ΑΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΕΡΓΑΣΤΗΡΙΟ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ (Prolog)
ΕΞΑΜΗΝΟ: Δ΄- Εαρινό 2014-15
ΚΑΘΗΓΗΤΕΣ: Δ.ΣΤΑΜΑΤΗΣ, Κ.ΔΙΑΜΑΝΤΑΡΑΣ

ΑΣΚΗΣΗ 2:
ΠΕΡΙΓΡΑΦΗ ΕΝΟΣ ΨΗΦΙΑΚΟΥ ΚΥΚΛΩΜΑΤΟΣ

Η άσκηση αυτή πραγματεύεται την αναπαράσταση και λειτουργία ψηφιακών κυκλωμάτων. Στην Prolog, σε αντίθεση με άλλες γλώσσες όπως η Java ή η C/C++, δεν υπάρχουν μέθοδοι ή συναρτήσεις αλλά **κατηγορήματα**. Η βασική διαφορά είναι ότι τα κατηγορήματα δεν επιστρέφουν τιμή ενός οποιουδήποτε τύπου (ή void) στην έξοδο. Τα κατηγορήματα είτε ικανοποιούνται, οπότε επιστρέφουν την απάντηση Yes (true), είτε δεν ικανοποιούνται οπότε επιστρέφουν την απάντηση No (false). Στην C για παράδειγμα, θα μπορούσαμε να υλοποιήσουμε το λογικό ΚΑΙ (AND) με μια συνάρτηση `and` που θα έπαιρνε δύο εισόδους A, B και θα επέστρεφε την έξοδο Y:

```
/* Κώδικας C */
int and(int A, int B)
{
   int Y;
   if (A==1 && B==1) Y=1;
   else Y=0;
   return Y;
}
```

Το αντίστοιχο κατηγόρημα της Prolog θα παίρνει τρία ορίσματα, A, B, Y, καθότι το Y δεν επιστρέφεται στην έξοδο όπως γίνεται στη C. Η μόνη λύση είναι να γίνει και αυτό ένα όρισμα μαζί με τις εισόδους A και B. Έτσι έχουμε το κατηγόρημα `and(A,B,Y)` το οποίο περιγράφεται από ένα σύνολο γεγονότων που μας δίνουν τον πίνακα αληθείας της πύλης AND:

```
and(0,0,0).
and(0,1,0).
and(1,0,0).
and(1,1,1).
```

Αν θέλουμε να μάθουμε την έξοδο της πύλης AND για εισόδους A=1, B=0 δεν έχουμε παρά να κάνουμε την ερώτηση:

```
?- and(1,0,Y).
Y = 0
```

Όμως συμβαίνει και κάτι ακόμα. Η Prolog δε γνωρίζει ότι το τρίτο όρισμα είναι η έξοδος της πύλης AND και τα δύο πρώτα είναι οι είσοδοι. Αυτή είναι μια δική μας σύμβαση και η γλώσσα ούτε το ξέρει αλλά ούτε και ενδιαφέρεται. Γι' αυτήν οι τρεις είσοδοι είναι μεταβλητές που σχετίζονται μεταξύ τους με τη σχέση (κατηγόρημα) `and` τις οποίες μεταχειρίζεται με τον ίδιο τρόπο. Έτσι όπως τα δύο πρώτα ορίσματα μπορούν να είναι γνωστά και το τρίτο άγνωστο, (βλέπε π.χ. την ερώτηση που κάναμε παραπάνω) θα μπορούσε το τρίτο όρισμα να ήταν γνωστό και τα δύο πρώτα άγνωστα, π.χ.

```
?- and(A,B,0).
```

Η παραπάνω ερώτηση ζητάει να μάθει ποιες είσοδοι θα μας δώσουν έξοδο Y=0. Η απάντηση που δίνει η Prolog είναι:

```
A=0
B=0;

A=0
B=1;

A=1
B=0;

No
```

