλα (sardine) και η ρέγκα (herring) είναι λιπαρά ψάρια (oily\_fish).

**Στην συνέχεια πρέπει να ορίσετε την σχέση `should_eat(Person, Food)` η οποία θα συνδυάζει το άτομο `Person` με την τροφή `Food` που πρέπει να λάβει. Έτσι πρέπει να ισχύει η συνθήκη `needs(Person, Nutrient)` και εννοείται ότι στην τροφή `Food`, πρέπει να υπάρχει το στοιχείο `Nutrient` που χρειάζεται το άτομο, δηλαδή να ισχύει η συνθήκη `found_in(Nutrient, Food)`.**

#### ΛΥΣΗ

`needs(X, folic_acid) :- pregnant(X).`

`needs(X, vitamin_C) :- has_cold(X).`

`needs(X, iron) :- aenemic(X).`

`needs(X, Vitamin_D) :- needs(X, calcium).`

`needs(X, Y) :- deficient(X,Y).`

`needs(X, calcium) :- has_rickets(X).`

`pregnant(mary).`

`pregnant(jane).`

`has_cold(john).`

`aenemic(peter).`

has\_rickets(jim).

deficient\_in(george, vitamin\_A).

found\_in(vitamin\_A, carrots).

found\_in(vitamin\_A, X) :- oily\_fish(X).

found\_in(vitamin\_D, X) :- oily\_fish(X).

found\_in(calcium, X) :- pulse(X).

found\_in(vitamin\_E, X) :- green\_veg(X).

found\_in(vitamin\_A, carrots) :- green\_veg(X).

found\_in(folic\_acid, X) :- green\_veg(X).

found\_in(folic\_acid, oranges).

found\_in(vitamin\_C, oranges).

found\_in(folic\_acid, X) :- pulse(X).

found\_in(vitamin\_D, eggs).

found\_in(iron, spinach).

green\_veg(lettuce).

green\_veg(cabbage).

green\_veg(parsley).

pulse(lentils).

pulse(peas).

pulse(beans).

oily\_fish(mackerel).

oily\_fish(sardines).

should\_eat(Person, Food) :- needs(Person, Nutrients), found\_in(Nutrient, Food).



### ΑΣΚΗΣΗ ΜΕ ΛΙΣΤΕΣ – ΑΝΤΙΣΤΟΙΧΗΣΗ ΟΡΙΣΜΩΝ

Στην άσκηση αυτή έχουμε αντιστοίχιση μεθέγοις και ορισμών της φυσικής.

orismos([], []).

orismos([H|T],[H1|T1]) :-orismos\_antistixixi(H,H1),

orismos(T,T1).

orismos\_antistixixi('ορμή', 'μάζα επί ταχύτητα').

orismos\_antistixixi('έργο', 'δύναμη επί απόσταση').

orismos\_antistixixi('ισχύς', 'έργο δια χρόνος').

orismos\_antistixixi('απόσταση', 'ταχύτητα επι χρόνος').

orismos\_antistixixi('ταχυτητα', 'αποσταση δια χρόνος').

orismos\_antistixixi('δυναμη', 'μαζα επι επταχυνση').

Ερωτήματα προς μελέτη

orismos\_antistixixi('δυναμη', 'μαζα επι επταχυνση').

βγαζει TRUE

orismos\_antistixixi('δυναμη', C).

BGAZEI C=μαζα επι επταχυνση



## ΑΣΚΗΣΗ ΜΕ ΛΙΣΤΕΣ – ΑΝΤΙΣΤΟΙΧΗΣΗ

orismos([], []).

orismos([H|T],[H1|T1]) :-orismos\_antistixixi(H,H1),

orismos(T,T1).

orismos\_antistixixi('ορμή', 'μάζα επί ταχύτητα').

orismos\_antistixixi('έργο', 'δύναμη επί απόσταση').

orismos\_antistixixi('ισχύς', 'έργο δια χρόνος').

orismos\_antistixixi('απόσταση', 'ταχύτητα επι χρόνος').

orismos\_antistixixi('ταχυτητα', 'αποσταση δια χρόνος').

orismos\_antistixixi('δυναμη', 'μαζα επι επταχυνση').

orismos\_antistixixi('δυναμη', 'μαζα επι επταχυνση').

βγαζει TRUE

orismos\_antistixixi('δυναμη', C).

BGAZEI C=μαζα επι επταχυνση

## **ΑΣΚΗΣΗ: ΣΥΝΑΝΤΗΣΗ ΦΟΙΤΗΤΩΝ**

Το παρακάτω project αφορά την παρακολούθηση μαθημάτων από φοιτητές. Υπάρχουν τα ακόλουθα γεγονότα:

1. Το γεγονός `course(X)` δηλώνει τον κωδικό `X` ενός μαθήματος
2. Το γεγονός `department(X,Y)` δηλώνει ότι το μάθημα με κωδικό `X` προσφέρεται από το Τμήμα `Y`,
3. Το γεγονός `student(X)` δηλώνει ότι ο `X` είναι φοιτητής
4. Το γεγονός `enrolled(X,Y)` δηλώνει ότι ο φοιτητής `X` παρακολουθεί το μάθημα `Y`.

### **Αναλυτικότερα:**

Όπου τα γεγονότα `course(X)` δηλώνουν μάθημα με κωδικό `X`, τα γεγονότα `department(X,Y)` δηλώνουν ότι το μάθημα με κωδικό `X` προσφέρεται από το Τμήμα `Y`, τα γεγονότα `student(X)` δηλώνουν ότι ο `X` είναι φοιτητής και τέλος τα γεγονότα `enrolled(X,Y)` δηλώνουν ότι ο φοιτητής `X` παρακολουθεί το μάθημα `Y`.

Δίνονται μαθήματα με κωδικούς 312, 322, 315 και 371.

Τα μαθήματα με κωδικούς 312 και 322 προσφέρονται από Τμήμα Υπολογιστών.

Το μάθημα με κωδικό 315 προσφέρεται από το Τμήμα Μαθηματικών.

Το μάθημα με κωδικό 371 προσφέρεται από το Φυσικής.

Η Μαίρη, Ο Γιάννης, η Τζέιν και η Πέτρος είναι φοιτητές.

Η Μαίρη παρακολουθεί τα μαθήματα 322 και 312 και 315.

Ο Γιάννης παρακολουθεί τα μαθήματα 322 και 315.

Η Τζέιν παρακολουθεί τα μαθήματα 312 και 322.

Ο Πέτρος παρακολουθεί το μάθημα 371.

Στην συνέχεια πρέπει να γράψουμε κατηγορημα όπου οι όλοι φοιτητές πηγαίνουν υποχρεωτικά τις παραδόσεις και ενδεχομένως να συναντιούνται κάποιοι με κάποιους. Το `meet(X,Y)` μας δίνει ποιοι από τους φοιτητές συναντούνται μεταξύ τους στις παραδόσεις.

### **Απαντήστε στα ακόλουθα ερωτήματα:**

Ποια μαθήματα παρακολουθεί η Μαίρη.

Ποιοι φοιτητές παρακολουθούν το μάθημα με κωδικό 322.

### ΛΥΣΗ

course(312).

course(322).

course(315).

course(371).

department(312, computer\_science).

department(322, computer\_science).

department(315, mathematics).

department(371, physics),

student(mary).

student(jane).

student(john).

student(peter).

enrolled(mary, 322).

enrolled(mary, 312).

enrolled(mary, 315).

enrolled(john, 322).

enrolled(john, 315).

enrolled(jane, 312).

enrolled(jane, 322).

enrolled(peter, 371).

meet(X, Y):student(X),

student(Y),

not(X = Y),

enrolled(X, Z),

enrolled(Y, Z),!

## ΑΣΚΗΣΗ ΙΚΑΝΟΤΗΤΑ ΓΙΑ ΣΥΝΤΑΞΗ

Δίνεται ο παρακάτω πίνακας που αφορά τους υπαλλήλους (employees) και τους διευθυντές (managers) μίας εταιρείας.

Όνομα	Ηλικία age	Προϋπηρεσία service	Ειδικότητα
petros	60	10	shopfloor
larian	55	5	manager
kostas	57	26	shopfloor
na8anail	60	9	shopfloor
iakovos	49	24	shopfloor
ilarios	56	8	manager
panos	58	30	shopfloor

Οι υπάλληλοι που πληρούν τις παρακάτω προϋποθέσεις δικαιούνται σύνταξης (syntaxi) και η εταιρεία μπορεί να τους ενημερώσει για αυτό.

### Προϋποθέσεις:

1. Ικανός για λήψη σύνταξης είναι ο υπάλληλος που έχει προϋπηρεσία πάνω από 20 έτη (Years).
2. Δεν δικαιούται σύνταξης υπάλληλος που η ηλικία του είναι κάτω από 50 έτη
3. Δεν δικαιούται σύνταξης από το ΙΚΑ ένας manager όμως δικαιούται ένας shopfloor.

Από τον παρακάτω πίνακα προκύπτουν τα ακόλουθα facts

age(petros,60).

age(larian,55).

age(kostas,57).  
age(na8anail,60).  
age(iakovos,49).  
age(ilarios,56).  
age(panos,58).

service(kostas,26).  
service(petros,10).  
service(na8anail,19).  
service(iakovos,24).  
service(larian,5).  
service(ilarios, 8).  
service(panos,30).

manager(larian).  
manager(ilarios).

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

Δημιουργήστε τους κατάλληλους κανόνες για ένα project το οποίο θα αποθηκεύσετε ως **ika.pl** και στην συνέχεια βρείτε ποιοι υπάλληλοι με βάση τον πίνακα είναι ικανοί προς συνταξιοδότηση.

syntaxi(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).

ΕΡΩΤΗΜΑ

**syntaxi(Employee)**

#### ΔΙΝΕΤΑΙ Η ΒΑΣΗ ΓΝΩΣΗΣ

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).

#### ΑΠΑΝΤΗΣΤΕ ΣΤΑ ΑΚΟΛΟΥΘΑ ΕΡΩΤΗΜΑΤΑ

Ποια οχήματα έχουν τιμή 25000 €; vehicle(X,A,B,T,25000).

Εμφανίστε όλα τα datsun οχήματα. vehicle(datsun,X,A,B,G).

Εμφανίστε όλα τα red αυτοκίνητα. vehicle(A,B,C,red,Y).

Εμφανίστε μόνο τα car ford. car(ford,B,C,D,T).

Εμφανίστε τα car ford και τα truck με χρώμα black.

car(ford,B,C,D,T);truck(A,B,C,black,R).

Εμφανίστε τα οχήματα με τιμή μεγαλύτερη από 20000.

vehicle(A,B,C,D,Price),Price>20000.

Εμφανίστε τα οχήματα με χιλιόμετρα λιγότερα από 50000.

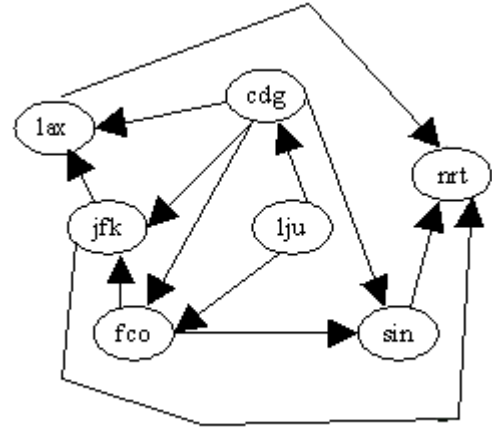
vehicle(A,Odometer,C,D,F),Odometer<20000.

## ΚΑΤΕΥΘΥΝΤΙΚΟΣ ΓΡΑΦΟΣ ΜΕ ΒΑΡΗ

```

flight(lju,cdg,2,955).
flight(lju,fco,1,506).
flight(cdg,fco,3,1100).
flight(cdg,jfk,7,5830).
flight(cdg,lax,10,9096).
flight(cdg,sin,12,10712).
flight(jfk,lax,5,3971).
flight(jfk,nrt,6,5486).
flight(lax,nrt,7,6230).
flight(fco,jfk,8,6861).
flight(fco,sin,11,10039).
flight(sin,nrt,11,9857).

```



```

waiting(cdg,4).
waiting(fco,5).
waiting(jfk,2).
waiting(lax,2).
waiting(sin,1).
waiting(lju,1).
waiting(nrt,1).

```

```

connExists(Start,Destination):- flight(Start, Destination,_,_).
connExists(Start,Destination):- flight(Start, X,_,_),connExists(X,Destination).

```

```

travelTime(Start,Destination,Time,Path):-
flight(Start,Destination,FlightTime,_),waiting(Start,WTimeStart), waiting(Destination,WTimeDest),
Time is FlightTime + WTimeStart + WTimeDest,conc([Start],[Destination],Path).

```

```

travelTime(Start,Destination,Time,Path):- flight(Start,X,FlightTime,_), waiting(Start,WTimeStart1),
travelTime(X,Destination,TimeX,PathX), Time is FlightTime + TimeX
+WTimeStart1,conc([Start],PathX,Path).

```

```

allShortheestPaths:- waiting(X,_), waiting(Y,_), X == Y,
setof(T,(X^Y^C^P^travelTime(X,Y,T,P)),L), sort(L,[M|_]), assert(shortestPath(X,Y,M)), fail.

```

```
%pathLength(Path,Length)
```

```

pathLength([],0).
pathLength([_],0).
pathLength([X,Y|L],Length):- flight(X,Y,_,T),pathLength([Y|L], Length1), Length is Length1+ T.

```



## ΠΑΡΑΔΕΙΓΜΑ ΑΠΟΚΟΠΗΣ

Ορισμένα κατηγορήματα μπορούν να έχουν πολλούς διαφορετικούς τρόπους αποτίμησης ανάλογα με τις τιμές των παραμέτρων τους. Για παράδειγμα ένα κατηγορήμα που υπολογίζει την υποτροφία που πρέπει να πάρει ένας φοιτητής μπορεί να χρησιμοποιεί ως παραμέτρους το εισόδημα της οικογένειας του φοιτητή, τον αριθμό των μελών της οικογένειας, την απόδοση του φοιτητή καθώς και την περιοχή στην οποία σπουδάζει. Με βάση αυτά τα στοιχεία το κατηγορήμα κατατάσσει τον φοιτητή σε κάποια κατηγορία και έτσι υπολογίζεται το ποσό που πρέπει να πάρει. Για παράδειγμα το κατηγορήμα θα μπορούσε να έχει την παρακάτω μορφή :

A(FamilyMembers, Income, Grade, 'Athens', Money) :-  
B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 1.7.

A(FamilyMembers, Income, Grade, 'Patra', Money) :- B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 1.2.

A(FamilyMembers, Income, Grade, 'Larisa', Money) :- B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 3.7.

Αν όμως υπήρχε μια βασική απαίτηση προκειμένου να πάρει κανείς υποτροφία, όπως για παράδειγμα ο μέσος όρος (Grade) να είναι μεγαλύτερος του 7, τότε αυτός ο περιορισμός θα έπρεπε να καταγραφεί σε κάθε έκφραση του κατηγορήματος. Αυτό συχνά δεν είναι εύκολο και πολλές φορές οδηγεί σε δυσανάγνωστα κατηγορήματα. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε το κατηγορήμα ! σε συνδυασμό με το κατηγορήμα fail για να δηλώσουμε ότι αν ισχύει μια συγκεκριμένη συνθήκη τότε το κατηγορήμα σίγουρα πρέπει να αποτύχει. Για το κατηγορήμα A θα μπορούσαμε να κάνουμε την παρακάτω τροποποίηση η οποία δηλώνει ξεκάθαρα ότι πρώτα απ' όλα ο μέσος όρος πρέπει να είναι μεγαλύτερος του 7, διαφορετικά ο φοιτητής δεν παίρνει υποτροφία.

A(FamilyMembers, Income, Grade, City, Money) :- Grade < 7.0, !, fail.

A(FamilyMembers, Income, Grade, 'Athens', Money) :-  
B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 1.7.

A(FamilyMembers, Income, Grades, 'Patra', Money) :-  
B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 1.2.

A(FamilyMembers, Income, Grades, 'Larisa', Money) :-  
B(FamilyMembers, Income, X), C(Grade,Y), Money = X \* Y \* 3.7.

Αν ισχύει η συνθήκη  $\text{Grade} < 7.0$  τότε αποτιμάται το κατηγορήμα ! το οποίο και αποτρέπει την χρήση οποιουδήποτε εναλλακτικού τρόπου αποτίμησης του κατηγορήματος Α. Έτσι το Α εξαρτάται μόνο από τα κατηγορήματα που ακολουθούν το !. Όμως εδώ ακολουθεί το fail το οποίο πάντα αποτυγχάνει και έτσι το Α αποτυγχάνει χωρίς να εξεταστούν οι υπόλοιπες περιπτώσεις.

## ΠΑΡΑΔΕΙΓΜΑ ΑΠΟΚΟΠΗΣ - ΚΟΡΩΝΟΙΟΣ

Σήμερα αντιμετωπίζουμε το πρόβλημα του κορωνοιού. Σε πολλά νοσοκομεία για την καταστολή του δίνεται η χημική ουσία χλωροκίνη σε διάφορες δόσεις ανάλογα με, την ηλικία, τους παλμούς της καρδιάς και την πίεση του ασθενούς. Τα παρακάτω κατηγορήματα παρουσιάζουν την ποσότητα της χλωροκίνης που χορηγείται με βάση τα παραπάνω χαρακτηριστικά:

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 2.$

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 4.$

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 6.$

Στην προκειμένη περίπτωση όμως οι γιατροί μας λένε ότι για να δοθεί η συγκεκριμένη ουσία θα πρέπει η μικρή πίεση του ασθενούς πρέπει να είναι κάτω 8 και συνεπώς ο περιορισμός θα πρέπει να καταγραφεί σε κάθε έκφραση του κατηγορήματος. Κάτι τέτοιο θα οδηγήσει σε δυσανάγνωστα κατηγορήματα. Για τον λόγο αυτό θα χρησιμοποιήσουμε το κατηγορήμα ! σε συνδυασμό με το κατηγορήμα fail για να δηλώσουμε ότι αν ισχύει μια συγκεκριμένη συνθήκη τότε το κατηγορήμα σίγουρα πρέπει να αποτύχει δηλαδή δεν θα χορηγηθεί χλωροκίνη σε ασθενείς με μικρή πίεση κάτω από 8.

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- \text{Piesi} < 8.0, !, \text{fail}.$

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 2.$

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 4.$

$A(\text{Palmoi}, \text{Piesi}, \text{Ilikia}, \text{DosiChlorokinis}) :- B(\text{Palmoi}, \text{Piesi}, X), C(\text{Ilikia}, Y),$   
 $\text{DosiChlorokinis} = X + Y * 6.$

## ΠΡΟΓΡΑΜΜΑ ΦΑΡΜΑΚΕΥΤΙΚΗΣ ΑΓΩΓΗΣ

Το παρακάτω project έχει ως θέμα του την φαρμακευτική αγωγή που μπορεί να λάβει ένας ασθενής που υποφέρει από την καρδιά του. Οι στόχοι σας για το συγκεκριμένο παράδειγμα είναι:

1. Να μπορέσετε κατ' αρχήν να τρέξετε τις περιπτώσεις από goal1-goal12.
2. Να μπορέσετε να βρείτε τα όμοια αποτελέσματα
3. Να μπορέσετε να ερμηνεύσετε τον κάθε κανόνα
4. Να μπορέσετε να βρείτε σε ποια goals γίνεται backtracking\* και σε ποια goals το αποφεύγουμε\*\*.
5. Να μπορέσετε να ερμηνεύσετε το κάθε αποτέλεσμα

kardia(bivol).

kardia(lepur).

kardia(lasix).

kardia(acupron).

goal1:- kardia(X), write(X), nl, fail.

goal2:- kardia(X), !, write(X), nl, fail.

goal3:- kardia(X), write(X), nl, !, fail.

goal4:- !, kardia(X), write(X), nl, fail.

goal5:- kardia(X), write(X), nl, fail, !.

goal6:- kardia(X), write(X), nl.

goal7:- kardia(X), nl.

goal8:- kardia(X), write(X),!, nl.

goal9:- kardia(X), write(X), nl,!,

goal10:- !,kardia(X), write(X), nl.

goal11:- kardia(X).

goal12:- kardia(X), write(X).

\* backtracking με πολύ απλά λόγια είναι ο προς τα πίσω έλεγχος κλήσεων με σκοπό να βρεθούν όσο το δυνατό περισσότερα ορθά αποτελέσματα έστω και αν δεν υπάρχουν στην πραγματικότητα

\*\* Για να αποφύγουμε το backtracking χρησιμοποιούμε αποκοπή.

## ΑΣΚΗΣΗ ΜΕ ΥΠΟΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΤΩΝ

Δημιουργήστε αρχείο .pl με όνομα: το Επώνυμο σας2.pl, με λατινικούς χαρακτήρες στο οποίο γράψτε τον κώδικα της εφαρμογής, με βάση τις παρακάτω προδιαγραφές. Το αρχείο θα αποθηκευτεί εντός του φακέλου που δημιουργήσατε στο 1<sup>ο</sup> θέμα.

### Προδιαγραφές

Μία μηχανή παραγωγής ενός προϊόντος έχει 5 υποσυστήματα στα οποία ο κώδικας λειτουργίας τους μπορεί να είναι είτε σε άριστη, είτε σε καλή, είτε σε μέτρια είτε σε κακή κατάσταση. Έχει κατασκευαστεί λογισμικό που ανά πάσα στιγμή ειδοποιεί το κεντρικό χειριστή της μηχανής για την κατάσταση του κάθε υποσυστήματος. Την Τρίτη 7-6-2016 λήφθηκαν τα ακόλουθα αποτελέσματα:

Υποσύστημα	Κατάσταση Κώδικα	Βαθμός
ypos1	καλή	7
ypos2	άριστη	10
ypos3	μέτρια	6
ypos4	καλή	7
ypos5	κακή	4

### Για τα αποτελέσματα αυτά:

1. Δημιουργήστε λίστα με τους ανάλογους κανόνες (εκφράσεις). Χρησιμοποιήστε τις λέξεις **yposistema** και **katastasi**. Για κάθε υποσύστημα από τα 5 χρησιμοποιήστε την λέξη ypos1 κ.λπ. Όλα τα γεγονότα θα δημιουργηθούν με βάση τον πίνακα. Μπορείτε να χρησιμοποιήσετε είτε την 1<sup>η</sup> μαζί με την 2<sup>η</sup> στήλη, είτε την 1<sup>η</sup> μαζί με την 3<sup>η</sup> στήλη ή όλες μαζί.
2. Δημιουργήστε κατηγορία που θα αφαιρεί από την λίστα όσους κώδικες βρίσκονται σε κακή κατάσταση. Για δική σας διευκόλυνση χρησιμοποιήστε την 1<sup>η</sup> και 3<sup>η</sup> στήλη για αυτό.
3. Δημιουργήστε κατηγορία με την βοήθεια του append που θα αντικαθιστά τους κώδικες που βρίσκονται σε κακή κατάσταση με άλλους που βρίσκονται σε άριστη και θα επιστρέφει μία νέα λίστα. Χρησιμοποιήστε την 2<sup>η</sup> στήλη του πίνακα για αυτό και λατινικούς χαρακτήρες.

### Στην συνέχεια:

#### Δημιουργήστε τα ακόλουθα ερωτήματα με βάση τον παραπάνω πίνακα:

1. Ερώτημα από το οποίο θα προκύπτουν τα υποσυστήματα που βρίσκονται σε καλή κατάσταση.
2. Ερώτημα με το οποίο θα αφαιρούνται οι κώδικες των υποσυστημάτων που βρίσκονται σε κακή κατάσταση δηλαδή με βαθμό κάτω από 5. Χρησιμοποιήστε κατηγορία με όνομα enresi.
3. Ερώτημα με το οποίο θα αντικαθιστώνται οι κώδικες που βρίσκονται σε κακή κατάσταση από άλλους που βρίσκονται σε άριστη. Χρησιμοποιήστε κατηγορία με όνομα change.

Γράψτε, εκτελέστε (για δική σας επαλήθευση) και αποθηκεύστε και τα 3 ερωτήματα σε ένα νέο αρχείο txt με όνομα Επώνυμο σας3.txt. στον φάκελο που δημιουργήσατε.

```
yposistema([],[]).  
yposistema([H|T], [H1|T1]) :-katastasi(H,H1), yposistema(T,T1).
```

```
katastasi(ypos1,'kali').  
katastasi(ypos2, 'metria').  
katastasi(ypos3, 'kali').  
katastasi(ypos4, 'kali').  
katastasi(ypos5, 'kaki').
```

```
change(kaki,aristi,PL,NL) :-append(A,[kaki|B],PL), append(A,[aristi|B],NL).
```

Δημιουργήστε τα ακόλουθα ερωτήματα:

1. Ερώτημα από το οποίο να προκύπτουν τα υποσυστήματα που βρίσκονται σε καλή κατάσταση

**katastasi(X,'kali').**

2. Ερώτημα με το οποίο θα αντικαθιστώνται οι κώδικες που βρίσκονται σε κακή κατάσταση από άλλους που βρίσκονται σε άριστη κατάσταση. Από την εκτέλεση του συγκεκριμένου ερωτήματος αυτού θα προκύπτει νέα λίστα

**change(kaki, aristi,[kali,metria,kali,kali,kaki],X).**

```

agwnas(barc,real,[2,2]).
agwnas(barc,pao,[3,1]).
agwnas(barc,oly,[0,0]).
agwnas(pao,barc,[4,2]).
agwnas(pao,oly,[1,0]).
agwnas(pao,real,[0,3]).
agwnas(real,oly,[0,1]).
agwnas(real,barc,[1,6]).
agwnas(real,pao,[3,0]).
agwnas(oly,barc,[0,4]).
agwnas(oly,real,[3,3]).
agwnas(oly,pao,[2,0]).

```

```

omada('pao').
omada('oly').
omada('barc').
omada('real').

```

```

apanthsh('nikh', 0).
apanthsh('isopalia', 1).
apanthsh('htta', 2).

```

go:-

```

write('ti thelete na mathete?(nikh,isopalia,htta)\n'),nl,
read(Input),nl,
apanthsh(Input, Output),

```

(Output>1->

```

write('Ti ita omada thes?'),nl,
read(Apan), nl,
write('Httes omadas Entos: '), httaEN(Apan), nl,

```



```
write('Httes omadas ekstos: '), httpaEK(Apan), nl  
;
```

Output>0->

```
write('Ti iso omada thes?'),nl,  
read(Apan), nl,  
write('Isopalia Entos: '), isopaliaEN(Apan), nl,  
write('Isopalia Ektos: '), isopaliaEK(Apan), nl  
;  
write('Ti niki omada thes?'),nl,  
read(Apan), nl,  
nikhEN(Apan), nl,  
nikhEK(Apan), nl).
```

nikhEN(X):- omada(X),omada(Y),agwnas(X,Y,[Z,N]),Z>N.

nikhEK(X):-omada(Y),omada(X),agwnas(Y,X,[Z,N]),Z<N.

httpaEN(X):-omada(X),omada(Y),agwnas(X,Y,[Z,N]),Z<N.

httpaEK(X):-omada(Y),omada(X),agwnas(Y,X,[Z,N]),Z>N.

isopaliaEN(X):-omada(X),omada(Y),agwnas(X,Y,[Z,N]),Z=N.

isopaliaEK(X):-omada(X),omada(Y),agwnas(Y,X,[Z,N]),Z=N.

nikes(X):-omada(X),nikhEN(X);nikhEK(X).

httes(X):-omada(X),httpaEN(X);httpaEK(X).

isopalies(X):-omada(X),isopaliaEN(X);isopaliaEK(X).

**5 ΜΑΙΟΥ 2020**

**ΑΣΚΗΣΕΙΣ ΜΕ MATCHING, ΛΙΣΤΕΣ, APPEND, REVERSE, LENGTH**

1. Δημιουργείτε μία δομή δεδομένων (όρισμα μέσα σε κατηγορημα) μέσα σε ένα fact - κατηγορημα όπου θα εμφανίζεται η ημερομηνία γέννησης ενός ανθρώπου.
2. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε μία γραμμή.
3. Στην συνέχεια κάντε matching ημερομηνιών γεννήσεως και ανθρώπων σε πολλές γραμμές.
4. Κατόπιν βρείτε αυτούς που γεννήθηκαν μία συγκεκριμένη χρονολογία.
5. Κατασκευάστε υπό μορφή λίστας ένα fact – κατηγορημα στο οποίο θα αποτυπώνονται ο πατέρας και οι τέσσερις του γιοι.
6. Δημιουργείτε ένα ερώτημα όπου θα γίνεται matching του πατέρα και των τεσσάρων γιών του που να εμφανίζεται η κεφαλή και η ουρά της λίστας. Να σημειωθεί ότι η συγκεκριμένη εμφάνιση είναι ΑΝΑΔΡΟΜΗ γιατί;
7. Δημιουργήστε μία ερώτηση σε prolog όπου θα εξετάσετε αν κάποιο στοιχείο είναι μέλος της λίστας.
8. Δημιουργήστε μία ερώτηση σε prolog όπου θα αποτυπώνονται όλα τα στοιχεία μίας λίστας .
9. Να ορισθεί αναδρομικά το member δηλαδή:
  - Ο πρώτος κανόνας (true) θα είναι ο κανόνας τερματισμού δηλαδή ότι αληθεύει το member όταν το πρώτο όρισμα είναι το X και το δεύτερο όρισμα είναι μία λίστα με κεφαλή X και τα υπόλοιπα στοιχεία οποιαδήποτε. (αληθεύει το member αν το X είναι το πρώτο στοιχείο της λίστας.

- Ο δεύτερος κανόνας θα είναι ο αναδρομικός (false) όπου αν το στοιχείο X ανήκει στην λίστα L τότε το member αληθεύει . ΑΝΑΔΡΟΜΗ.

10. Δημιουργήστε ένα fact – append όπου θα ενώνονται οι λίστες L1, L2 και θα σχηματίζεται η L.

11. Δημιουργήστε ένα fact-append στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την πρόσθεση μίας λίστας σε μία άλλη από αριστερά.

12. Δημιουργήστε ένα fact-append στο οποίο θέλετε να πάρετε μία τελική λίστα η οποία θα προκύπτει από την συνένωση δύο λιστών με όλους τους δυνατούς τρόπους.

13. Να ορισθεί αναδρομικά το append δηλαδή:

- Θα κατασκευάσετε έναν κανόνα τερματισμού όπου θα συνενώνουμε μία κενή λίστα με μία άλλη L και το αποτέλεσμα θα είναι η L.
- Θα κατασκευάσετε ένα αναδρομικό κανόνα όπου θα συνενώνεται μία λίστα L1 με την L2 και θα προκύπτει η λίστα L3 με κεφαλή το X και ουρά μία λίστα L.

14. Δημιουργήστε ένα ερώτημα στο οποίο θα γίνεται η καταμέτρηση των στοιχείων της λίστας.

15. Δημιουργήστε ένα ερώτημα με το οποίο θα γίνεται η αντιστροφή μίας λίστας.

born(panos,date(23,8,1990)).

born(maria,date(24,8,1990)).

born(panos,X). // matching se 1 grammh

born(panos,date(X,A,B)). // matching se polles grammes

born(X,date(\_,\_,1990)).// olles oi hmeromhnies 1990

children(a,[b,c,d,e]).// o a einai pateras kai exei paidia to b,c,d,e

children(a,[A|B]). // matching me anadromh(6)

member(1,[1,2,3,4]) //erwthsh(7)

member(X,[X|\_]). //to x einai prwto stoixeio ths listas (9)

member(X,[\_|L]):-member(X,L). /// to x einai meros tou listas

append([],[],L). (10)

append(L,[6,7],[4,6,7]).// apo aristera(11)

append([],L,L).// erwthsh (13)

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

## ΑΣΚΗΣΗ - ΔΙΑΓΝΩΣΤΙΚΟ

Σχεδιάστε μία διαγνωστική εφαρμογή για τον εντοπισμό βλαβών σε έναν Η/Υ όπου:

1. Όταν η οθόνη παρουσιάζει παραμορφώσεις και μαυρίζει τότε υπάρχει βλάβη στην κάρτα γραφικών.
2. Όταν η οθόνη μαυρίζει τότε η βλάβη είναι καμένη οθόνη.
3. Όταν ο υπολογιστής κάνει επανεκκινήσεις ή κλείνει απότομα τότε υπάρχει ιός.
4. Όταν κλείνει τότε υπάρχει υπερθέρμανση.
5. Όταν κολλάει τότε υπάρχει βλάβη στον σκληρό δίσκο

Στις 11 Απριλίου 2020 η εταιρεία επισκευής παρέλαβε τους εξής υπολογιστές με τα ακόλουθα προβλήματα:

simptoma(pc1,paramorfosi).

simptoma(pc1,mavro).

simptoma(pc2,mavro).

simptoma(pc3,epanekinei).

simptoma(pc4,kleinei).

simptoma(pc5,kollaei).

simptoma(pc6,paramorfosi).

simptoma(pc7,kollaei).

simptoma(pc7,kleinei).

simptoma(pc8,epanekinei).

simptoma(pc8,mavro).

### Ερωτήσεις:

1. Γράψτε τους αντίστοιχους κανόνες για την παραπάνω λεκτική περιγραφή.
2. Εντοπίστε αν υπάρχουν pc που έχουν πρόβλημα με την κάρτα γραφικών.
3. Τι βλάβη έχει το pc2;
4. Εντοπίστε αν υπάρχουν pc που έχουν ιό.

#### Άσκηση 1 (14-Apr-20)

**vlavi(X,karta\_grafikwn):-simptoma(X,paramorfosi),simptoma(X,mavro).**

**vlavi(X,kameni\_othoni):-simptoma(X,mavro).**

**vlavi(X,ios):-simptoma(X,eppanekinisi);simptoma(X,kleinei). (((!!!)))//Χρήση του cut.**

**vlavi(X,uperthermansi):-simptoma(X,kleinei).**

**vlavi(X,skliros):-simptoma(X,kollaei).**

**Ποιό pc έχει πρόβλημα με την gru?**

**vlavi(x,karta grafikwn).**

**Τι βλάβη έχει το pc2?**

**vlavi(pc2,X)**

**Υπάρχουν pc που εχουν ιο?**

**vlavi(X,ios).**

### Άσκηση 1

Ένας ηλεκτρονικός υπολογιστής έχει ένα πλήθος επικίνδυνων διασκορπισμένων αρχείων στον δίσκο του, που ολόκληρος είναι ένας ενιαίος φάκελος. Θεωρώντας ότι τα ονόματα αυτών των αρχείων είναι ίδια (trojan) και όλα μαζί τα αρχεία αποτελούν λίστα ονομάτων, να δημιουργήσετε πρόγραμμα σε prolog που να αφαιρεί από τη λίστα των ονομάτων, τα αρχεία με όνομα trojan.

Δοκιμάστε το πρόγραμμά σας με τη λίστα [f1, trojan, d, trojan, f2]).

### Άσκηση 2

Μία ηλεκτρική συσκευή αποτελείται από 5 διαφορετικά εξαρτήματα  $p_1, p_2, p_3, p_4, p_5$  τα οποία βρίσκονται στην κεντρική πλακέτα της και λειτουργούν το καθένα ξεχωριστά με βάση συγκεκριμένο λογισμικό  $s_1, s_2 \dots s_5$ . Το κάθε εξάρτημα δηλαδή έχει το δικό του λογισμικό το οποίο όμως περιοδικά χρειάζεται αντικατάσταση με αναβάθμισή του σε νεότερο  $n_i$  με  $i=1,2..5$ . Την αντικατάσταση αναλαμβάνει διαγνωστικό σύστημα που ελέγχει τα λογισμικά και αντικαθιστά το παρωχημένο  $s_i$  με νεότερο  $n_i$  και αυτό γίνεται με κατάλληλο πρόγραμμα της prolog όταν συνδέσουμε τη συσκευή στο διαγνωστικό σύστημα.

Γράψτε το πρόγραμμα του διαγνωστικού συστήματος που υλοποιεί την αντικατάσταση του λογισμικού, συγκεκριμένου εξαρτήματος, με το αναβαθμισμένο νεότερο λογισμικό. Κάντε τις σχετικές δοκιμές.

### Άσκηση 3

Υποθέτουμε ότι ο covid-19 αποτελείται από 5 διαφορετικά είδη μικροβίων, των οποίων ο πληθυσμός του κάθε είδους καταχωρείται ημερήσια σε κατάλληλη λίστα, με συγκεκριμένη καθορισμένη σειρά. Η συνάρτηση αξιολόγησης της διάγνωσης του ασθενούς είναι της μορφής  $2*\alpha_i - 3*\beta_i$ , όπου  $\alpha_i$  ο πληθυσμός του  $i$  είδους μικροβίου την πρώτη ημέρα και  $\beta_i$  ο αντίστοιχος πληθυσμός την δεύτερη ημέρα. Οι γιατροί βλέπουν τον πίνακα μικροβίων πώς διαμορφώνεται μετά τη δεύτερη ημέρα και αν ο πληθυσμός οποιουδήποτε από τα 5 μικρόβια είναι θετικός αποφαινόνται ότι εξεταζόμενος άνθρωπος είναι θετικός στον

covid-19. Στην περίπτωση όπου όλοι οι αριθμοί είναι αρνητικοί ή μηδέν τότε αποφαίνονται ότι είναι αρνητικός.

Διαμορφώστε κατάλληλο πρόγραμμα σε prolog που να κάνει τη σχετική διάγνωση. Δοκιμάστε το πρόγραμμά σας καταχωρώντας σε δυο λίστες διάφορους πληθυσμούς μικροβίων και ελέγξτε τα αποτελέσματα που προκύπτουν από τη συνάρτηση αξιολόγησης σε τρίτη λίστα.

#### Άσκηση 4

Όσοι ασχοληθείτε με την prolog σε επαγγελματικό επίπεδο θα συναντήσετε το παραθυρικό περιβάλλον της prolog όπου ένα από τα πιο χρήσιμα προγράμματα είναι η δημιουργία back-up αρχείων. Ο Στρατός η Αστυνομία και οι Δικαστικές αρχές έχουν προγράμματα που δημιουργούν κλώνους (δηλαδή ίδια αρχεία). Στα πλαίσια αυτού του μαθήματος ζητείται να δημιουργήσετε στο περιβάλλον SWI-PROLOG ένα project όπου δίπλα σε κάθε στοιχείο μίας λίστας να εμφανίζεται ένα όμοιο του, να δημιουργεί δηλαδή δεύτερη λίστα στην οποία θα εμφανίζονται δύο φορές τα στοιχεία της πρώτης.

#### Άσκηση 5

Η prolog δίνει την δυνατότητα να αντιστοιχίζονται κάποιες μεταβλητές ή κάποιες σταθερές με κάποιες άλλες. Μία πολύ καλή εφαρμογή θα μπορούσε να ήταν ένα λεξικό όρων – μεταφραστικό όπου μία ελληνική λέξη αντιστοιχίζεται σε μία αγγλική και αντίστροφα. Αυτό μπορεί να αποτελεί τη στοιχειώδη γνωσιακή βάση του συστήματος.

Να αναπτύξετε ένα απλό πρόγραμμα σε prolog στο οποίο να δίνεται σειρά λέξεων στα ελληνικά και το πρόγραμμα να αποδίδει τη σειρά των λέξεων στα αγγλικά και αντίστροφα.

ask1

**afairw( \_, [], []):-!.**

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

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



afairesh me cut !

ask2

E=palio stoixeio

NE=kainourgio stoixeio

L=uparxon lista

NL=kainourgia lista

**change(E,NE,L,NL):-append(A,[E | B],L), append(A,[NE | B],NL).**

ask3

**addlist([],[],[]).**

**addlist([H1 | T1],[H2 | T2],[H3 | T3]):-H3 is 2\*H1-3\*H2,addlist(T1,T2,T3).**

ask4

**double([],[]):-!.**

**double([H | J],[H,H | D]):-double(J,D).**

ask5

**a([],[]).**

**a([H | T],[H1 | T1]):-b(H,H1),a(T,T1).**

**a(L1,L2):-a(L2,L1).**

**b('house','σπίτι').**

**b('is','είναι').**



## ΑΣΚΗΣΗ ΙΚΑ

Δίνεται ο παρακάτω πίνακας που αφορά τους υπαλλήλους (employees) και τους διευθυντές (managers) μίας εταιρείας.

Όνομα	Ηλικία age	Προϋπηρεσία service	Ειδικότητα
petros	60	10	shopfloor
Iarian	55	5	manager
kostas	57	26	shopfloor
na8anail	60	9	shopfloor
iakovos	49	24	shopfloor
ilarios	56	8	manager
panos	58	30	shopfloor

Οι υπάλληλοι που πληρούν τις παρακάτω προϋποθέσεις δικαιούνται σύνταξης (syntaxi) και η εταιρεία μπορεί να τους ενημερώσει για αυτό.

### Προϋποθέσεις:

1. Ικανός για λήψη σύνταξης είναι ο υπάλληλος που έχει προϋπηρεσία πάνω από 20 έτη (Years).
2. Δεν δικαιούται σύνταξης υπάλληλος που η ηλικία του είναι κάτω από 50 έτη
3. Δεν δικαιούται σύνταξης από το ΙΚΑ ένας manager όμως δικαιούται ένας shopfloor.

Από τον παρακάτω πίνακα προκύπτουν τα ακόλουθα facts

age(petros,60).

age(Iarian,55).

age(kostas,57).

age(na8anail,60).

age(iakovos,49).

age(ilarios,56).

age(panos,58).

service(kostas,26).

service(petros,10).

service(na8anail,19).

service(iakovos,24).

service(larian,5).

service(ilarios, 8).

service(panos,30).

manager(larian).

manager(ilarios).

shopfloor(petros).

shopfloor(iakovos).

shopfloor(na8anail).

shopfloor(kostas).

shopfloor(panos).

Δημιουργήστε τους κατάλληλους κανόνες για ένα project το οποίο θα αποθηκεύσετε ως **ika.pl** και στην συνέχεια βρείτε ποιοι υπάλληλοι με βάση τον πίνακα είναι ικανοί προς συνταξιοδότηση.

### Άσκηση 2 (14-Apr-20)

**sintaxi(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).

**Ποιοι εργαζόμενοι δικαιούνται συνταξη?**

sintaxi(Employee).

paragogos(\*(C,X),X,C).

paragogos(X,X,1).

paragogos(e(X),X,e(X)).

paragogos(+(F,G),X,+(DF,DG)) :- paragogos(F,X,DF), paragogos(G,X,DG).

paragogos(-(F,G),X,-(DF,DG)) :- paragogos(F,X,DF), paragogos(G,X,DG).

paragogos(-(F),X,-(DF)):- paragogos(F,X,DF).

paragogos(\*(F,G),X,(\*(F,DG),\*(DF,G))) :- paragogos(F,X,DF), paragogos(G,X,DG).

paragogos(/(F,G),X,/(-(G,DF),\*(F,DG)),^(G,2))) :- paragogos(F,X,DF), paragogos(G,X,DG).

paragogos(sin(X),X,cos(X)).

paragogos(cos(X),X,-(sin(X))).

paragogos(ln(X),X,/(1,X)).

```
connect(arad,sibiu).
connect(sibiu,rimnicu).
connect(rimnicu,pitesti).
connect(pitesti,bucharest).
connect(arad,zerind).
connect(zerind,oradea).
connect(oradea,sibiu).
connect(sibiu,fagaras).
connect(fagaras,bucharest).
connect(arad,timisoara).
connect(timisoara,lugoj).
connect(lugoj,mehadia).
connect(mehadia,drobeta).
connect(drobeta,craiova).
connect(craiova,pitesti).
connection(X,Y):-connect(X,Z), connect(Z,Y).
route(X,Y):-connect(X,Y).
route(X,Y):-connect(X,Z),route(Z,Y).
```

teacher(sinatkas).

teacher(kranas).

teacher(mpatos).

student(papadopoulos).

student(georgiadis).

student(petridis).

teaching(sinatkas, tnlp\_th).

teaching(sinatkas, tnlp\_et1).

teaching(sinatkas, dd\_th).

teaching(kranas, tnlp\_et2).

teaching(kranas, diad\_ef\_et1).

teaching(mpatos, diad\_ef\_et2).

attending(papadopoulos, tnlp\_et1).

attending(papadopoulos, dd\_th).

attending(papadopoulos, diad\_ef\_et1).

attending(georgiadis, tnlp\_et2).

attending(georgiadis, dd\_th).

attending(georgiadis, diad\_ef\_et2).

attending(petridis, tnlp\_th).

attending(petridis, tnlp\_et1).

teachsubject(tnlp, sinatkas).

teachsubject(tnlp, kranas).

teachsubject(dd, sinatkas).

teachsubject(diad\_ef, kranas).

teachsubject(diad\_ef, mpatos).

watchsubject(papadopoulos, tnlp).



watchsubject(papadopoulos, dwmes).

watchsubject(papadopoulos, diad\_ef).

watchsubject(georgiadis, tnlp).

watchsubject(georgiadis, dd).

watchsubject(georgiadis, diad\_ef).

watchsubject(petridis, tnlp).

partners(X,Y) :- teachsubject(A,X), teachsubject(A,Y), \+ X==Y.

costudents(X,Y) :- attending(X,A), attending(Y,A), \+ X==Y.

teachstudent(X,Y) :- teaching(X,A), attending(Y,A), \+ X==Y.

male(kiriakos).

male(kostas).

female(katerina).

female(helen).

female(mary).

married(kiriakos,katerina).

married(helen,kostas).

married(mary,katerina).

parent(kiriakos,helen).

parent(katerina,helen).

parent(helen,katerina2).

parent(kostas,mary).

parent(helen,mary).

parent(kostas,katerina2).

father(Fa,Ch) :- parent(Fa,Ch), male(Fa).

mother(Mo,Ch) :- parent(Mo,Ch), female(Mo).

ancestor(X,Anc) :- parent(Anc,X), parent(\_\_\_\_,Anc).

siblings(Ch1, Ch2) :- father(Fa, Ch1), father(Fa, Ch2), mother(Ma,Ch1), mother(Ma,Ch2), Ch1 \= Ch2.

grandchildren(X,Gch) :- parent(X,Y),parent(Y,Gch).