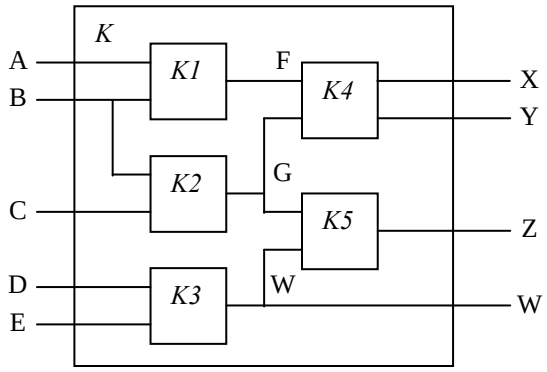
Η ευελιξία της Prolog στην διαχείριση των μεταβλητών που τίθενται ως παράμετροι άλλοτε ως εισόδους και άλλοτε ως εξόδους είναι κάτι που δεν υπάρχει σε άλλες γλώσσες όπως η C ή η Java. Για παράδειγμα, στην C η εντολή 0=and(A,B) θα προκαλούσε συντακτικό λάθος κατά τη μετάφραση (compilation).

Φυσικά μπορούμε να κάνουμε ερωτήσεις και με άλλους συνδυασμούς γνωστών και αγνώστων μεταβλητών. Πχ.

```
?- and(A,1,0).      % η δεύτερη είσοδος και η έξοδος γνωστές
?- and(0,B,1).      % η πρώτη είσοδος και η έξοδος γνωστές
?- and(A,B,Y).      % τίποτα γνωστό: δώσε τον πίνακα αληθείας
```

Περιγραφή πιο πολύπλοκων κυκλωμάτων

Όταν το κύκλωμα είναι πιο πολύπλοκο τότε ο πίνακας αληθείας μπορεί να είναι πιο μεγάλος οπότε είναι πολύ πιο κοπιαστική η καταχώρησή του με ένα σύνολο γεγονότων. Αν όμως το κύκλωμα αποτελείται από άλλα απλούστερα κυκλώματα τότε μπορούμε να το περιγράψουμε με κανόνες. Για παράδειγμα θεωρήστε το κύκλωμα *K* που έχει 5 εισόδους (A, B, C, D, E) και 4 εξόδους (X, Y, X, W) όπως φαίνεται στο παρακάτω σχήμα



Το κύκλωμα *K* αποτελείται από τα επί μέρους κυκλώματα *K1*, *K2*, *K3*, *K4*, *K5*. Αν υποθέσουμε ότι έχουμε υλοποιήσει τα αντίστοιχα κατηγορήματα γι' αυτά τα απλούστερα κυκλώματα θα μπορούσαμε να υλοποιήσουμε το κύκλωμα K με τον παρακάτω κανόνα:

```
k(A,B,C,D,E,X,Y,Z,W) :-
      k1(A,B,F),
      k2(B,C,G),
      k3(D,E,W),
      k4(F,G,X,Y),
      k5(G,W,Z).
```

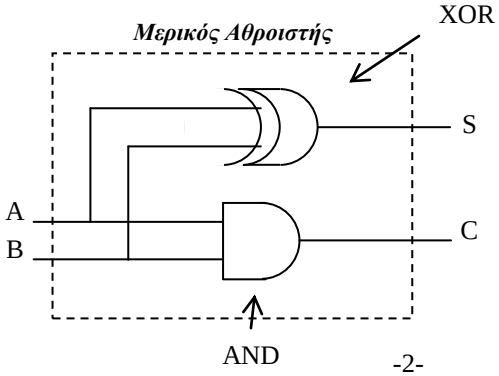
Κάθε κατηγόρημα έχει τόσα ορίσματα όσο το άθροισμα του αριθμού των εισόδων και του αριθμού των εξόδων του αντίστοιχου κυκλώματος. Έτσι *πχ.* το κύκλωμα *K* περιγράφεται από το κατηγόρημα `k()` με 9 ορίσματα (5είσοδοι + 4έξοδοι) ενώ το κύκλωμα *K3* περιγράφεται από το κατηγόρημα `k3()` με 3 ορίσματα (2 είσοδοι + 1 έξοδος).

Από τη στιγμή που θα ορίσουμε το κύκλωμα `k` όπως παραπάνω μπορούμε να το χρησιμοποιήσουμε για την υλοποίηση πιο σύνθετων κυκλωμάτων κ.ο.κ.

ΤΙ ΠΡΕΠΕΙ ΝΑ ΚΑΝΕΤΕ

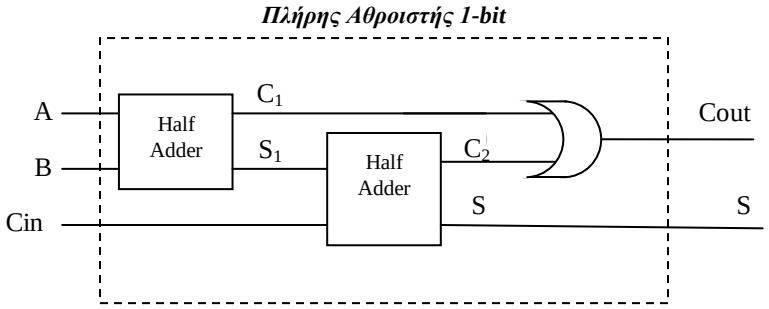
(α) Υλοποιήστε τις πύλες AND, OR, XOR με κατάλληλα κατηγορήματα.

(β) Υλοποιήστε το κύκλωμα του μερικού αθροιστή (Half-Adder) που δίνεται παρακάτω



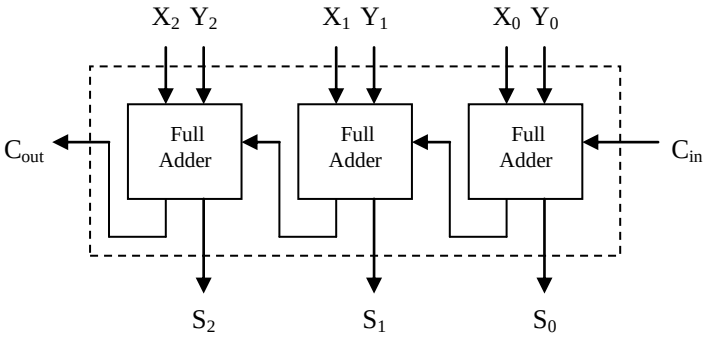
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(γ) Υλοποιήστε το κύκλωμα του πλήρους αθροιστή 1-bit (Full Adder) που φαίνεται παρακάτω



A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(δ) Υλοποιήστε τον αθροιστή 3-bit το κύκλωμα του οποίου δίνεται παρακάτω. Το κύκλωμα αυτό προσθέτει δύο τριψήφιους δυαδικούς αριθμούς X2_X1_X0 και Y2_Y1_Y0 έχοντας ως Cin πιθανό κρατούμενο από προηγούμενη πράξη και δίνει ως αποτέλεσμα ένα τριψήφιο δυαδικό S2_S1_S0 και πιθανό κρατούμενο για επόμενη πράξη το Cout.



	X2	X1	X0	
+	Y2	Y1	Y0	Cin
Cout	S2	S1	S0	

	1	0	1	
+	1	1	0	0
Cout	S2	S1	S0	

Δοκιμάστε να κάνετε την δίπλα πρόσθεση με τη χρήση του κατηγορήματος σας.

(ε) Ποια είναι τα αποτελέσματα για τις παρακάτω πράξεις, όπως μπορούν να βρεθούν με τη χρήση του παραπάνω κατηγορήματός σας; Τι κάνει πλέον το κατηγόρημά σας;

	X2	X1	X0	
+	1	1	0	Cin
1	0	1	1	

Απαντήσεις:

	X2	X1	X0	
+	Y2	Y1	Y0	Cin
1	0	1	1	

Απαντήσεις: