ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2022 (17/6/2022)

1. (**15/60**) Έστω ότι έχουμε ορίσει ένα κατηγόρημα `solveprob/1`, το οποίο με βάση κάποια δεδομένα που του δίνονται (π.χ. μέσω γεγονότων ή με ανάγνωση της εισόδου) επιλύει ένα πρόβλημα και επιστρέφει στο όρισμά του μία λύση του προβλήματος, σαν μία λίστα στοιχείων. Το κατηγόρημα είναι σε θέση να επιστρέψει μέσω οπισθοδρόμησης όλες τις λύσεις του προβλήματος. Ορίζουμε ότι μία λύση του προβλήματος είναι βέλτιστη όταν το πλήθος των στοιχείων της είναι ελάχιστο. Η υλοποίηση του κατηγορήματος δεν μας ενδιαφέρει, μπορεί να είναι και εξαιρετικά πολύπλοκη, αλλά, για λόγους απλότητας, ας θεωρήσουμε ότι το κατηγόρημα είναι υλοποιημένο με ένα σύνολο γεγονότων όπως φαίνεται στη συνέχεια.

```
solveprob([a,b,c,d]).          solveprob([k,m]).          solveprob([n,a,z,t,r,g,h]).
solveprob([k,r,w]).            solveprob([p,u]).          solveprob([r,e]).
solveprob([o,e,n,m]).          solveprob([b,o,k]).        solveprob([w,w]).
```

Συμπληρώστε κατάλληλα τα ..... στις δύο παρακάτω ερωτήσεις, έτσι ώστε η μεν πρώτη να εκτυπώνει την πρώτη βέλτιστη λύση του προβλήματος και η δεύτερη να εκτυπώνει όλες τις βέλτιστες λύσεις. Σε κάθε περίπτωση, δεν απαιτείται, και δεν πρέπει, να υλοποιήσετε επιπλέον κατηγορήματα. Φυσικά, στα κενά που θα συμπληρώσετε, μπορείτε να χρησιμοποιήσετε συνήθη ενσωματωμένα κατηγορήματα της Prolog.

```
?- length(.....), solveprob(.....), ..... .
[k, m]
No

?- length(.....), solveprob(.....), ..... .
[k, m]
[p, u]
[r, e]
[w, w]
No
```

2. (**10/60**) Μία εταιρεία που λειτουργεί 24 ώρες την ημέρα και 7 ημέρες την εβδομάδα απασχολεί 50 υπαλλήλους. Προφανώς, οι υπάλληλοι πρέπει να εργάζονται σε βάρδιες. Κάθε υπάλληλος μπορεί, για κάποια ημέρα, να εργάζεται σε πρωινή βάρδια (M), ή απογευματινή (E), ή νυχτερινή (N), ή μπορεί να έχει ανάπαυση (R). Η εταιρεία θέλει να δημιουργήσει ένα πρόγραμμα εργασίας των υπαλλήλων της για ένα διάστημα 30 συνεχόμενων ημερών, στο οποίο όμως να ισχύουν οι εξής κανόνες:

- Κάθε ημέρα πρέπει να εργάζονται τουλάχιστον 15 υπάλληλοι στην πρωινή βάρδια, τουλάχιστον 10 στην απογευματινή και τουλάχιστον 8 στη νυχτερινή.
- Κανένας υπάλληλος δεν επιτρέπεται αν μία ημέρα είχε νυχτερινή βάρδια, την επόμενη να έχει πρωινή.
- Κάθε υπάλληλος πρέπει σε οποιεσδήποτε 7 συνεχόμενες ημέρες του προγράμματος εργασίας (από την 1η έως την 7η, από την 2η έως την 8η, κοκ.) να έχει τουλάχιστον δύο ημέρες ανάπαυσης.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί), διατυπώνοντας την απάντησή σας με ένα ξεκάθαρο μαθηματικό φορμαλισμό. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα

βαθμολογηθεί. Επειδή, προφανώς, μπορεί το πρόβλημα να έχει πολλές λύσεις, προτείνετε μία εύλογη αντικειμενική συνάρτηση που να ποσοτικοποιεί την ποιότητα (ή το κόστος) μίας λύσης, διατυπώνοντας και αυτή με μαθηματική ακρίβεια.

3. (**20/60**) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(a)))).
p(X) :- p(f(X)).
p(f(b)) :- p(a).
p(f(f(f(b)))) :- p(c).
```

   (α′) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό;

   (β′) Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό.

   (γ′) Είναι η βάση Herbrand μοντέλο του προγράμματος; Αιτιολογήστε την απάντησή σας.

   (δ′) Μπορεί το `p(c)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(c)`.

   (ε′) Πιστεύετε ότι υπάρχει μία SLD-απόρριψη για τον εξής στόχο;
   `?- p(a), p(b), p(f(a)), p(f(b)), p(f(f(a))), p(f(f(b))).`
   Αιτιολογήστε την απάντησή σας χωρίς να προσπαθήσετε να κατασκευάσετε μία τέτοια SLD-απόρριψη.

4. (**15/60**) Έστω ότι έχουμε μια βάση γνώσης στην οποία η γνώση αναπαρίσταται από *if_then* κανόνες των οποίων η προϋπόθεση είναι είτε απλή πρόταση είτε σύζευξη προτάσεων, π.χ.

```
if a and b then e.
if a then b.
if b and c and d then f.
...
```

   Υποθέστε ότι ο τελεστής "`and`" είναι δεξιά προσεταιριστικός (`xfy`) και, για λόγους απλότητας, απαγορεύεται η χρήση παρενθέσεων. Δηλαδή, για παράδειγμα, απαγορεύεται να έχουμε κανόνες της μορφής `if (a and b) and c then e`.

   Να υλοποιήσετε μια μηχανή συμπερασμού για την αναπαράσταση αυτή, κατά την οποία η συλλογιστική θα γίνεται με τον εμπρόσθιο (forward) τρόπο, παράγοντας ένα συμπέρασμα την φορά ως εξής:

   (α′) Είτε η προϋπόθεση ενός κανόνα είναι αληθής,

   (β′) είτε, σε περίπτωση σύζευξης στην προϋπόθεση, η πρώτη πρόταση είναι αληθινή και για κάθε μία από τις υπόλοιπες προτάσεις της σύζευξης, να ισχύει ότι είτε είναι αληθινή είτε αποτελεί συμπέρασμα κανόνα του οποίου η προϋπόθεση αποτελείται από μία μόνο πρόταση και η οποία είναι αληθινή. Τα "ενδιάμεσα" συμπεράσματα που παράγονται με αυτόν τον ενός-βήματος οπίσθιο τρόπο, είτε τα καταγράφετε ως αληθή, είτε όχι, όπως προτιμάτε.

   Για να γίνει κατανοητό το δεύτερο εναλλακτικό, θεωρήστε ότι (είτε εξ αρχής, είτε λόγω συμπερασμού) ισχύει το `a`. Τότε, επιλέγοντας τον πρώτο κανόνα του παραδείγματος, μπορούμε να συμπεράνουμε ότι ισχύει και το `e`, διότι το `a` που είναι πρώτο ισχύει και το `b` προκύπτει από τον δεύτερο κανόνα άμεσα, εφόσον η προϋπόθεσή του αποτελείται από μόνο μία πρόταση, το `a`, και το οποίο αυτή τη στιγμή είναι αληθές. Για την υλοποίηση που θα προτείνετε, μπορείτε να βασιστείτε και να τροποποιήσετε/επεκτείνετε τον κώδικα των σημειώσεων.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2022 (5/9/2022)

1. (**15/60**) Να συμπληρωθεί κατάλληλα ο παρακάτω ημιτελής ορισμός σε Prolog του κατηγορήματος `maxcntunif/2`, έτσι ώστε όταν αυτό καλείται σαν `maxcntunif(List, MaxSub)`, να βρίσκει τη μέγιστη υπολίστα συνεχόμενων ενοποιήσιμων στοιχείων στη λίστα `List`, **χωρίς όμως να γίνεται η ενοποίηση**, και να επιστρέφει το αποτέλεσμα στη λίστα `MaxSub`. Αν υπάρχουν περισσότερες της μίας υπολίστες μεγίστου μήκους με συνεχόμενα ενοποιήσιμα στοιχεία, να επιστρέφονται όλες οι δυνατές λύσεις μέσω οπισθοδρόμησης.

```
maxcntunif(List, MaxSub) :-
   cntunif(N, MaxSub, List),
   \+ (...................).

cntunif(N, Sublist, List) :-
   append(...................),
   append(...................),
   unifall(Sublist),
   ...................

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

unifone(...................).
unifone(...................) :-
   ...................
```

Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- maxcntunif([a,a,b,b,b,b,c,d,d,d,e,e,e,e,f], MaxSub).
MaxSub = [b, b, b, b]    --> ;
MaxSub = [e, e, e, e]
?- maxcntunif([f(1),f(X),g(X),g(2),g(Y),g(3),g(Z),h(X),h(Y),h(Z)], MaxSub).
MaxSub = [g(X), g(2), g(Y)]   --> ;
MaxSub = [g(Y), g(3), g(Z)]   --> ;
MaxSub = [h(X), h(Y), h(Z)]
?- maxcntunif([1,2,3], MaxSub).
MaxSub = [1]   --> ;
MaxSub = [2]   --> ;
MaxSub = [3]
?- maxcntunif([X,Y,Z,W,Y,W], MaxSub).
MaxSub = [X, Y, Z, W, Y, W]
```

2. (**10/60**) Στην προτασιακή λογική, ένας τύπος είναι σε *συζευκτική κανονική μορφή – ΣΚΜ* (conjunctive normal form – CNF), όταν είναι μία σύζευξη διαζεύξεων, όπου κάθε διάζευξη, που ονομάζεται και *πρόταση* (clause), περιέχει προτασιακά σύμβολα ή αρνήσεις προτασιακών συμβόλων. Για παράδειγμα, ο ακόλουθος τύπος είναι σε ΣΚΜ:

   $$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee x_4)$$

   Το πρόβλημα της *μέγιστης ικανοποιησιμότητας – ΜΕΓΙΚ* (maximum satisfiability – MAX-SAT) έγκειται στην εύρεση κατάλληλων αναθέσεων τιμών στα προτασιακά σύμβολα (`true` ή `false`) έτσι ώστε να μεγιστοποιείται το πλήθος των προτάσεων του τύπου που αληθεύουν. Για παράδειγμα, στον παραπάνω τύπο, που αποτελείται από 7 προτάσεις, δεν υπάρχει ανάθεση τιμών στα προτασιακά σύμβολα που να κάνει αληθείς όλες τις προτάσεις, αλλά μπορεί να βρεθεί ανάθεση ώστε να αληθεύουν 6 από αυτές. Κατ΄ αρχήν, μπορείτε να βρείτε εμπειρικά μία τέτοια ανάθεση, που κάνει 6 από τις προτάσεις του τύπου αληθείς;

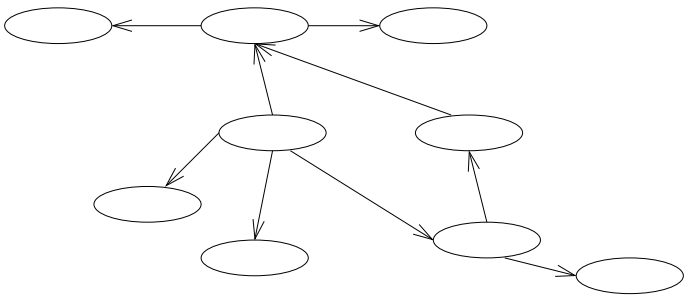
   Στη συνέχεια, μοντελοποιήστε το πρόβλημα ΜΕΓΙΚ σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους, τους ενδεχόμενους περιορισμούς μεταξύ τους και τη συνάρτηση κόστους. Η διατύπωση των περιορισμών και της συνάρτησης κόστους θα πρέπει να γίνει με μαθηματική ακρίβεια. Για να διευκολυνθείτε, δώστε την απάντησή σας με βάση τον παραπάνω τύπο. Αν σας ζητείτο να κάνετε και την υλοποίηση στην ECL^{i}PS^e, ποια δυνατότητά της, εκτός από τα προφανή κατηγορήματα αναζήτησης της `ic` και βελτιστοποίησης της `branch_and_bound`, πιστεύετε ότι θα σας φαινόταν ιδιαίτερα χρήσιμη;

3. (**20/60**) Γράψτε το **ελάχιστο** δυνατό οριστικό πρόγραμμα P (ως προς το πλήθος των χαρακτήρων που θα χρησιμοποιήσετε), το οποίο να έχει ως ελάχιστο μοντέλο το σύνολο:

   $$\begin{aligned} M_P \;=\; \{&\texttt{p(a)}, \texttt{p(f(f(a)))}, \texttt{p(f(f(f(f(a)))))}, \texttt{p(f(f(f(f(f(a))))))}, \\ &\texttt{q(f(a))}, \texttt{q(f(f(f(a))))}, \texttt{q(f(f(f(f(f(a))))))}\} \end{aligned}$$

   Ποια θα ήταν η προφανής απάντηση στο ερώτημα, αν δεν υπήρχε η απαίτηση του "ελάχιστου προγράμματος"; Δώστε και ένα άλλο μοντέλο του προγράμματος, που να είναι απειροσύνολο, αλλά να μην είναι η βάση Herbrand.

4. (**15/60**) Δώστε ονόματα, όπου νομίζετε ότι χρειάζεται, ώστε η παρακάτω εικόνα να αναπαριστά μέρος μιας βάσης γνώσης. Ποια μεθοδολογία ακολουθεί η αναπαράσταση; Γράψτε ένα πρόγραμμα Prolog που να υλοποιεί τη γνώση που αναπαριστάνεται στο (συμπληρωμένο) σχήμα σας.



ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2021 (17/6/2021)

1. (**15/60**) Στο πρόβλημα του σακιδίου (*knapsack problem*) δίνονται ένα σύνολο από αντικείμενα, που καθένα έχει συγκεκριμένο βάρος και συγκεκριμένη αξία, και ένα σακίδιο συγκεκριμένης χωρητικότητας (σε βάρος). Το ζητούμενο είναι να βρεθεί ποια αντικείμενα πρέπει να τοποθετήσουμε στο σακίδιο ώστε να μην υπερβούμε τη χωρητικότητά του και η συνολική αξία των αντικειμένων που θα τοποθετήσουμε σε αυτό να είναι η μέγιστη δυνατή. Τα δεδομένα για συγκεκριμένα αντικείμενα μπορούν να αναπαρασταθούν σε Prolog με ένα γεγονός `items/1` όπως φαίνεται στη συνέχεια, όπου το όρισμά του είναι μία λίστα με στοιχεία της μορφής `W/V`, ένα στοιχείο για κάθε αντικείμενο, όπου `W` είναι το βάρος του αντικειμένου και `V` είναι η αξία του.

   ```
   items([6/12, 15/10, 22/32, 14/19, 15/17, 18/11, 6/22, 14/13, 34/42, 40/7]).
   ```

   Το ζητούμενο είναι να ορίσουμε ένα κατηγόρημα `knapsack/4` σε Prolog το οποίο, με δεδομένο τον ορισμό ενός γεγονότος `items/1`, να δέχεται σαν πρώτο όρισμα τη χωρητικότητα `Capacity` ενός σακιδίου και να επιστρέφει σαν δεύτερο όρισμα ένα υποσύνολο των διαθέσιμων αντικειμένων, στη μορφή λίστας από στοιχεία της μορφής `W/V`, τα οποία να μπορούν να τοποθετηθούν στο σακίδιο, χωρίς να γίνει υπέρβαση της χωρητικότητάς του, και η συνολική αξία των αντικειμένων να είναι η μέγιστη δυνατή. Στο τρίτο και τέταρτο όρισμα, το κατηγόρημα `knapsack/4` να επιστρέφει το συνολικό βάρος και τη συνολική αξία των αντικειμένων που τοποθετήθηκαν στο σακίδιο. Επίσης, πρέπει να επιστρέφονται μέσω οπισθοδρόμησης όλες οι εξ ίσου βέλτιστες λύσεις, αν υπάρχουν περισσότερες από μία. Συμπληρώστε κατάλληλα το παρακάτω ημιτελές πρόγραμμα ώστε να υλοποιεί το ζητούμενο. Τα `.....` μπορούν να συμπληρωθούν με κώδικα κατά βούληση, αλλά τα δεδομένα τμήματα του προγράμματος δεν επιτρέπεται να μεταβληθούν.

   ```
   knapsack(Capacity, Solution, TotW, TotV) :-
      items(Items),
      solution(Capacity, Items, Solution, TotW, TotV),
      \+ (.................................).

   solution(_, [], [], ................).
   solution(Capacity, [W/V|Items], [W/V|Solution], TotW, TotV) :-
      solution(Capacity, Items, ................),
      ....................................
   solution(Capacity, [_|Items], Solution, TotW, TotV) :-
      ....................................
   ```

   Κάποιο παράδειγμα εκτέλεσης:

   ```
   ?- knapsack(80, Solution, Weight, Value).
   Solution = [6/12, 22/32, 14/19, 15/17, 6/22, 14/13]
   Weight = 77
   Value = 115                 --> ;
   Solution = [22/32, 14/19, 6/22, 34/42]
   Weight = 76
   Value = 115                 --> ;
   No
   ```

2. (**10/60**) Θεωρήστε την παραλλαγή του προβλήματος που διατυπώθηκε στο προηγούμενο θέμα κατά την οποία δίνεται πάλι ένα σύνολο N αντικειμένων με συγκεκριμένο βάρος W_i και συγκεκριμένη αξία V_i το καθένα ($1 \leq i \leq N$), αλλά έχουμε περισσότερα από ένα σακίδιο, έστω πλήθους K, με συγκεκριμένη χωρητικότητα (σε βάρος) C_j το καθένα ($1 \leq j \leq K$). Το ζητούμενο είναι να βρεθεί ποια αντικείμενα θα τοποθετηθούν σε ποια σακίδια, ώστε να μην γίνει υπέρβαση της

χωρητικότητας κανενός σακιδίου και να μεγιστοποιηθεί η συνολική αξία των αντικειμένων που θα τοποθετηθούν σε σακίδια. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί. Εξειδικεύστε την απάντησή σας για τα αντικείμενα του προηγούμενου θέματος, θεωρώντας ότι έχουμε τρία σακίδια με χωρητικότητες 30, 35, 40, παραθέτοντας τις μεταβλητές, τους περιορισμούς και τη συνάρτηση κόστους για το στιγμιότυπο αυτό. **Υπόδειξη:** Μοντελοποιήσεις των ασκήσεων 4 και 6.

3. (**20/60**) Δίνεται το εξής λογικό πρόγραμμα:

```
p(0,X,X).
p(s(X),Y,s(Z)) :- p(X,Y,Z).
```

   (α′) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό;

   (β′) Βρείτε το ελάχιστο μοντέλο του προγράμματος, είτε εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό, είτε έχοντας διαισθανθεί τη σωστή απάντηση, αφού διαγνώσετε ποια είναι πραγματικά η λειτουργικότητα αυτού του λογικού προγράμματος. Στη δεύτερη περίπτωση, να αναφέρετε και τη λειτουργικότητα αυτή.

   (γ′) Είναι η βάση Herbrand μοντέλο του προγράμματος; Αιτιολογήστε την απάντησή σας.

   (δ′) Μπορεί το `p(0,s(0),s(s(0)))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(0,s(0),s(s(0)))`.

   (ε′) Πιστεύετε ότι υπάρχει μία SLD-απόρριψη για τον εξής στόχο;
   `?- p(0,0,0), p(s(s(0)),s(0),s(s(s(0)))), p(s(0),s(0),s(0)).`
   Αιτιολογήστε την απάντησή σας χωρίς να προσπαθήσετε να κατασκευάσετε μία τέτοια SLD-απόρριψη.

4. (**15/60**) Ας θεωρήσουμε μια υλοποίηση των σημασιολογικών δικτύων σε Prolog στην οποία αναπαριστούμε την κάθε συσχέτιση (ακμή) του δικτύου με ένα γεγονός το οποίο έχει κατηγόρημα το όνομα της συσχέτισης, δύο ορίσματα που αντιστοιχούν στους κόμβους του δικτύου που συνδέει η ακμή και ένα ακόμα όρισμα που αντιστοιχεί στον βαθμό βεβαιότητας C για την αλήθεια της πρότασης ($0 \leq C \leq 1$). Θεωρήστε ότι οι εξειδικεύσεις οντοτήτων και τα στιγμιότυπα δηλώνονται μέσω μιας συσχέτισης με κατηγόρημα `isa`. Για παράδειγμα:

```
isa(albatross, bird, 1).      isa(ross, albatross, 0.9).    .............
active_at(bird, daylight, 0.75).                            .............
moving_method(bird, fly, 0.8).                              .............
colour(albatross, black_and_white, 0.95).                   .............
```

   Στην περίπτωση κληρονομικότητας, η βεβαιότητα μιας πρότασης για μια οντότητα που είναι εξειδίκευση μιας άλλης οντότητας προκύπτει από το γινόμενο της βεβαιότητας της πρότασης για τη γενικότερη οντότητα και της βεβαιότητας της πρότασης που δηλώνει την εξειδίκευση. Αν η συσχέτιση εντοπίζεται στην ιεραρχία των οντοτήτων σε βάθος μεγαλύτερο του 1, οι βεβαιότητες κάθε βήματος πολλαπλασιάζονται μεταξύ τους. Δηλαδή, στο παραπάνω παράδειγμα, ισχύει ότι η `moving_method` για το `ross` είναι `fly` με βεβαιότητα 0.72 ($0.72 = 0.9 \times 1 \times 0.8$). Υλοποιήστε σε Prolog ένα κατηγόρημα `certainty/4`, με γενική μορφή `certainty(RelName, EntName, Val, Cert)`, το οποίο να επιτυγχάνει όταν η οντότητα `EntName` έχει τιμή `Val` για τη συσχέτιση `RelName` με βεβαιότητα `Cert`. Για παράδειγμα:

```
?- certainty(moving_method, ross, Val, Cert).
Val = fly
Cert = 0.72
```

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β′ Περιόδου 2021 (30/8/2021)

1. (**15/60**) Θεωρήστε ότι αναπαριστούμε ένα μη-κατευθυνόμενο γράφο σε Prolog μέσω ενός γεγονότος με κατηγόρημα `a_graph/2`, όπου το πρώτο του όρισμα είναι το πλήθος των κόμβων του γράφου, έστω `N`, και το δεύτερο όρισμα είναι οι ακμές του γράφου, ως λίστα στοιχείων της μορφής `I-J`, όπου `I` και `J` είναι οι αύξοντες αριθμοί των κόμβων που συνδέει η αντίστοιχη ακμή ($1 \leq$ `I` $<$ `J` $\leq$ `N`). Ένα παράδειγμα είναι το εξής:

```
a_graph(8, [1-4, 1-5, 1-8, 2-3, 2-5, 2-6, 2-7, 3-7, 4-5, 6-7, 7-8]).
```

Μία *κλίκα* (clique) σε ένα γράφο είναι ένα υποσύνολο των κόμβων του γράφου, τέτοιο ώστε να υπάρχει ακμή μεταξύ δύο οποιωνδήποτε κόμβων της κλίκας. Το πρόβλημα της *μέγιστης κλίκας* (maximum clique) συνίσταται στην εύρεση κλίκας σε ένα γράφο που περιλαμβάνει μέγιστο αριθμό κόμβων. Το ζητούμενο είναι να ορίσουμε ένα κατηγόρημα `max_clique/4` σε Prolog το οποίο να δέχεται σαν πρώτο και δεύτερο όρισμα το πλήθος των κόμβων και τις ακμές, αντίστοιχα, ενός γράφου, στη μορφή που περιγράφηκε προηγουμένως, και να επιστρέφει στο τρίτο και τέταρτο όρισμα ένα υποσύνολο των κόμβων που αποτελούν μέγιστη κλίκα και το πλήθος τους, αντίστοιχα. Επίσης, πρέπει να επιστρέφονται μέσω οπισθοδρόμησης όλες οι μέγιστες κλίκες, αν υπάρχουν περισσότερες από μία. Συμπληρώστε κατάλληλα το παρακάτω ημιτελές πρόγραμμα ώστε να υλοποιεί το ζητούμενο. Τα `.....` μπορούν να συμπληρωθούν με κώδικα κατά βούληση, αλλά τα δεδομένα τμήματα του προγράμματος δεν επιτρέπεται να μεταβληθούν.

```
max_clique(N, Edges, MaxCl, MaxSize) :-
   clique(N, Edges, MaxCl, MaxSize),
   \+ (.....).

clique(N, Edges, Clique, Size) :-
   findall(K, between(1, N, K), Nodes),
   is_subset(Nodes, Clique),
   \+ (.....),
   length(Clique, Size).

.....
```

Κάποια παραδείγματα εκτέλεσης:

```
?- a_graph(N, Edges), max_clique(N, Edges, Clique, Size).
N = 8
Edges = [1 - 4, 1 - 5, 1 - 8, 2 - 3, 2 - 5, 2 - 6, 2 - 7, 3 - 7, 4 - 5, 6 - 7, 7 - 8]
Clique = [1, 4, 5]
Size = 3                --> ;
N = 8
Edges = [1 - 4, 1 - 5, 1 - 8, 2 - 3, 2 - 5, 2 - 6, 2 - 7, 3 - 7, 4 - 5, 6 - 7, 7 - 8]
Clique = [2, 3, 7]
Size = 3                --> ;
N = 8
Edges = [1 - 4, 1 - 5, 1 - 8, 2 - 3, 2 - 5, 2 - 6, 2 - 7, 3 - 7, 4 - 5, 6 - 7, 7 - 8]
Clique = [2, 6, 7]
Size = 3                --> ;
No

?- a_graph(N, Edges), clique(N, Edges, Clique, 4).
No
```

2. (**10/60**) Μοντελοποιήστε το πρόβλημα εύρεσης μέγιστης κλίκας σε γράφο που διατυπώθηκε στο προηγούμενο θέμα σαν πρόβλημα ικανοποίησης περιορισμών με βελτιστοποίηση (μεταβλητές, πεδία, περιορισμοί, συνάρτηση κόστους). Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί. Μπορείτε να δώσετε την απάντησή σας είτε γενικά, για ένα γράφο με N κόμβους και ένα σύνολο ακμών E, είτε εξειδικεύοντάς την για την περίπτωση του παραδείγματος γράφου του προηγούμενου θέματος. Σε κάθε περίπτωση, η απάντησή σας θα πρέπει να χαρακτηρίζεται από μαθηματική ακρίβεια.

3. (**20/60**) Δίνεται το εξής λογικό πρόγραμμα:

```
p(0,0).
p(k(0,X),s(Y)) :- p(X,Y).
```

   (α′) Βρείτε το ελάχιστο μοντέλο του προγράμματος εφαρμόζοντας κατάλληλη μέθοδο για το σκοπό αυτό.

   (β′) Ποιο ενσωματωμένο κατηγόρημα της Prolog σας θυμίζει η λειτουργικότητα αυτού του λογικού προγράμματος; Ποιες είναι οι αντιστοιχίες μεταξύ του ενσωματωμένου κατηγορήματος και του λογικού προγράμματος;

   (γ′) Μπορεί το `p(k(0,0),0)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(k(0,0),0)`.

   (δ′) Πιστεύετε ότι υπάρχει μία SLD-απόρριψη για τον εξής στόχο;
   `?- p(k(0,0),s(0)), p(k(0,k(0,0)),s(s(0))), p(k(0,0),s(s(0))).`
   Αιτιολογήστε την απάντησή σας χωρίς να προσπαθήσετε να κατασκευάσετε μία τέτοια SLD-απόρριψη.

4. (**15/60**) Πού δίνεται η έμφαση στην αναπαράσταση γνώσης μέσω σημασιολογικών δικτύων και ποιος μηχανισμός παράγει νέα γνώση από υπάρχουσα στην περίπτωσή τους;

   Ας θεωρήσουμε μια υλοποίηση των σημασιολογικών δικτύων σε Prolog στην οποία αναπαριστούμε την κάθε συσχέτιση (ακμή) του δικτύου με ένα γεγονός το οποίο έχει κατηγόρημα το όνομα της συσχέτισης και δύο ορίσματα που αντιστοιχούν στους κόμβους του δικτύου που συνδέει η ακμή. Θεωρήστε ότι οι εξειδικεύσεις οντοτήτων και τα στιγμιότυπα δηλώνονται μέσω μιας συσχέτισης με κατηγόρημα `isa`. Για παράδειγμα:

```
isa(albatross, bird).       isa(ross, albatross).       ..................
active_at(bird, daylight).                              ..................
moving_method(bird, fly).                               ..................
colour(albatross, black_and_white).                     ..................
```

   Είναι γνωστό ότι στα ευφυή συστήματα είναι συχνά επιθυμητό όχι μόνο να καταλήγουν σε ένα συμπέρασμα σχετικά με ένα ερώτημα του χρήστη, αλλά και να παρέχουν στο χρήστη το *μονοπάτι* συμπερασμού που οδηγεί στο συμπέρασμα, δηλαδή να αιτιολογούν την απάντησή τους ή, αλλιώς, να απαντάνε σε ερωτήσεις "*πώς*".

   Αφού αποφασίσετε πώς μπορεί να αναπαρασταθεί σε Prolog ένα τέτοιο μονοπάτι συμπερασμού για τα σημασιολογικά δίκτυα, υλοποιήστε ένα κατηγόρημα `value/4`, με γενική μορφή `value(RelName, EntName, Val, Path)`, όπου `RelName` και `EntName` είναι ονόματα συσχέτισης και οντότητας, αντίστοιχα, και `Val`, `Path` είναι μεταβλητές που αντιστοιχούν στην τιμή, `Val`, της συσχέτισης `RelName` για την οντότητα `EntName` και στο μονοπάτι συμπερασμού, `Path`, που οδήγησε στην απάντηση.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2020

1. (**25/75**) Έστω ότι σε μία βιομηχανία πρέπει να προγραμματισθεί η εκτέλεση ενός συνόλου εργασιών. Κάθε εργασία **διαρκεί μία μονάδα χρόνου**, απαιτεί για την εκτέλεσή της ένα σύνολο από πόρους και η εκτέλεση των εργασιών πρέπει να έχει ολοκληρωθεί μέχρι κάποια δεδομένη χρονική στιγμή. Τα δεδομένα αυτού του προβλήματος μπορούν να αναπαρασταθούν σε Prolog με γεγονότα `task/2` όπως φαίνεται στη συνέχεια, όπου το πρώτο όρισμα του `task/2` είναι η ταυτότητα της εργασίας και το δεύτερο όρισμα είναι η λίστα των πόρων που απαιτούνται για την εκτέλεσή της.

```
task(t1,[a,b,c]).  task(t2,[a,d]).  task(t3,[b,d,e,f]).  task(t4,[c,e]).  task(t5,[f]).
```

Θεωρώντας ότι δεν επιτρέπεται να εκτελούνται παράλληλα δύο εργασίες που απαιτούν κοινό πόρο, το ζητούμενο είναι να ορίσουμε ένα κατηγόρημα `tsched/2` σε Prolog το οποίο να δέχεται σαν πρώτο όρισμα την προθεσμία `Deadline` εκτέλεσης του συνόλου των εργασιών και να επιστρέφει σαν δεύτερο όρισμα το πρόγραμμα εκτέλεσης, σαν μία λίστα από όρους της μορφής `Tid/Slot`, όπου `Tid` η ταυτότητα μίας εργασίας και `Slot` η σχισμή (μοναδιαίο χρονικό διάστημα) που θα εκτελεσθεί η εργασία (από 1 έως `Deadline`). Συμπληρώστε κατάλληλα το παρακάτω ημιτελές πρόγραμμα ώστε να υλοποιεί το ζητούμενο. Τα ..... μπορούν να συμπληρωθούν με κώδικα κατά βούληση, αλλά τα δεδομένα τμήματα του προγράμματος δεν επιτρέπεται να μεταβληθούν.

```
tsched(Deadline, Schedule) :-
   .....................................
   schall(Tasks, Deadline, [], Schedule).
schall([], _, Schedule, Schedule).
schall([T/Rs|Tasks], Deadline, SoFarSchedule, Schedule) :-
   .....................................
   append(SoFarSchedule, [T/N], NewSoFarSchedule),
   schall(Tasks, Deadline, NewSoFarSchedule, Schedule).
.....................................
```

Κάποια παραδείγματα εκτέλεσης:

```
?- tsched(4,S).
S = [t1/1, t2/2, t3/3, t4/2, t5/1]  --> ;
S = [t1/1, t2/2, t3/3, t4/2, t5/2]  --> ;
S = [t1/1, t2/2, t3/3, t4/2, t5/4]  --> ;
.....................................
?- tsched(2,S).
No
```

2. (**15/75**) Θεωρήστε την παραλλαγή του προβλήματος που διατυπώθηκε στο προηγούμενο θέμα κατά την οποία δεν δίνεται `Deadline`, αλλά ζητούμε να βρούμε ένα βέλτιστο πρόγραμμα εκτέλεσης των εργασιών, υπό την έννοια ότι πρέπει να ολοκληρώνεται το συντομότερο δυνατό. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί. Εξειδικεύστε την απάντησή σας για τα δεδομένα που δίνονται στο προηγούμενο θέμα, δηλαδή αναφέρατε ρητά ποιες μεταβλητές θα πρέπει να ορισθούν για το συγκεκριμένο στιγμιότυπο προβλήματος, ποια θα είναι τα πεδία τους, ποιοι θα είναι οι περιορισμοί μεταξύ τους και ποια θα είναι η συνάρτηση κόστους.

3. (**20/75**) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(a)))).
p(X) :- p(f(X)).
p(f(b)) :- p(a).
p(f(f(f(b)))) :- p(c).
```

   (α′) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό;

   (β′) Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό.

   (γ′) Είναι η βάση Herbrand μοντέλο του προγράμματος; Αιτιολογήστε την απάντησή σας.

   (δ′) Μπορεί το `p(c)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(c)`.

   (ε′) Πιστεύετε ότι υπάρχει μία SLD-απόρριψη για τον εξής στόχο;

   ```
   ?- p(f(f(a))), p(f(b)), p(c).
   ```

   Αιτιολογήστε την απάντησή σας χωρίς να αποπειραθείτε να κατασκευάσετε μία τέτοια SLD-απόρριψη.

4. (**15/75**) Σε ποιες περιπτώσεις αναπαράστασης γνώσης εξυπηρετεί η χρήση πλαισίων; Δώστε μια υλοποίηση της αναπαράστασης σε Prolog, διαφορετική από αυτή που παρουσιάστηκε στο μάθημα, καθώς και μια υλοποίηση της μηχανής συμπερασμού με βάση αυτή. **Σημείωση:** Η υποστήριξη υπολογισμού τιμής σχισμών με εκτέλεση διαδικασιών δεν χρειάζεται να υποστηρίζεται.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β′ Περιόδου 2020

1. (**25/75**) Έστω ότι δίνονται τα παρακάτω γεγονότα Prolog:

```
activity(1, act(0,3)).     activity(2, act(0,4)).     activity(3, act(1,5)).
activity(4, act(4,6)).     activity(5, act(6,8)).     activity(6, act(6,9)).
activity(7, act(9,10)).    activity(8, act(9,13)).    activity(9, act(11,14)).
```

Τα γεγονότα `activity/2` κωδικοποιούν δραστηριότητες που θα πρέπει να στελεχωθούν από άτομα. Κάθε γεγονός `activity(<AId>, act(<Start>,<End>))` σημαίνει ότι η δραστηριότητα `<AId>` αρχίζει τη χρονική στιγμή `<Start>` και τελειώνει τη χρονική στιγμή `<End>`. Υποθέστε ότι κάθε δραστηριότητα πρέπει να στελεχωθεί από ένα ακριβώς άτομο και ότι κάθε άτομο μπορεί να αναλάβει οσεσδήποτε δραστηριότητες, αρκεί να ισχύει ο περιορισμός ότι όχι μόνο δεν μπορεί ένα άτομο να αναλάβει δύο δραστηριότητες που επικαλύπτονται χρονικά, αλλά θα πρέπει μεταξύ δύο οποιωνδήποτε διαδοχικών δραστηριοτήτων κάθε ατόμου να μεσολαβεί τουλάχιστον μία μονάδα χρόνου. Το ζητούμενο είναι να ορίσουμε σε Prolog ένα κατηγόρημα `assignment/2`, το οποίο, όταν καλείται ως `assignment(<NPersons>, <Assignment>)`, όπου `<NPersons>` είναι το πλήθος των διαθέσιμων ατόμων, να αναθέτει τις δοθείσες δραστηριότητες στα δοθέντα άτομα με εφικτό τρόπο, επιστρέφοντας στη μεταβλητή `<Assignment>` τις αναθέσεις με κατάλληλη κωδικοποίηση, για παράδειγμα σαν μία λίστα από στοιχεία της μορφής `<AId>-<PId>`, που σημαίνουν ότι η δραστηριότητα `<AId>` ανατίθεται στο άτομο `<PId>`. Το κατηγόρημα `assignment/2` θα πρέπει να επιστρέφει όλες τις εφικτές λύσεις μέσω οπισθοδρόμησης. Συμπληρώστε κατάλληλα το παρακάτω ημιτελές πρόγραμμα ώστε να υλοποιεί το ζητούμενο.

```
assignment(NPersons, Assignment) :-
   ................................,              % Gather all activities in list AIds
   assign(AIds, NPersons, Assignment).
assign([], _, []).
assign([AId|AIds], NPersons, [AId-PId|Assignment]) :-
   assign(AIds, NPersons, Assignment),
   ................................,            % Select a person PId for activity AId
   activity(AId, act(Ab, Ae)),
   ................................,   % Gather in list APIds so far activities of PId
   valid(Ab, Ae, APIds).          % Is current assignment consistent with previous ones?
valid(_, _, []).
valid(Ab1, Ae1, [APId|APIds]) :-
   activity(APId, act(Ab2, Ae2)),
   ................................,
   valid(Ab1, Ae1, APIds).
................................     % Definitions of possible auxiliary predicates
```

Κάποια παραδείγματα εκτέλεσης:

```
?- assignment(3, Assignment).
Assignment = [1 - 2, 2 - 3, 3 - 1, 4 - 2, 5 - 1, 6 - 3, 7 - 1, 8 - 2, 9 - 1]  --> ;
Assignment = [1 - 2, 2 - 1, 3 - 3, 4 - 2, 5 - 1, 6 - 3, 7 - 1, 8 - 2, 9 - 1]  --> ;
................................
?- assignment(2, Assignment).
No
?- findall(Assignment, assignment(3, Assignment), Assignments), length(Assignments, N).
................................
N = 48
```

2. (**15/75**) Θεωρήστε την επέκταση του προβλήματος που διατυπώθηκε στο προηγούμενο θέμα κατά την οποία πρέπει να ισχύει και ο επιπλέον περιορισμός ότι η συνολική διάρκεια των δραστηριοτήτων που θα ανατεθούν σε ένα άτομο δεν πρέπει να υπερβαίνει κάποιο εκ των προτέρων καθορισμένο όριο, π.χ. `<MaxTime>`. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί). Η διατύπωση των περιορισμών θα πρέπει να γίνει με μαθηματική αυστηρότητα. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί. Εξειδικεύστε την απάντησή σας για τα δεδομένα που δίνονται στο προηγούμενο θέμα, θεωρώντας ότι `<MaxTime> = 10`. Δηλαδή, αναφέρατε ρητά ποιες μεταβλητές θα πρέπει να ορισθούν για το συγκεκριμένο στιγμιότυπο προβλήματος, ποια θα είναι τα πεδία τους και ποιοι θα είναι οι περιορισμοί μεταξύ τους. Τέλος, μπορείτε να επεκτείνετε το πρόβλημα ώστε να μετατραπεί σε πρόβλημα βελτιστοποίησης με κάποια εύλογη συνάρτηση κόστους; Διατυπώστε μαθηματικά τη συνάρτηση κόστους που προτείνετε.

3. (**20/75**) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(f(a)))),f(f(f(a)))).
p(X,Y) :- p(X,f(f(Y))).
p(X,Y) :- p(f(f(X)),Y).
q(X) :- p(X,X).
```

   (α′) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό;

   (β′) Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό.

   (γ′) Είναι η βάση Herbrand μοντέλο του προγράμματος; Αιτιολογήστε την απάντησή σας.

   (δ′) Μπορεί το `p(f(f(a)),a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(f(a)),a)`.

   (ε′) Πιστεύετε ότι υπάρχει μία SLD-απόρριψη για τον εξής στόχο;
   `?- p(f(f(a)),f(a)), p(a,f(a)), p(f(f(a)),f(f(f(f(a)))))`.
   Αιτιολογήστε την απάντησή σας χωρίς να αποπειραθείτε να κατασκευάσετε μία τέτοια SLD-απόρριψη.

4. (**15/75**) Τι εννοούμε όταν μιλάμε για "αβεβαιότητα (uncertainty) στη γνώση"; Στα πλαίσια του μαθήματος, θεωρήσαμε μια θεωρία χειρισμού αβεβαιότητας όπου η βεβαιότητα για μια πρόταση αναπαρίσταται από έναν αριθμό $0 \leq Cert \leq 1$ και οι νόμοι της για τη διάζευξη, σύζευξη και τους if-then κανόνες υλοποιούνται από το παρακάτω Prolog πρόγραμμα:

```
certainty(P, Cert) :- given(P, Cert).
certainty(Cond1 and Cond2, Cert) :-
   certainty(Cond1, Cert1), certainty(Cond2, Cert2), min(Cert1, Cert2, Cert).
certainty(Cond1 or Cond2, Cert) :-
   certainty(Cond1, Cert1), certainty(Cond2, Cert2), max(Cert1, Cert2, Cert).
certainty(P, Cert) :-
   if Cond then P : C1, certainty(Cond, C2), Cert is C1 * C2.
```

   Αν ο στόχος "`?- certainty(c19_drug_to_be_discovered,Cert).`" αποτύχει, τι συμπεραίνετε για την πρόταση `c19_drug_to_be_discovered`; Χωρίς να προσθέσετε πληροφορία στη βάση γνώσης για τη συγκεκριμένη πρόταση, π.χ. μέσω προσθήκης ενός γεγονότος με κατηγόρημα `given/2`, αλλάξτε το παραπάνω πρόγραμμα ώστε να παραμείνει σωστό, αλλά να μην αποτυγχάνει ποτέ.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2019

1. (α΄) Συμπληρώστε κατάλληλα το παρακάτω πρόγραμμα Prolog (**τρεις** προτάσεις μόνο), έτσι ώστε τα κατηγορήματα `evenlength/1` και `oddlength/1` να επιτυγχάνουν όταν καλούνται με όρισμα μία λίστα που έχει άρτιο ή περιττό πλήθος στοιχείων, αντίστοιχα. Δεν επιτρέπεται να χρησιμοποιήσετε ενσωματωμένα κατηγορήματα αριθμητικής της Prolog. Επίσης, δεν επιτρέπεται να τροποποιήσετε, ούτε κατ΄ ελάχιστο, το δεδομένο τμήμα προγράμματος.

```
.................
evenlength(L) :- .................
oddlength(L) :- .................
```

   (β΄) Υποθέστε ότι έχουμε ένα σύστημα Prolog, το οποίο δεν διαθέτει ενσωματωμένα κατηγορήματα αριθμητικής, οπότε η κλασική αναπαράσταση των αριθμών δεν έχει κάποια σημασιολογική αξία. Θα μπορούσαμε όμως στο σύστημα αυτό να αναπαραστήσουμε τους μη αρνητικούς ακεραίους με λίστες, όπου το πλήθος των στοιχείων μίας λίστας αντιστοιχεί στον αριθμό που αναπαριστά η λίστα. Για παράδειγμα, οι αριθμοί 0, 1, 2, 3, κλπ. μπορούν να αναπαρασταθούν, αντίστοιχα, από τις λίστες `[]`, `[_]`, `[_,_]`, `[_,_,_]`, κλπ. Αυτά καθεαυτά τα στοιχεία των λιστών δεν μας ενδιαφέρουν. Μπορεί να είναι οτιδήποτε. Στα προηγούμενα παραδείγματα είναι απλώς ανώνυμες μεταβλητές. Ακολουθώντας αυτή την αναπαράσταση των μη αρνητικών ακεραίων, ορίστε σε Prolog τα κατηγορήματα `add/3`, `sbt/3`, `mlt/3`, `dvd/3` και `mdl/3`, που να υλοποιούν τις πράξεις της πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, ακέραιας διαίρεσης (πηλίκο) και υπολοίπου διαίρεσης, αντίστοιχα, μεταξύ των δύο πρώτων ορισμάτων τους, επιστρέφοντας το αποτέλεσμα στο τρίτο όρισμα. Κάποια παραδείγματα εκτέλεσης:

```
?- add([_,_,_],[_,_,_,_],N).
N = [_,_,_,_,_,_,_]
?- sbt([_,_,_,_,_],[_,_],N).
N = [_,_,_]
?- sbt([_,_,_],[_,_,_,_],N).
no
?- mlt([_,_],[_,_,_,_],N).
N = [_,_,_,_,_,_,_,_]
?- dvd([_,_,_,_,_,_,_],[_,_,_],N).
N = [_,_]
?- mdl([_,_,_,_,_,_,_],[_,_,_],N).
N = [_]
```

2. (α΄) Στους γενετικούς αλγορίθμους ένα (δυαδικό) άτομο είναι μία ακολουθία από 0 και 1. Ένα *σχήμα* είναι ένα πρότυπο που περιλαμβάνει 0, 1 και το αδιάφορο σύμβολο *. Ορίζουμε ότι ένα άτομο ταιριάζει με ένα σχήμα, όταν οι αντίστοιχες θέσεις τους ταιριάζουν. Το 0 ταιριάζει με το 0 και το * και το 1 ταιριάζει με το 1 και το *. Είναι αυτονόητο ότι αναγκαία συνθήκη για να ταιριάζουν ένα άτομο και ένα σχήμα είναι να έχουν ίδιο μήκος. Ορίστε σε Prolog ένα κατηγόρημα `matching/2`, το οποίο όταν καλείται με πρώτο όρισμα μία λίστα από σχήματα, όπου κάθε σχήμα αναπαρίσταται σαν λίστα των στοιχείων του, να επιστρέφει στο δεύτερο όρισμα άτομα (λίστες των στοιχείων τους) που ταιριάζουν με όλα τα σχήματα. Μέσω οπισθοδρόμησης να επιστρέφονται όλες οι πιθανές λύσεις. Παράδειγμα:

```
?- matching([[1,0,*,*,0,*,1,0],[1,*,*,0,0,*,*,*],[*,0,1,*,*,*,1,*]],A).
A = [1,0,1,0,0,0,1,0]  --> ;
A = [1,0,1,0,0,1,1,0]
```

(β′) Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο N πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς 1, 2, 3, …, N. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει M συνδυασμούς πτήσεων P_i ($1 \leq i \leq M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις 1, 2, 3, …, N και, επίσης, τα P_i δεν είναι κατ΄ ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί ξένοι μεταξύ τους, που καλύπτουν ακριβώς τις πτήσεις της εταιρείας, με σκοπό να ανατεθούν σε συγκεκριμένους κυβερνήτες. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.) στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος. Για να γίνει περισσότερο σαφές το ζητούμενο, δείτε το εξής παράδειγμα:

Έστω $N = 5$, $M = 7$ και

i	P_i	C_i
1	$\{1, 2\}$	6
2	$\{1, 2, 3\}$	7
3	$\{1, 5\}$	5
4	$\{2\}$	3
5	$\{3, 4\}$	4
6	$\{4, 5\}$	8
7	$\{5\}$	6

Τότε, η βέλτιστη λύση είναι η $\{\{1, 5\}, \{2\}, \{3, 4\}\}$, με κόστος 12 (=5+3+4), ενώ οι άλλες πιθανές λύσεις $\{\{1, 2\}, \{3, 4\}, \{5\}\}$ και $\{\{1, 2, 3\}, \{4, 5\}\}$ έχουν κόστη 16 (=6+4+6) και 15 (=7+8), αντίστοιχα.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(X,f(Y)) :- p(f(X),Y).
p(f(Y),X) :- p(X,f(Y)).
p(f(a),b).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(a,a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(a,a)`. Επίσης, μπορεί το `p(b,f(b))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(b,f(b))`.

(β′) Στη μεθοδολογία αναπαράστασης γνώσης των σημασιολογικών δικτύων, ο συμπερασμός επιτυγχάνεται μέσω κληρονομικότητας. Η πολλαπλή κληρονομικότητα είναι ένα είδος κληρονομικότητας όπου μια οντότητα είναι άμεση εξειδίκευση δύο ή παραπάνω οντοτήτων. Τότε, ένας κόμβος του δικτύου συνδέεται άμεσα με παραπάνω από έναν άλλο κόμβο μέσω ακμής isa. Σε αυτήν την περίπτωση, υποθέστε ότι η κληρονομικότητα τιμών ιδιοτήτων (και συσχετίσεων) προκύπτει από τον κοντινότερο πρόγονο, δηλαδή τη συντομότερη ακολουθία σχέσεων isa που οδηγούν στο ζητούμενο. Σε περίπτωση ακολουθιών ίσου μήκους, να επιστρέφονται μέσω οπισθοδρόμησης όλες οι πιθανές απαντήσεις. Υλοποιήστε σε Prolog μία μηχανή συμπερασμού που υποστηρίζει πολλαπλή κληρονομικότητα με τη λειτουργικότητα αυτή. (*Σημείωση:* Μπορείτε να βασιστείτε στην υλοποίηση των σημειώσεων).

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2019

1. (α΄) Τι απάντηση θα δώσει ένα σύστημα Prolog στην εξής ερώτηση;

```
?- p(X0, X1, X2, f(Y0, Y0), f(Y1, Y1), Y2) = p(a, f(X0, X0), f(X1, X1), Y1, Y2, X2).
```

(β΄) Υιοθετώντας την αναπαράσταση ενός συνόλου σε Prolog ως λίστας των στοιχείων του, συμπληρώστε τον παρακάτω ορισμό του κατηγορήματος `allsubsets/2`, έτσι ώστε όταν καλείται με πρώτο όρισμα ένα σύνολο, να επιστρέφει στο δεύτερο όρισμα μία λίστα όλων των υποσυνόλων του. Δεν επιτρέπεται να ορίσετε άλλες προτάσεις εκτός από τις ημιτελείς που δίνονται, τις οποίες πρέπει να συμπληρώσετε κατάλληλα.

```
allsubsets(Set, Subsets) :- findall(......, subs(.............), ......).
subs(.............).
subs(.............) :- ........................
subs(.............) :- ........................
```

Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- allsubsets([a,b,c], Subsets).
Subsets = [[a,b,c], [a,b], [a,c], [a], [b,c], [b], [c], []]
```

2. (α΄) Ένας πίθηκος, ένα κουτί και μία μπανάνα βρίσκονται σε ένα δωμάτιο. Ο πίθηκος ή/και το κουτί μπορούν να βρίσκονται σε κάποιες πιθανές θέσεις στο δωμάτιο (`atdoor`, `atwindow`, `eastcorner`, `middle`). Ο πίθηκος μπορεί να βρίσκεται είτε στο έδαφος (`onfloor`) είτε επάνω στο κουτί (`onbox`). Η μπανάνα μπορεί να βρίσκεται κρεμασμένη από το ταβάνι στο κέντρο του δωματίου (`ceiling`), ή στα χέρια του πιθήκου (`athand`) ή μπορεί ο πίθηκος να την έχει φάει (`instomach`). Ο πίθηκος μπορεί να είναι πεινασμένος (`hungry`) ή χορτάτος (`fullup`).

i) Προτείνετε μία αναπαράσταση σε Prolog του κόσμου που περιγράφηκε προηγουμένως. Με βάση την αναπαράσταση αυτή, πώς θα διατυπώνατε μία αρχική κατάσταση του προβλήματος στην οποία ο πίθηκος βρίσκεται στην πόρτα, το κουτί στην ανατολική γωνία του δωματίου, η μπανάνα κρεμασμένη στο ταβάνι στο κέντρο του δωματίου και ο πίθηκος είναι πεινασμένος; Επίσης, θεωρώντας ότι συνήθως οι πίθηκοι χορταίνουν με μία μπανάνα, πώς θα διατυπώνατε μία τελική κατάσταση του προβλήματος στην οποία ο πίθηκος βρίσκεται στην πόρτα, το κουτί στο παράθυρο και ο πίθηκος είναι χορτάτος;

ii) Θεωρήστε ότι ο πίθηκος μπορεί να περπατά μέσα στο δωμάτιο, μεταξύ των διαφόρων θέσεών του, να σπρώχνει το κουτί από θέση σε θέση, να σκαρφαλώνει στο κουτί ή να κατεβαίνει από το κουτί, εφόσον βρίσκεται στην ίδια θέση με το κουτί, να πιάνει την μπανάνα από το ταβάνι, εφόσον είναι επάνω στο κουτί στο κέντρο του δωματίου, διαφορετικά δεν την φτάνει, και να τρώει την μπανάνα αν είναι πεινασμένος, οπότε χορταίνει. Προτείνετε μία αναπαράσταση σε Prolog των πιθανών ενεργειών που μπορούν να γίνουν στον κόσμο του πιθήκου και ορίστε κατάλληλες προτάσεις Prolog που περιγράφουν με σαφήνεια πότε μπορεί να εφαρμοσθεί κάθε ενέργεια και τι αποτελέσματα έχει.

iii) Χωρίς να κάνετε κάποια υλοποίηση, πώς θα προτείνατε να αντιμετωπίσετε σε Prolog αυτό το πρόβλημα σχεδιασμού (planning), ώστε δεδομένης μίας αρχικής και μίας τελικής κατάστασης και του ορισμού των δυνατών ενεργειών, να βρίσκεται βέλτιστη λύση, δηλαδή με το ελάχιστο πλήθος ενεργειών;

(β′) Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο N πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς 1, 2, 3, …, N. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει M συνδυασμούς πτήσεων P_i ($1 \leq i \leq M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις 1, 2, 3, …, N και, επίσης, τα P_i δεν είναι κατ΄ ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας, αλλά αυτό δεν αφορά το συγκεκριμένο πρόβλημα. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί, όχι κατ΄ ανάγκη ξένοι μεταξύ τους, τέτοιοι ώστε κάθε πτήση να περιλαμβάνεται σε τουλάχιστον έναν από τους επιλεγμένους συνδυασμούς. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.) στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος. Για να γίνει περισσότερο σαφές το ζητούμενο, δείτε το εξής παράδειγμα:
Έστω $N = 5$, $M = 7$ και

i	P_i	C_i
1	$\{1, 2\}$	4
2	$\{1, 2, 3\}$	5
3	$\{2, 4, 5\}$	6
4	$\{2, 5\}$	2
5	$\{3, 4\}$	7
6	$\{3, 4, 5\}$	8
7	$\{5\}$	5

Τότε, η βέλτιστη λύση είναι η $\{\{1, 2, 3\}, \{2, 4, 5\}\}$, με κόστος 11 (=5+6), ενώ υπάρχουν πολλές άλλες πιθανές λύσεις, καμία με κόστος μικρότερο του 11, για παράδειγμα η $\{\{1, 2\}, \{3, 4, 5\}\}$ με κόστος 12 (=4+8).
Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση. Δεν ζητείται να κάνετε κάποια υλοποίηση. Για την ακρίβεια, κώδικας στην απάντηση δεν θα βαθμολογηθεί.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(X,f(Y)) :- p(f(X),Y).
p(f(Y),X) :- p(X,f(Y)).
p(f(a),b).
```

Ποια από τα παρακάτω σύνολα είναι και ποια δεν είναι μοντέλα για το πρόγραμμα και γιατί;

i. `{p(f(a),b)}`
ii. `{p(f(a),b), p(a,f(b))}`
iii. `{p(f(a),b), p(a,f(b)), p(f(b),a), p(b,f(a))}`
iv. `{p(f(a),b), p(a,f(b)), p(f(b),a), p(b,f(a)), p(a,a)}`
v. `{p(f(a),b), p(a,f(b)), p(f(b),a), p(b,f(a)), p(b,f(b))}`
vi. `{p(f(a),b), p(a,f(b)), p(f(b),a), p(b,f(a)), p(b,f(b)), p(f(b),b)}`

(β′) Η μεθοδολογία αναπαράστασης γνώσης μέσω σημασιολογικών δικτύων και η μεθοδολογία αναπαράστασης γνώσης μέσω πλαισίων έχουν ομοιότητες. Αναφέρατε κάποιες από αυτές. Ποια χαρακτηριστικά του προβλήματός σας θα σας έκαναν να επιλέξετε τη μία μεθοδολογία και ποια την άλλη;
Με δεδομένες τις υλοποιήσεις σε Prolog που προτείνονται στις σημειώσεις, γράψτε ένα πρόγραμμα Prolog που να μετατρέπει μια βάση γνώσης που ακολουθεί τη μεθοδολογία των σημασιολογικών δικτύων σε μια αντίστοιχη βάση γνώσης που ακολουθεί εκείνη των πλαισίων. Επίσης, γράψτε κι ένα πρόγραμμα Prolog για το αντίστροφο. Θεωρήστε ότι οι Prolog προτάσεις που αντιστοιχούν σε μια βάση γνώσης δίνονται μέσω λίστας (αντί για πρόγραμμα σε αρχείο). Η περίπτωση των σχισμών πλαισίων με τιμές διαδικασίες να μην καλυφθεί, καθώς και η διάκριση στιγμιοτύπων και υποκλάσεων στα σημασιολογικά δίκτυα.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2018

1. (α΄) Τι απάντηση θα δώσει ένα σύστημα Prolog στην εξής ερώτηση;

```
?- [[X,a],X,Y|Z] = [Y,b,[b|W],c,W].
```

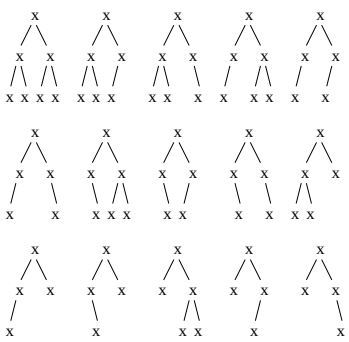
(β΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `packcons(List1,List2)`, το οποίο για δεδομένη λίστα `List1`, να επιστρέφει στη μεταβλητή `List2` τη λίστα που έχει ως στοιχεία τις λίστες των **συνεχόμενων** εμφανίσεων κάθε στοιχείου στη λίστα `List1`. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- packcons([a,a,a,b,c,c,d,d,d,d,a,e,c,c],List).
List = [[a,a,a],[b],[c,c],[d,d,d,d],[a],[e],[c,c]]
?- packcons([f(1),g(2),g(2),g(1),f(1),f(1),g(2)],List).
List = [[f(1)],[g(2),g(2)],[g(1)],[f(1),f(1)],[g(2)]]
```

Η υλοποίησή σας αρκεί να δουλεύει σωστά όταν τα στοιχεία της `List1` δεν περιέχουν μεταβλητές. Παρ΄ όλα αυτά, ποιες απαντήσεις θα παίρνατε αν στο πρόγραμμα που γράψατε υποβάλλατε τις παρακάτω ερωτήσεις;

```
?- packcons([f(1),f(X),f(2),f(Y),f(2)],List).
.....................
?- packcons([f(X),f(Y),g(X),g(1),g(2),g(Y)],List).
.....................
```

2. (α΄) Στο σχήμα φαίνονται όλα τα πιθανά δυαδικά δέντρα βάθους 3 (με πληροφορία σε κάθε κόμβο απλώς το σύμβολο `x`), τα οποία έχουν την επιπλέον ιδιότητα να είναι και *ισοζυγισμένα καθ΄ ύψος – ΙΚΥ* (height balanced – HB). Ένα δυαδικό δέντρο είναι ΙΚΥ, όταν για κάθε κόμβο του ισχύει ότι τα ύψη των δύο υποδέντρων του διαφέρουν το πολύ κατά 1. Κατ΄ αρχήν σχεδιάστε όλα τα πιθανά ΙΚΥ δυαδικά δέντρα βάθους 2 (με πληροφορία στους κόμβους το `x`). Στη συνέχεια, ορίστε σε Prolog ένα κατηγόρημα `hbaltr(H,T)`, το οποίο όταν καλείται με πρώτο όρισμα `H` ένα μη αρνητικό ακέραιο, να επιστρέφει στη μεταβλητή `T` μέσω οπισθοδρόμησης όλα τα ΙΚΥ δυαδικά δέντρα βάθους `H` (με πληροφορία στους κόμβους το `x`). Ως αναπαράσταση των δυαδικών δέντρων, χρησιμοποιήστε την προτεινόμενη στις σημειώσεις/διαφάνειες του μαθήματος, όπου το κενό δέντρο (βάθους 0) συμβολίζεται με το `nil` και το δέντρο με ρίζα τον κόμβο `X` και υποδέντρα τα `L` και `R` (αριστερό και δεξιό, αντίστοιχα) αναπαρίσταται από τον όρο Prolog `t(L,X,R)`. Κάποια παραδείγματα εκτέλεσης:



```
?- hbaltr(0,T).
T = nil

?- hbaltr(1,T).
T = t(nil,x,nil)

?- hbaltr(2,T).
T = t(t(nil,x,nil),x,t(nil,x,nil))   --> ;
T = t(t(nil,x,nil),x,nil)            --> ;
T = t(nil,x,t(nil,x,nil))
```

```
?- hbaltr(3,T).
T = t(t(t(nil,x,nil),x,t(nil,x,nil)),x,t(t(nil,x,nil),x,t(nil,x,nil)))   --> ;
T = t(t(t(nil,x,nil),x,t(nil,x,nil)),x,t(t(nil,x,nil),x,nil))            --> ;
T = t(t(t(nil,x,nil),x,t(nil,x,nil)),x,t(nil,x,t(nil,x,nil)))            --> ;
T = t(t(t(nil,x,nil),x,nil),x,t(t(nil,x,nil),x,t(nil,x,nil)))            --> ;
T = t(t(t(nil,x,nil),x,nil),x,t(t(nil,x,nil),x,nil))                     --> ;
T = t(t(t(nil,x,nil),x,nil),x,t(nil,x,t(nil,x,nil)))                     --> ;
T = t(t(nil,x,t(nil,x,nil)),x,t(t(nil,x,nil),x,t(nil,x,nil)))            --> ;
.....................................................................
```

(β′) Στην προτασιακή λογική, ένας τύπος είναι σε *συζευκτική κανονική μορφή – ΣΚΜ* (conjunctive normal form – CNF), όταν είναι μία σύζευξη διαζεύξεων, όπου κάθε διάζευξη, που ονομάζεται και *πρόταση* (clause), περιέχει προτασιακά σύμβολα ή αρνήσεις προτασιακών συμβόλων. Για παράδειγμα, ο ακόλουθος τύπος είναι σε ΣΚΜ:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_4) \wedge (x_2 \vee \neg x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_3 \vee x_4)$$

Το πρόβλημα της *μέγιστης ικανοποιησιμότητας – ΜΕΓΙΚ* (maximum satisfiability – MAXSAT) έγκειται στην εύρεση κατάλληλων αναθέσεων τιμών στα προτασιακά σύμβολα (`true` ή `false`) έτσι ώστε να μεγιστοποιείται το πλήθος των προτάσεων του τύπου που αληθεύουν. Για παράδειγμα, στον παραπάνω τύπο, που αποτελείται από 7 προτάσεις, δεν υπάρχει ανάθεση τιμών στα προτασιακά σύμβολα που να κάνει αληθείς όλες τις προτάσεις, αλλά μπορεί να βρεθεί ανάθεση ώστε να αληθεύουν 6 από αυτές. Κατ′ αρχήν, μπορείτε να βρείτε εμπειρικά μία τέτοια ανάθεση, που κάνει 6 από τις προτάσεις του τύπου αληθείς;

Στη συνέχεια, μοντελοποιήστε το πρόβλημα ΜΕΓΙΚ σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους, τους ενδεχόμενους περιορισμούς μεταξύ τους και τη συνάρτηση κόστους. Η διατύπωση των περιορισμών και της συνάρτησης κόστους θα πρέπει να γίνει με μαθηματική ακρίβεια. Για να διευκολυνθείτε, δώστε την απάντησή σας με βάση τον παραπάνω τύπο. Αν σας ζητείτο να κάνετε και την υλοποίηση στην ECL^{i}PS^e, ποια δυνατότητά της, εκτός από τα προφανή κατηγορήματα αναζήτησης της `ic` και βελτιστοποίησης της `branch_and_bound`, πιστεύετε ότι θα σας φαινόταν ιδιαίτερα χρήσιμη;

3. (α′) Γράψτε το **ελάχιστο** δυνατό οριστικό πρόγραμμα P (ως προς το πλήθος των χαρακτήρων που θα χρησιμοποιήσετε), το οποίο να έχει ως ελάχιστο μοντέλο το σύνολο:

$$M_P = \{\texttt{p(a)}, \texttt{p(f(f(a)))}, \texttt{p(f(f(f(f(a)))))}, \texttt{p(f(f(f(f(f(a))))))}, \texttt{q(f(a))}, \texttt{q(f(f(f(a))))}, \texttt{q(f(f(f(f(f(a))))))}\}$$

Ποια θα ήταν η προφανής απάντηση στο ερώτημα, αν δεν υπήρχε η απαίτηση του "ελάχιστου προγράμματος"; Δώστε και ένα άλλο μοντέλο του προγράμματος, που να είναι απειροσύνολο, αλλά να μην είναι η βάση Herbrand.

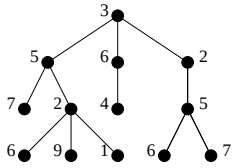
(β′) Για την υλοποίηση της εμπρόσθιας συλλογιστικής της μεθόδου αναπαράστασης με if-then κανόνες, χρησιμοποιήσαμε το ενσωματωμένο κατηγόρημα `assert/1`. Προτείνετε ένα διαφορετικό τρόπο υλοποίησης. Για την υλοποίηση που θα προτείνετε, σχολιάστε πού/αν είναι καλύτερη ή χειρότερη.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2018

1. (α΄) Τι απαντήσεις θα δώσει ένα σύστημα Prolog στις παρακάτω ερωτήσεις; Αιτιολογήστε την απάντησή σας. Αν πιστεύετε ότι κάποια (ή κάποιες) από τις ερωτήσεις έχουν συντακτικό λάθος ή θα δώσουν λάθος εκτέλεσης, δεν είναι απαραίτητο να γράψετε το ακριβές μήνυμα που θα δώσει κάποιο σύστημα Prolog. Αρκεί να περιγράψετε το πρόβλημα που υπάρχει.

```
?- member(X, [1,2,3,4]), member(Y, [2,3,4,5]), X > Y.
?- member(X, [1,2,3,4]), X < Y, member(Y, [2,3,4,5]).
?- X = Y, member(X, [1,2,3,4]), member(Y, [2,3,4,5]).
```

(β΄) Προτείνετε έναν τρόπο αναπαράστασης δέντρων (όχι απαραίτητα δυαδικών), όπως αυτό του διπλανού σχήματος, σαν σύνθετους όρους Prolog. Δώστε την αναπαράσταση του διπλανού δέντρου, σύμφωνα με την πρότασή σας. Επίσης, ορίστε ένα κατηγόρημα Prolog το οποίο να αληθεύει όταν ένα στοιχείο αποτελεί κόμβο ενός δέντρου. Το κατηγόρημα αυτό να μπορεί να χρησιμοποιηθεί είτε για να ελεγχθεί αν δεδομένο στοιχείο ανήκει στο δέντρο είτε για να γεννηθούν μέσω οπισθοδρόμησης όλα τα στοιχεία του δέντρου.



2. (α΄) Έστω ότι δίνονται τα παρακάτω γεγονότα Prolog:

```
activity(1,  act(0,3)).     activity(2,  act(0,4)).     activity(3,  act(1,5)).
activity(4,  act(4,6)).     activity(5,  act(6,8)).     activity(6,  act(6,9)).
activity(7,  act(9,10)).    activity(8,  act(9,13)).    activity(9,  act(11,14)).
activity(10, act(12,15)).   activity(11, act(14,17)).   activity(12, act(16,18)).
activity(13, act(17,19)).   activity(14, act(18,20)).   activity(15, act(19,20)).
```

Τα γεγονότα `activity/2` κωδικοποιούν δραστηριότητες που θα πρέπει να στελεχωθούν από άτομα. Κάθε γεγονός `activity(<A>, act(<S>,<E>))` σημαίνει ότι η δραστηριότητα `<A>` αρχίζει τη χρονική στιγμή `<S>` και τελειώνει τη χρονική στιγμή `<E>`. Υποθέστε ότι κάθε δραστηριότητα πρέπει να στελεχωθεί από ένα ακριβώς άτομο και ότι κάθε άτομο μπορεί να αναλάβει όσες δραστηριότητες απαιτούνται, αρκεί ο συνολικός χρόνος εργασίας του να μην υπερβαίνει κάποιο καθορισμένο όριο. Επίσης, υπάρχει και ο περιορισμός ότι όχι μόνο δεν μπορεί ένα άτομο να αναλάβει δύο δραστηριότητες που επικαλύπτονται χρονικά, αλλά θα πρέπει μεταξύ δύο οποιωνδήποτε διαδοχικών δραστηριοτήτων κάθε ατόμου να μεσολαβεί τουλάχιστον μία μονάδα χρόνου. Με τα δεδομένα αυτά, γράψτε ένα πρόγραμμα Prolog, για παράδειγμα μέσω της υλοποίησης ενός κατηγορήματος `assignment/3`, το οποίο, όταν καλείται ως `assignment(<NP>, <ST>, <AS>)`, όπου `<NP>` είναι το πλήθος των διαθέσιμων ατόμων και `<ST>` είναι ο μέγιστος συνολικός χρόνος των δραστηριοτήτων που μπορεί να αναλάβει ένα άτομο, να αναθέτει τις δοθείσες δραστηριότητες στα δοθέντα άτομα με εφικτό τρόπο, επιστρέφοντας στη μεταβλητή `<AS>` τις αναθέσεις με κατάλληλη κωδικοποίηση, για παράδειγμα σαν μία λίστα από στοιχεία της μορφής `<A>-<P>`, που σημαίνουν ότι η δραστηριότητα `<A>` ανατίθεται στο άτομο `<P>`. Παραδείγματα ερωτήσεων και των αντίστοιχων απαντήσεων στο πρόγραμμα:

```
?- assignment(3, 14, A).
A = [1 - 1, 2 - 2, 3 - 3, 4 - 1, 5 - 2, 6 - 3, 7 - 2, 8 - 1,
     9 - 2, 10 - 3, 11 - 1, 12 - 2, 13 - 3, 14 - 1, 15 - 2]    --> ;
.....................................................................
```

```
?- assignment(2, 40, A).
No

?- assignment(3, 13, A).
No
```

(β′) Μοντελοποιήστε το πρόβλημα του προηγούμενου ερωτήματος 2.(α′) σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους και τους περιορισμούς μεταξύ τους. Η διατύπωση των περιορισμών θα πρέπει να γίνει με μαθηματική αυστηρότητα. Δεν ζητείται να δώσετε κάποια υλοποίηση. Για την ακρίβεια, απαντήσεις μέσω κώδικα δεν θα ληφθούν υπόψη. Τέλος, μπορείτε να επεκτείνετε το πρόβλημα ώστε να μετατραπεί σε πρόβλημα βελτιστοποίησης με κάποια εύλογη συνάρτηση κόστους; Διατυπώστε μαθηματικά τη συνάρτηση κόστους που προτείνετε.

3. (α′) Γράψτε το **ελάχιστο** δυνατό οριστικό πρόγραμμα P (ως προς το πλήθος των χαρακτήρων που θα χρησιμοποιήσετε), το οποίο να έχει ως ελάχιστο μοντέλο το σύνολο:

$$M_P = \{\texttt{p(a)}, \texttt{p(f(f(a)))}, \texttt{p(f(f(f(f(a)))))}, \texttt{p(f(f(f(f(f(a))))))}, \\ \texttt{q(f(a))}, \texttt{q(f(f(f(a))))}, \texttt{q(f(f(f(f(f(a))))))}\}$$

Ποια θα ήταν η προφανής απάντηση στο ερώτημα, αν δεν υπήρχε η απαίτηση του "ελάχιστου προγράμματος"; Δώστε και ένα άλλο μοντέλο του προγράμματος, που να είναι απειροσύνολο, αλλά να μην είναι η βάση Herbrand.

(β′) Θεωρήστε ένα παράδειγμα βάσης γνώσης (διαφορετικού κόσμου από εκείνον του παραδείγματος των σημειώσεων) για το οποίο η αναπαράσταση μέσω σημασιολογικών δικτύων είναι κατάλληλη, φτιάχνοντας το σημασιολογικό δίκτυο που το αναπαριστάνει. Στη συνέχεια:

- Αιτιολογήστε γιατί τα σημασιολογικά δίκτυα είναι η κατάλληλη μεθοδολογία αναπαράστασης για το παράδειγμά σας.
- Δώστε την υλοποίηση του δικτύου αυτού σε Prolog.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2017

1. (α΄) *"Στην Prolog, η απόδειξη ότι δεν αληθεύει ένας ισχυρισμός P γίνεται μέσω της εφαρμογής της διαδικασίας της ανάλυσης, έτσι ώστε να προκύψει ότι τελικά συμπεραίνεται η άρνηση του ισχυρισμού, δηλαδή το $\neg P$"*. Σωστό ή λάθος; Σχολιάστε πολύ σύντομα.

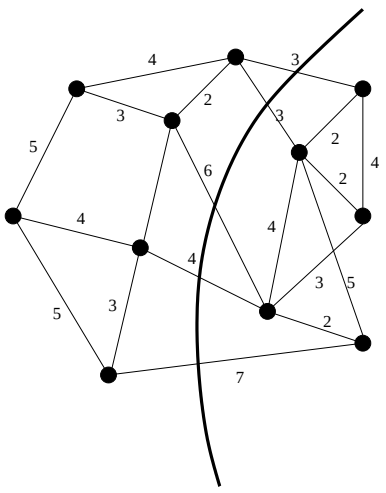
   (β΄) Με δεδομένη τη γνωστή υλοποίηση του κατηγορήματος `append/3` της Prolog για τη συνένωση δύο λιστών, συμπληρώστε κατάλληλα τα κενά στην παρακάτω διαλογική επικοινωνία.

```
?- append(.........., .........., [sun,mon,tue,wed,thu,fri,sat]).
FirstDay = sun
LastDay = sat
?- append(.........., .........., [sun,mon,tue,wed,thu,fri,sat]).
AllBeforeTuesday = [sun,mon]
AllAfterTuesday = [wed,thu,fri,sat]
?- append(.........., .........., [sun,mon,tue,wed,thu,fri,sat]).
JustBeforeFriday = thu
JustAfterFriday = sat
?- append(.........., .........., [sun,mon,tue,wed,thu,fri,sat]),
   append(.........., .........., SomeDays).
SomeDays = ..........
AllBetweenMondayAndFriday = [tue,wed,thu]
```

2. (α΄) Ορίστε σε Prolog ένα κατηγόρημα `spliteq/2`, το οποίο όταν καλείται με πρώτο όρισμα μία **μη κενή** λίστα L, να επιστρέφει στο δεύτερο όρισμα μία λίστα από λίστες (μία ή περισσότερες) ίσου μήκους (αν είναι δύο ή περισσότερες), τέτοιες ώστε αν συνενωθούν, με τη σειρά που εμφανίζονται, να προκύπτει η λίστα L. Το κατηγόρημα να επιστρέφει μέσω οπισθοδρόμησης όλες τις εναλλακτικές λύσεις. Παραδείγματα εκτέλεσης είναι τα εξής:

```
?- spliteq([a,b,c,d,e,f], SL).
SL = [[a],[b],[c],[d],[e],[f]]      -> ;
SL = [[a,b],[c,d],[e,f]]            -> ;
SL = [[a,b,c],[d,e,f]]              -> ;
SL = [[a,b,c,d,e,f]]
?- spliteq([1,2,3,4,5,6,7], SL).
SL = [[1],[2],[3],[4],[5],[6],[7]]  -> ;
SL = [[1,2,3,4,5,6,7]]
```

   (β΄) Έστω ένας γράφος με βάρη στις ακμές. Μία εκδοχή του προβλήματος διαμέρισης γράφου (graph partitioning) είναι να διασπάσουμε τον γράφο σε δύο υπογράφους, με πλήθη κόμβων στους δύο υπογράφους ίσα ή να διαφέρουν κατά ένα. Το κόστος μίας διαμέρισης ισούται με το άθροισμα των βαρών των ακμών του αρχικού γράφου που πρέπει να διασπασθούν για να γίνει διαμέριση. Για παράδειγμα, το κόστος της διαμέρισης που φαίνεται στο διπλανό σχήμα ισούται με 23 (= 3+3+6+4+7). Το ζητούμενο είναι να βρεθεί διαμέριση με ελάχιστο κόστος. Μοντελοποιήστε το πρόβλημα αυτό (βέλτιστη διαμέριση γράφου με ίσα ή διαφέροντα κατά ένα πλήθη κόμβων στους προκύπτοντες γράφους) σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους, τους περιορισμούς μεταξύ τους και τη συνάρτηση κόστους. Η διατύπωση των περιορισμών και της συνάρτησης



κόστους θα πρέπει να γίνει με μαθηματική ακρίβεια. Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α′) i. "*Το αποτέλεσμα της μεταγλώττισης ενός προγράμματος Prolog σε WAM διαφοροποιείται ανάλογα με την αρχιτεκτονική του επεξεργαστή του υπολογιστή στον οποίο θα χρησιμοποιηθεί το πρόγραμμα*". Σωστό ή λάθος; Σχολιάστε πολύ σύντομα.

ii. Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
q(f(X)) :- p(X).
r(f(X)) :- q(X).
p(f(X)) :- r(X).
s(X) :- p(X), r(f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για τον σκοπό αυτό. Μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(a))`.

(β′) i. Λέμε ότι τα έμπειρα συστήματα είναι επιθυμητό να μπορούν να απαντούν σε ερωτήσεις "*πώς*" (how) και "*γιατί*" (why). Σε τι αναφέρονται οι ερωτήσεις αυτές;

ii. Δίνεται το παρακάτω ημιτελές πρόγραμμα Prolog, το οποίο έχει σαν στόχο να πραγματοποιεί συλλογιστική κατά την οπίσθια φορά (backward reasoning) για αναπαράσταση γνώσης με χρήση if-then κανόνων. Συγκεκριμένα, το κατηγόρημα `is_true/1` δέχεται σαν όρισμα έναν στόχο τον οποίο αποπειράται να επιβεβαιώσει με τον παραπάνω τρόπο. Προκειμένου να το επιτύχει, χρησιμοποιεί ένα κατηγόρημα `is_true/2`, το οποίο, πέρα από τον στόχο αυτόν, δέχεται σαν δεύτερο όρισμα το ιστορικό των στόχων που προηγούμενες κλήσεις του έχουν επισκεφθεί. Συμπληρώστε την υλοποίηση του προγράμματος, επεκτείνοντας κατάλληλα τον ορισμό του κατηγορήματος `is_true/2` ώστε να επιτυγχάνει το παραπάνω για κανόνες με προϋποθέσεις απλές προτάσεις (propositions) — όχι σύνθετες λογικές εκφράσεις —. Επίσης, ζητείται να υποστηρίζει την δυνατότητα να δίνει ο χρήστης την τιμή αλήθειας προτάσεων καθώς και να παρέχονται απαντήσεις σε ερωτήσεις "*γιατί*". Θεωρήστε ότι οι προτάσεις των οποίων η τιμή αλήθειας μπορεί να δοθεί από τον χρήση, προσδιορίζονται με χρήση ενός κατηγορήματος `askable/1`.

```
:- op(800, fx, if).
:- op(700, xfx, then).

is_true(P) :- is_true(P, [P]).

is_true(P, _) :- fact(P), !.
is_true(P, SoFar) :- if Cond then P, .........

answer(yes, _).
answer(no, _) :- fail.
answer(why, SoFar) :- write("I need it to prove "), writeln(SoFar),
                writeln("Is it true? "), read(NA), answer(NA,SoFar).
```

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2017

1. (α΄) Γνωρίζετε ότι η Prolog υλοποιεί ένα υποσύνολο της λογικής πρώτης τάξης. Ποια είναι ακριβώς η "έκπτωση" που κάνει σε σχέση με ό,τι υποστηρίζεται από τη λογική; Τι κερδίζει και τι χάνει; Σχετίζεται κάπως με το θέμα ο χειρισμός της άρνησης από την Prolog; Σχολιάστε πολύ σύντομα (μέγιστο 10 γραμμές).

   (β΄) Έστω ο εξής ορισμός του κατηγορήματος `flatten/2`, όπως τον ορίσαμε στο μάθημα:

```
flatten([], []) :- !.
flatten([L1|L2], FL) :- !, flatten(L1, FL1), flatten(L2, FL2), append(FL1, FL2, FL).
flatten(X, [X]).
```

   Τι θα θέλατε να πάρετε σαν απάντηση στην παρακάτω ερώτηση και τι πραγματικά παίρνετε με την εν λόγω υλοποίηση;

```
?- flatten([[X,Y,a],Z,[[[b]]],[[c,W],U]], FL).
```

   Δώστε έναν ορισμό του `flatten/2`, ώστε να λειτουργεί με τον επιθυμητό τρόπο ακόμα και για την ερώτηση που δόθηκε προηγουμένως.

2. (α΄) Θεωρήστε την αναπαράσταση λογικών συναρτήσεων που εμπλέκουν τους γνωστούς τελεστές `and`, `or` και `not` ως όρων Prolog, στους οποίους χρησιμοποιούνται οι τελεστές αυτοί ως συναρτησιακά σύμβολα και οι ανεξάρτητες μεταβλητές των συναρτήσεων αναπαρίστανται ως μεταβλητές Prolog. Για παράδειγμα, μία τέτοια λογική συνάρτηση τεσσάρων μεταβλητών θα μπορούσε να ήταν η εξής:

```
and(not(or(A,B)),and(or(B,not(C)),not(and(A,D))))
```

   Γνωρίζετε πώς να υπολογίσετε την τιμή μίας λογικής συνάρτησης για μία πιθανή αποτίμηση των μεταβλητών της από το σύνολο τιμών {`true`,`false`}. Ορίστε σε Prolog ένα κατηγόρημα `table/2`, το οποίο, όταν καλείται με πρώτο όρισμα μία λογική συνάρτηση και δεύτερο όρισμα τη λίστα των μεταβλητών της συνάρτησης, να εκτυπώνει τον "πίνακα αλήθειας" της συνάρτησης. Δηλαδή, για κάθε πιθανό συνδυασμό τιμών των μεταβλητών της να βρίσκει και να εκτυπώνει την τιμή της συνάρτησης. Ένα παράδειγμα εκτέλεσης:

```
?- table(and(not(or(A,B)),and(or(B,not(C)),not(and(A,D)))), [A,B,C,D]).

  Var1  Var2  Var3  Var4 Value

  true  true  true  true false
  true  true  true false false
  true  true false  true false
  true  true false false false
  true false  true  true false
  true false  true false false
  true false false  true false
  true false false false false
 false  true  true  true false
 false  true  true false false
 false  true false  true false
 false  true false false false
 false false  true  true false
 false false  true false false
 false false false  true  true
 false false false false  true
```

(β′) Μία εταιρεία που λειτουργεί 24 ώρες την ημέρα και 7 ημέρες την εβδομάδα απασχολεί 50 υπαλλήλους. Προφανώς, οι υπάλληλοι πρέπει να εργάζονται σε βάρδιες. Κάθε υπάλληλος μπορεί, για κάποια ημέρα, να εργάζεται σε πρωινή βάρδια (M), ή απογευματινή (E), ή νυχτερινή (N), ή μπορεί να έχει ανάπαυση (R). Η εταιρεία θέλει να δημιουργήσει ένα πρόγραμμα εργασίας των υπαλλήλων της για ένα διάστημα 30 συνεχόμενων ημερών, στο οποίο όμως να ισχύουν οι εξής κανόνες:

- Κάθε ημέρα πρέπει να εργάζονται τουλάχιστον 15 υπάλληλοι στην πρωινή βάρδια, τουλάχιστον 10 στην απογευματινή και τουλάχιστον 8 στην νυχτερινή.
- Κανένας υπάλληλος δεν επιτρέπεται αν μία ημέρα είχε νυχτερινή βάρδια, την επόμενη να έχει πρωινή.
- Κάθε υπάλληλος πρέπει σε οποιεσδήποτε 7 συνεχόμενες ημέρες του προγράμματος εργασίας (από την 1η έως την 7η, από την 2η έως την 8η, κ.ο.κ.) να έχει τουλάχιστον δύο ημέρες ανάπαυσης.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και επειδή, προφανώς, μπορεί να έχει πολλές λύσεις, προτείνετε μία εύλογη αντικειμενική συνάρτηση που να ποσοτικοποιεί την ποιότητα (ή το κόστος) μίας λύσης, οπότε πλέον έχουμε ένα πρόβλημα βελτιστοποίησης.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
q(f(X)) :- p(X).
r(f(X)) :- q(X).
p(f(X)) :- r(X).
s(X) :- p(X), r(f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για τον σκοπό αυτό. Μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(a))`.

(β′) Δώστε ένα παράδειγμα γνωστικού πεδίου όπου ενδείκνυται η χρήση σημασιολογικών δικτύων σαν μεθοδολογία αναπαράστασης γνώσης και δώστε μια μικρή βάση γνώσης από αυτό σε Prolog (15 προτάσεις), διαφορετικό από το παράδειγμα των σημειώσεων. Στη συνέχεια, επαυξήστε το με κανόνες if-then. Τι ρόλο μπορεί να παίξει η Prolog στην υλοποίηση;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2016

1. (α΄) Γνωρίζετε το ενσωματωμένο κατηγόρημα `name/2` της Prolog, το οποίο είναι σε θέση να αντιστοιχεί μία σταθερά (άτομο) Prolog στη λίστα των ASCII κωδικών των χαρακτήρων της. Παράδειγμα:

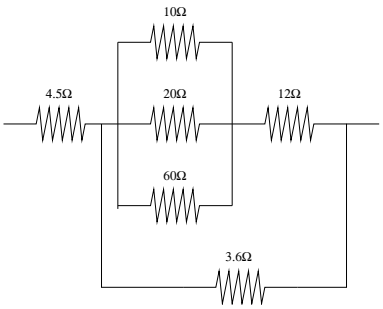
```
?- name('ab-cd', Chars), name(Atom, [65, 66, 67, 68]).
Chars = [97, 98, 45, 99, 100]
Atom = 'ABCD'
```

Ορίστε ένα κατηγόρημα `tokenize/2`, το οποίο όταν καλείται με πρώτο όρισμα μία σταθερά που αντιπροσωπεύει μία φράση φυσικής γλώσσας, δηλαδή μία ακολουθία από λέξεις χωρισμένες με έναν κενό χαρακτήρα, να επιστρέφει στο δεύτερο όρισμα μία λίστα από τις λέξεις της φράσεις, ως σταθερές. Ένα παράδειγμα εκτέλεσης:

```
?- tokenize('This is a test phrase/sentence', Words).
Words = ['This', is, a, test, 'phrase/sentence']
```

Μπορείτε να θεωρήσετε δεδομένο ότι μεταξύ κάθε διαδοχικών λέξεων της φράσης υπάρχει ακριβώς ένας κενός χαρακτήρας και ότι δεν υπάρχουν κενοί χαρακτήρες στην αρχή ή στο τέλος της φράσης. Δεν μπορείτε όμως να θεωρήσετε δεδομένους συγκεκριμένους ASCII κωδικούς και, συγκεκριμένα, αυτόν του κενού χαρακτήρα.

(β΄) Στην Prolog μπορούμε να αναπαραστήσουμε ένα σύνθετο κύκλωμα αντιστάσεων μέσω δύο συναρτησιακών συμβόλων, έστω του `seq/2` και του `par/2`, τα οποία κωδικοποιούν την σε σειρά και την παράλληλη σύνδεση δύο αντιστάσεων, με ορίσματα τις τιμές των αντιστάσεων αυτών. Για παράδειγμα, το διπλανό κύκλωμα θα μπορούσε να αναπαρασταθεί από τον όρο Prolog:




```
seq(4.5,par(seq(par(10,par(20,60)),12),3.6))
```

Ορίστε ένα κατηγόρημα `totres/2`, το οποίο όταν καλείται με πρώτο όρισμα έναν όρο που αναπαριστά ένα σύνθετο κύκλωμα αντιστάσεων, να επιστρέφει στο δεύτερο όρισμα την ισοδύναμη ολική αντίσταση του κυκλώματος.[1] Μία ενδεικτική εκτέλεση είναι η εξής:

```
?- totres(seq(4.5,par(seq(par(10,par(20,60)),12),3.6)), Rtot).
Rtot = 7.5
```

2. (α΄) Υπάρχουν οκτώ βάτραχοι, τέσσερις καφέ και τέσσερις πράσινοι, και εννέα πέτρες στη σειρά σε μία λίμνη, όπως φαίνεται στη διπλανή εικόνα. Οι καφέ βάτραχοι βρίσκονται στις τέσσερις δεξιά πέτρες και οι πράσινοι στις τέσσερις αριστερά. Ο στόχος τους είναι να ανταλλάξουν θέσεις, δηλαδή οι καφέ να πάνε στις τέσσερις αριστερά πέτρες και οι πράσινοι στις τέσσερις δεξιά. Ένας καφέ βάτραχος μπορεί να κινηθεί μόνο προς τα

[1] Γνωρίζετε από τη Φυσική ότι δύο αντιστάσεις R_1 και R_2 έχουν ολική αντίσταση R_{tot}, που δίνεται από τον τύπο $R_{tot} = R_1 + R_2$, όταν οι αντιστάσεις είναι συνδεδεμένες σε σειρά, και από τον τύπο $\frac{1}{R_{tot}} = \frac{1}{R_1} + \frac{1}{R_2}$, όταν είναι συνδεδεμένες παράλληλα.

αριστερά, είτε μετακινούμενος στη κενή πέτρα, αν αυτή είναι ακριβώς μπροστά του, είτε πηδώντας επάνω από έναν πράσινο βάτραχο, αν αυτός είναι ακριβώς μπροστά του και ακριβώς πίσω από τον πράσινο βάτραχο βρίσκεται η κενή πέτρα. Ομοίως, ένας πράσινος βάτραχος μπορεί να κινηθεί μόνο προς τα δεξιά, είτε μετακινούμενος στη κενή πέτρα, αν αυτή είναι ακριβώς μπροστά του, είτε πηδώντας επάνω από έναν καφέ βάτραχο, αν αυτός είναι ακριβώς μπροστά του και ακριβώς πίσω από τον καφέ βάτραχο βρίσκεται η κενή πέτρα.

Μοντελοποιήστε σε Prolog μία ενδεχόμενη κατάσταση του προβλήματος και ορίστε κατάλληλα κατηγορήματα `initial_state/1`, `final_state/1` και `move/2`, τα οποία να περιγράφουν την αρχική κατάσταση, την τελική κατάσταση και τις δυνατές μεταβάσεις μεταξύ καταστάσεων, αντίστοιχα, έτσι ώστε σε συνδυασμό με μία υλοποίηση της πρώτα κατά βάθους μεθόδου αναζήτησης, όπως αυτή δίνεται στη σελ. 148 των σημειώσεων του μαθήματος, να είναι δυνατόν να αντιμετωπισθεί το πρόβλημα. Η υλοποίησή σας αρκεί να επιλύει απλώς την περίπτωση των οκτώ βατράχων, όμως θα πρέπει να μπορεί εύκολα να τροποποιηθεί ώστε να μπορεί να χρησιμοποιηθεί και για οποιοδήποτε, οσοδήποτε μεγάλο, πλήθος βατράχων.

(β′) Έστω ότι θέλετε να βρείτε ένα μέγιστο υποσύνολο μελών του Facebook, τέτοιο ώστε κάθε δύο στοιχεία του υποσυνόλου να είναι φίλοι. Έχετε σαν δεδομένα ποια είναι τα μέλη του δικτύου και ποια ζευγάρια έχουν σχέση φιλίας μεταξύ τους. Μοντελοποιήστε το πρόβλημα αυτό (εύρεση μέγιστης κλίκας σε γράφο) σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους, τους περιορισμούς μεταξύ τους και τη συνάρτηση κόστους. Η διατύπωση των περιορισμών και της συνάρτησης κόστους θα πρέπει να γίνει με μαθηματική ακρίβεια. Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(f(a))))).
p(X) :- q(f(X)).
q(X) :- p(f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(f(f(f(a))))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(f(f(a))))`. Μπορεί να υπάρξει μοντέλο του προγράμματος που να είναι απειροσύνολο και να μην είναι η ίδια η βάση Herbrand; Αν όχι, εξηγήστε γιατί. Αν ναι, δώστε ένα τέτοιο μοντέλο.

(β′) Γνωρίζετε ότι ένας τρόπος υλοποίησης της μεθοδολογίας αναπαράστασης γνώσης μέσω πλαισίων στην Prolog είναι να χρησιμοποιείται, για κάθε πλαίσιο, ένα κατηγόρημα το οποίο ορίζεται από ένα σύνολο από γεγονότα. Για κάθε σχισμή του πλαισίου, αντιστοιχεί ένα τέτοιο γεγονός. Δώστε έναν διαφορετικό από τον προηγούμενο τρόπο υλοποίησης των πλαισίων. Επίσης, ορίστε ένα κατηγόρημα που να υλοποιεί συμπερασμό για την πρότασή σας.

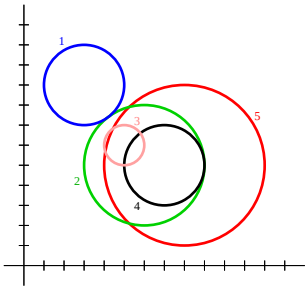
Σημείωση: Δεν είναι απαραίτητο να καλύψετε την περίπτωση σχισμών που αντί για τιμή να περιγράφεται η διαδικασία υπολογισμού της.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2016

1. (α΄) Μία ακολουθία από σύμβολα μπορεί να κωδικοποιηθεί σύμφωνα με τη λογική του *τρέχοντος μήκους* (run length encoding) με τον εξής τρόπο. Κάθε μέγιστη υπο-ακολουθία από το ίδιο σύμβολο S μήκους $N(>1)$ κωδικοποιείται σαν ένα ζευγάρι (S, N). Υπο-ακολουθίες μήκους 1 δεν κωδικοποιούνται, αλλά απλά παραμένει στο αποτέλεσμα το σύμβολο S. Για παράδειγμα, η ακολουθία "a,a,a,b,b,c,d,d,d,d,e" κωδικοποιείται σαν "(a,3),(b,2),c,(d,4),e". Ορίστε ένα κατηγόρημα Prolog `decode_rl/2`, το οποίο όταν καλείται με πρώτο όρισμα μία λίστα που παριστάνει μία κωδικοποιημένη ακολουθία συμβόλων, σύμφωνα με τα παραπάνω, να την αποκωδικοποιεί και να επιστρέφει στο δεύτερο όρισμα το αποτέλεσμα. Τα σύμβολα μπορεί να είναι όροι Prolog, όχι όμως μεταβλητές. Μπορούν να είναι όμως δομές που περιέχουν μεταβλητές. Κάποια παραδείγματα εκτέλεσης:

```
?- decode_rl([(a,3),(b,2),c,(d,4),e], L).
L = [a,a,a,b,b,c,d,d,d,d,e]
?- decode_rl([(f(5,a),7)], L).
L = [f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a)]
?- decode_rl([g(X),(h(Y),3),k(Z),(m(W),4),n(U)], L).
L = [g(X),h(Y),h(Y),h(Y),k(Z),m(W),m(W),m(W),m(W),n(U)]
```

(β΄) Ένας κύκλος στο επίπεδο καθορίζεται πλήρως από τις συντεταγμένες του κέντρου του και την ακτίνα του. Έτσι, το σύνολο των κύκλων του διπλανού σχήματος θα μπορούσε να αναπαρασταθεί από τα γεγονότα που ακολουθούν, όπου το πρώτο όρισμα του `circle/3` είναι η ταυτότητα του κύκλου, το δεύτερο οι συντεταγμένες του κέντρου του και το τρίτο το μήκος της ακτίνας του.




```
circle(1, p(3,9), 2).      circle(2, p(6,5), 3).
circle(3, p(5,6), 1).      circle(4, p(7,5), 2).
circle(5, p(8,5), 4).
```

Ορίστε σε Prolog ένα κατηγόρημα `show_rel_pos/0`, το οποίο, δεδομένων κάποιων κύκλων που έχουν αναπαρασταθεί μέσω του κατηγορήματος `circle/3`, όπως παραπάνω, να εκτυπώνει, για κάθε ζευγάρι κύκλων, τη σχετική θέση των δύο μελών του (εξωτερικοί ο ένας του άλλου, εφάπτονται εξωτερικά, τέμνονται, εφάπτονται εσωτερικά, εσωτερικός ο ένας του άλλου). Ένα παράδειγμα εκτέλεσης για τους κύκλους του σχήματος είναι:

```
?- show_rel_pos.
Circles 1 and 2: touch externally
Circles 1 and 3: are external to each other
Circles 1 and 4: are external to each other
Circles 1 and 5: are external to each other
Circles 2 and 3: one is completely inside the other
Circles 2 and 4: touch internally
Circles 2 and 5: intersect
Circles 3 and 4: intersect
Circles 3 and 5: intersect
Circles 4 and 5: one is completely inside the other
```

2. (α′) Υλοποιήστε σε Prolog την κωδικοποίηση τρέχοντος μήκους, όπως αυτή περιγράφεται στο θέμα 1(α′), ορίζοντας ένα κατηγόρημα `encode_rl/2`, το οποίο να επιτελεί την αντίστροφη λειτουργία από αυτήν του `decode_rl/2`. Δηλαδή να παίρνει σαν πρώτο όρισμα μία λίστα συμβόλων (όρων Prolog πλην μεταβλητών) και να επιστρέφει στο δεύτερο όρισμα την κωδικοποιημένη εκδοχή της. Παραδείγματα:

```
?- encode_rl([a,a,a,b,b,c,d,d,d,d,e], L).
L = [(a,3),(b,2),c,(d,4),e]
?- encode_rl([f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a),f(5,a)], L).
L = [(f(5,a),7)]
?- encode_rl([g(X),h(Y),h(Y),h(Y),k(Z),m(W),m(W),m(W),m(W),n(U)], L).
L = [g(X),(h(Y),3),k(Z),(m(W),4),n(U)]
```

Τι απάντηση θα δοθεί, με βάση τον ορισμό που δώσατε, στην εξής ερώτηση;

```
?- encode_rl([p(3),p(X),q(X),q(Y),q(4)], L).
```

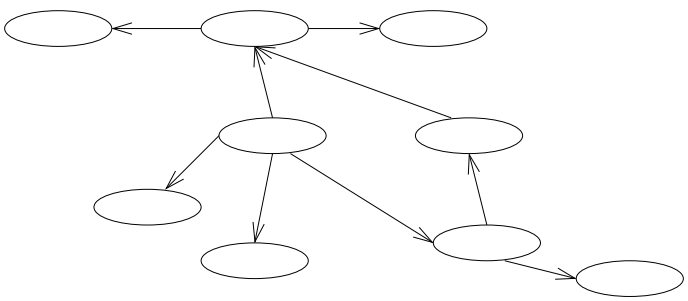
(β′) Έστω ότι θέλετε να βρείτε ένα μέγιστο υποσύνολο μελών του Facebook, τέτοιο ώστε κάθε δύο στοιχεία του υποσυνόλου να είναι φίλοι. Έχετε σαν δεδομένα ποια είναι τα μέλη του δικτύου και ποια ζευγάρια έχουν σχέση φιλίας μεταξύ τους. Μοντελοποιήστε το πρόβλημα αυτό (εύρεση μέγιστης κλίκας σε γράφο) σαν πρόβλημα ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους, τους περιορισμούς μεταξύ τους και τη συνάρτηση κόστους. Η διατύπωση των περιορισμών και της συνάρτησης κόστους θα πρέπει να γίνει με μαθηματική ακρίβεια. Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(f(a))))).
p(X) :- q(f(X)).
q(X) :- p(f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(f(f(f(a))))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(f(f(a))))`. Μπορεί να υπάρξει μοντέλο του προγράμματος που να είναι απειροσύνολο και να μην είναι η ίδια η βάση Herbrand; Αν όχι, εξηγήστε γιατί. Αν ναι, δώστε ένα τέτοιο μοντέλο.

(β′) Δώστε ονόματα όπου νομίζετε ότι χρειάζεται ώστε η παρακάτω εικόνα να αναπαριστά μέρος μιας βάσης γνώσης. Ποια μεθοδολογία ακολουθεί η αναπαράσταση; Γράψτε ένα πρόγραμμα Prolog που να υλοποιεί τη γνώση που αναπαριστάνεται στο (συμπληρωμένο) σχήμα σας.



ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2015

1. (α΄) Δίνεται το εξής πρόγραμμα Prolog:

```
p(1).  p(2).  p(5).  p(8).

r(2).  r(4).  r(5).  r(7).  r(9).

s(X) :- p(X), not r(X).
s(X) :- r(X), not p(X).
```

Ποιες απαντήσεις θα δοθούν στην ερώτηση "`?- s(W).`"; Ποια είναι η φυσική σημασία του κατηγορήματος `s/1` από τη σκοπιά της θεωρίας συνόλων και ποια από τη σκοπιά των λογικών πυλών;

(β΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `maxcun/2` έτσι ώστε όταν αυτό καλείται σαν `maxcun(List, MaxSub)`, να βρίσκει τη μέγιστη υπολίστα συνεχόμενων ενοποιήσιμων στοιχείων στη λίστα `List`, **χωρίς όμως να γίνεται η ενοποίηση**, και να επιστρέφει το αποτέλεσμα στη λίστα `MaxSub`. Αν υπάρχουν περισσότερες της μίας υπολίστες μεγίστου μήκους με συνεχόμενα ενοποιήσιμα στοιχεία, να επιστρέφονται όλες οι δυνατές λύσεις μέσω οπισθοδρόμησης. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- maxcun([a,a,b,b,b,b,c,d,d,d,e,e,e,e,f], MaxSub).
MaxSub = [b, b, b, b]    --> ;
MaxSub = [e, e, e, e]
?- maxcun([f(1),f(X),g(X),g(2),g(Y),g(3),g(Z),h(X),h(Y),h(Z)], MaxSub).
MaxSub = [g(X), g(2), g(Y)]   --> ;
MaxSub = [g(Y), g(3), g(Z)]   --> ;
MaxSub = [h(X), h(Y), h(Z)]
?- maxcun([1,2,3], MaxSub).
MaxSub = [1]   --> ;
MaxSub = [2]   --> ;
MaxSub = [3]
?- maxcun([X,Y,Z,W,Y,W], MaxSub).
MaxSub = [X, Y, Z, W, Y, W]
```

2. (α΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `langford/1` έτσι ώστε όταν αυτό καλείται σαν `langford(List)`, να επιστρέφει στη μεταβλητή `List` μία λίστα 14 στοιχείων που περιλαμβάνει τους αριθμούς 1, 2, 3, 4, 5, 6, 7 από δύο φορές τον καθένα, έτσι ώστε μεταξύ των δύο 1, να μεσολαβεί ακριβώς 1 στοιχείο, μεταξύ των δύο 2, να μεσολαβούν ακριβώς 2 στοιχεία και, γενικά, μεταξύ των δύο N, για κάθε N από 1 έως 7, να μεσολαβούν ακριβώς N στοιχεία. Ένα παράδειγμα εκτέλεσης:

```
?- langford(List).
List = [7, 3, 6, 2, 5, 3, 2, 4, 7, 6, 5, 1, 4, 1]   --> ;
List = [7, 2, 6, 3, 2, 4, 5, 3, 7, 6, 4, 1, 5, 1]   --> ;
List = [7, 2, 4, 6, 2, 3, 5, 4, 7, 3, 6, 1, 5, 1]   --> ;
List = [7, 3, 1, 6, 1, 3, 4, 5, 7, 2, 6, 4, 2, 5]   --> ;
.....................................................
```

(β′) Προτείνετε μία μοντελοποίηση του προβλήματος του προηγούμενου ερωτήματος ως προβλήματος ικανοποίησης περιορισμών. Δηλαδή, ορίστε με σαφήνεια τις μεταβλητές του προβλήματος, τα πεδία τους και τους περιορισμούς μεταξύ τους. Η διατύπωση των περιορισμών θα πρέπει να γίνει με μαθηματική ακρίβεια ή με αναφορά σε συνήθη κατηγορήματα περιορισμών που υποστηρίζονται από καθιερωμένα συστήματα προγραμματισμού με περιορισμούς. Στην περίπτωση που το προηγούμενο ερώτημα αντιμετωπίστηκε μέσω προγραμματισμού με περιορισμούς, προτείνετε μία εναλλακτική μοντελοποίηση αυτής που χρησιμοποιήσατε στην υλοποίηση του κατηγορήματος `langford/1`.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα P:

```
p(n,W,W).
p(f(n,X),Y,f(n,Z)) :- p(X,Y,Z).
```

Ανήκει το `p(f(n,f(n,n)),f(n,n),f(n,f(n,f(n,n))))` στο **ελάχιστο μοντέλο** M_P του προγράμματος ή όχι; Αιτιολογήστε την απάντησή σας. Αν M_I είναι η τομή **όλων των μοντέλων** στα οποία ανήκει το `p(n,n,f(n,n))`, με τι ισούται η διαφορά $M_I - M_P$;

(β′) Ποια μεθοδολογία αναπαράστασης γνώσης στοχεύει στην αναπαράσταση σχέσεων; Ποια σχέση είναι εκείνη που χρησιμοποιείται για συμπερασμό; Οι if_then κανόνες μπορούν να θεωρηθούν σαν μια ειδική μορφή σχέσης μεταξύ συνθήκης και συμπεράσματος. Υλοποιήστε σε Prolog ένα μετα-διερμηνέα που να μπορεί να βγάλει συμπεράσματα από ένα ενιαίο πλαίσιο αναπαράστασης σχέσεων και if_then κανόνων, θεωρώντας τους κανόνες σαν μια ειδική μορφή σχέσης. Δώστε ένα παράδειγμα βάσης γνώσης που να περιέχει τόσο σχέσεις όσο και if_then κανόνες, στο οποίο να μπορεί να εφαρμοστεί ο μετα-διερμηνέας που υλοποιήσατε.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2015

1. (α΄) Σ΄ έναν αναδρομικό κανόνα Prolog, όταν ο αναδρομικός στόχος είναι ο τελευταίος στο σώμα του κανόνα, λέμε ότι έχουμε *αναδρομή ουράς*. Πιστεύετε ότι είναι προτιμότερο να επιδιώκουμε στην υλοποίηση κάποιου κατηγορήματος να έχουμε αναδρομή ουράς ή όχι; Αιτιολογήστε την απάντησή σας μέσω ενός συγκεκριμένου παραδείγματος υλοποίησης κατηγορήματος Prolog που να αναδεικνύει το πλεονέκτημα ή το μειονέκτημα της χρήσης αναδρομής ουράς.

   (β΄) Έστω ότι έχετε στη διάθεσή σας ένα κατηγόρημα `randint/2`, το οποίο όταν καλείται σαν `randint(N,R)`, με το `N` να είναι θετικός ακέραιος, επιστρέφει στη μεταβλητή `R` ένα τυχαίο ακέραιο στο διάστημα `[1,N]`. Με τη βοήθεια αυτού του κατηγορήματος, υλοποιήστε ένα κατηγόρημα `randperm/2`, το οποίο όταν καλείται σαν `randperm(L1,L2)` με δεδομένη λίστα `L1`, να επιστρέφει στη μεταβλητή `L2` μία λίστα που να είναι μία τυχαία μετάθεση των στοιχείων της `L1`. Για παράδειγμα:

```
?- randperm([a,b,c,d,e,f], L).
L = [d,a,f,e,b,c]
```

   Επίσης, με τη βοήθεια του `randperm/2`, υλοποιήστε και το `randpermcont/2`, το οποίο να συμπεριφέρεται όπως το πρώτο, αλλά να επιστρέφει συνεχώς μέσω οπισθοδρόμησης τυχαίες μεταθέσεις της δεδομένης λίστας. Δεν υπάρχει απαίτηση να μην υπάρχουν επαναλαμβανόμενες απαντήσεις, ούτε είναι απαραίτητο το πλήθος των απαντήσεων να είναι πεπερασμένο. Για παράδειγμα:

```
?- randpermcont([1,2,3,4], L).
L = [3,2,4,1]   --> ;
L = [2,3,4,1]   --> ;
L = [3,1,4,2]   --> ;
L = [1,4,2,3]   --> ;
L = [2,3,4,1]   --> ;
.......................
```

2. (α΄) Δυαδικό λεξικό είναι ένα δυαδικό δέντρο του οποίου κάθε κόμβος είναι "μεγαλύτερος" από όλους τους κόμβους του αριστερού υποδέντρου του και "μικρότερος" από όλους τους κόμβους του δεξιού υποδέντρου του. Οι όροι "μεγαλύτερος" και "μικρότερος" αναφέρονται σε μία δεδομένη σχέση διάταξης. Υιοθετώντας μία κατάλληλη αναπαράσταση δυαδικών δέντρων (άρα και δυαδικών λεξικών), ορίστε σε Prolog ένα κατηγόρημα `dictionary/1` το οποίο να επιτυγχάνει όταν το όρισμα που του δίνεται είναι δυαδικό λεξικό. Φροντίστε η υλοποίηση του κατηγορήματος `dictionary/1` να είναι τέτοια ώστε η χρονική πολυπλοκότητά του να είναι γραμμική ως προς το πλήθος των κόμβων του δυαδικού δέντρου που του δίνεται σαν όρισμα.

   (β΄) Μία εταιρεία που λειτουργεί 24 ώρες την ημέρα και 7 ημέρες την εβδομάδα απασχολεί 50 υπαλλήλους. Προφανώς, οι υπάλληλοι πρέπει να εργάζονται σε βάρδιες. Κάθε υπάλληλος μπορεί, για κάποια ημέρα, να εργάζεται σε πρωινή βάρδια (M), ή απογευματινή (E), ή νυχτερινή (N), ή μπορεί να έχει ανάπαυση (R). Η εταιρεία θέλει να δημιουργήσει ένα πρόγραμμα εργασίας των υπαλλήλων της για ένα διάστημα 30 συνεχόμενων ημερών, στο οποίο όμως να ισχύουν οι εξής κανόνες:

- Κάθε ημέρα πρέπει να εργάζονται τουλάχιστον 15 υπάλληλοι στην πρωινή βάρδια, τουλάχιστον 10 στην απογευματινή και τουλάχιστον 8 στην νυχτερινή.
- Κανένας υπάλληλος δεν επιτρέπεται αν μία ημέρα είχε νυχτερινή βάρδια, την επόμενη να έχει πρωινή.
- Κάθε υπάλληλος πρέπει σε οποιεσδήποτε 7 συνεχόμενες ημέρες του προγράμματος εργασίας (από την 1η έως την 7η, από την 2η έως την 8η, κ.ο.κ.) να έχει τουλάχιστον δύο ημέρες ανάπαυσης.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και επειδή, προφανώς, μπορεί να έχει πολλές λύσεις, προτείνετε μία εύλογη αντικειμενική συνάρτηση που να ποσοτικοποιεί την ποιότητα (ή το κόστος) μίας λύσης, οπότε πλέον έχουμε ένα πρόβλημα βελτιστοποίησης.

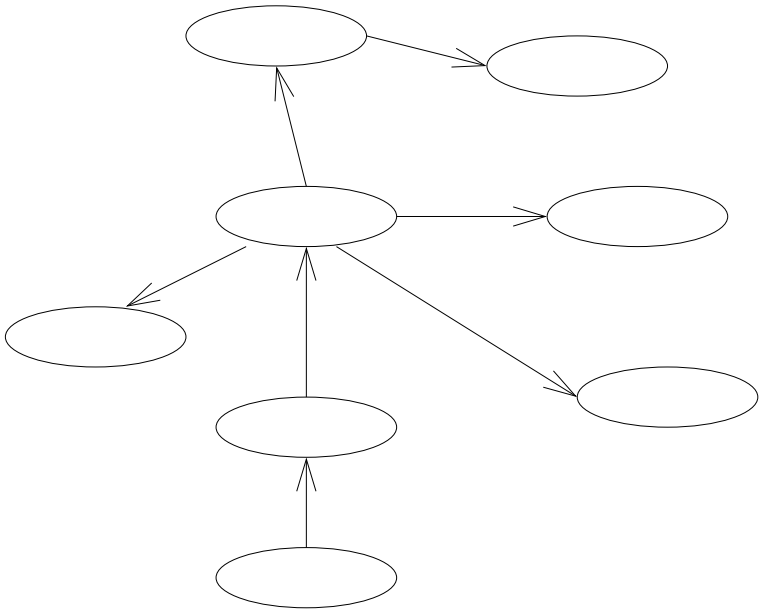
3. (α΄) i. Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
p(f(b)).
p(g(f(X))) :- p(f(X)).
p(f(g(X))) :- p(g(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(b)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(b)`. Επίσης, μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(a))`.

ii. Γνωρίζετε ότι το αποτέλεσμα της μεταγλώττισης ενός προγράμματος C είναι διαφορετικό μεταξύ μίας αρχιτεκτονικής Intel και μίας Sparc. Ισχύει το ίδιο και για το αποτέλεσμα της μεταγλώττισης ενός προγράμματος Prolog; Δικαιολογήστε την απάντησή σας.

(β΄) Δώστε ονόματα όπου νομίζετε ότι χρειάζεται ώστε η παρακάτω εικόνα να αναπαριστά μέρος μιας βάσης γνώσης. Ποια μεθοδολογία ακολουθεί η αναπαράσταση; Γράψτε ένα πρόγραμμα Prolog που να υλοποιεί τη γνώση που αναπαριστάνεται στο (συμπληρωμένο) σχήμα σας.



ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2014

1. (α΄) Δεδομένου ενός συνόλου γεγονότων Prolog που κωδικοποιούν σχέσεις γονέα-παιδιού μέσω του κατηγορήματος `parent/2`, θα μπορούσαμε να ορίσουμε τη σχέση "πρόγονος" με κάποιον από τους δύο εναλλακτικούς ορισμούς που φαίνονται στη συνέχεια για τα κατηγορήματα `pred1/2` και `pred2/2`.

```
pred1(X,Z) :- parent(X,Z).
pred1(X,Z) :- parent(X,Y), pred1(Y,Z).

pred2(X,Z) :- parent(X,Z).
pred2(X,Z) :- parent(Y,Z), pred2(X,Y).
```

Πότε είναι περισσότερο συμφέρων ο ορισμός `pred1/2` και πότε ο ορισμός `pred2/2`; Αιτιολογήστε την απάντησή σας, κάνοντας, όπου χρειάζεται, τις απαιτούμενες διερευνήσεις.

(β΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `drop(List1,N,List2)`, το οποίο για δεδομένη λίστα `List1` και θετικό ακέραιο `N`, να επιστρέφει στη μεταβλητή `List2` τη λίστα που προκύπτει αν διαγραφεί από τη `List1` κάθε `N`-οστό της στοιχείο. Για παράδειγμα, αν το `N` ισούται με 3, από τη λίστα `List1` θα πρέπει να διαγραφούν το 3ο, το 6ο, το 9ο, κλπ. στοιχεία της, για να προκύψει η `List2`. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- drop([a,b,c,d,e,f,g,h,i,j,k,l,m,n],4,L).
L = [a,b,c,e,f,g,i,j,k,m,n]
?- drop([a,b,c,d,e,f,g,h,i,j,k,l,m,n],3,L).
L = [a,b,d,e,g,h,j,k,m,n]
?- drop([1,2,3,4,5,6,7],1,L).
L = []
```

2. (α΄) Ζητήθηκε από κάποιον να ορίσει ένα κατηγόρημα `unifiable(List1,Term,List2)`, έτσι ώστε δεδομένης κάποιας λίστας `List1` και ενός όρου `Term`, να επιστρέφει στη μεταβλητή `List2` τη λίστα των στοιχείων της `List1` που είναι ενοποιήσιμα με τον όρο `Term`. Η απάντηση που έδωσε ήταν η εξής:

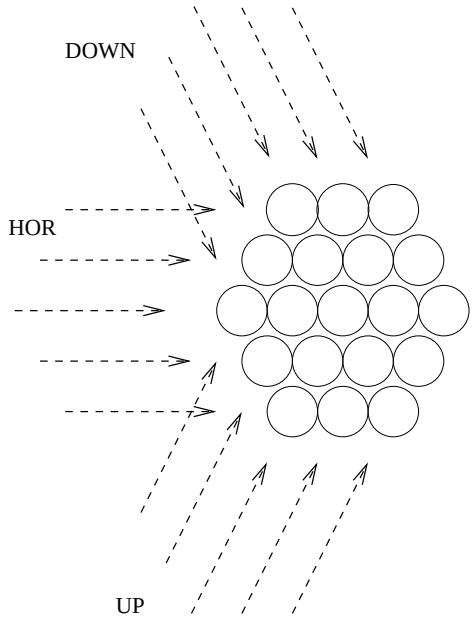
```
unifiable([], _, []).
unifiable([X|List1], Term, [X|List2]) :-
   X = Term,
   unifiable(List1, Term, List2).
unifiable([X|List1], Term, List2) :-
   X \= Term,
   unifiable(List1, Term, List2).
```

Σχολιάστε την απάντηση αυτή. Δίνει το σωστό αποτέλεσμα στην εξής ερώτηση;

```
?- unifiable([a,f(b),c,f(d),f(e)], f(X), List).
```

Αν όχι, τι αποτέλεσμα δίνει, ποιο θα έπρεπε να ήταν το σωστό αποτέλεσμα και ποια μικρή αλλαγή θα έπρεπε να κάνετε (της τάξης των 3-5 χαρακτήρων) στον ορισμό του `unifiable/3` ώστε να προκύπτει αυτό το αποτέλεσμα;

(β′) Έστω ότι θέλουμε να συμπληρώσουμε στους κύκλους του διπλανού σχήματος τους αριθμούς από το 1 έως το 19 έτσι ώστε στις πέντε γραμμές της οριζόντιας κατεύθυνσης (HOR), στις πέντε ανιούσες γραμμές (UP) και στις πέντε κατιούσες (DOWN) να έχουμε το ίδιο άθροισμα αριθμών (πόσο;). Γράψτε πρόγραμμα που να επιλύει το πρόβλημα αυτό σε κάποιο σύστημα λογικού προγραμματισμού με περιορισμούς της επιλογής σας, για παράδειγμα την ECL^{i}PS^e. Για το σκοπό αυτό, ορίστε ένα κατηγόρημα `hexagon(List)`, το οποίο να επιστρέφει στη μεταβλητή `List` τη λίστα των αριθμών που πρέπει να συμπληρωθούν στους κύκλους, από επάνω προς τα κάτω και από αριστερά προς τα δεξιά. Για κάθε λύση του προβλήματος, πόσες συμμετρικές με αυτή λύσεις υπάρχουν, οι οποίες προκύπτουν με περιστροφές ή κατοπτρικούς μετασχηματισμούς του εξαγώνου; Σχολιάστε ό,τι κρίνετε ενδιαφέρον σχετικά με τις συμμετρίες των λύσεων του προβλήματος, όπως, για παράδειγμα, πώς θα μπορούσε το πρόγραμμα που γράψατε να τροποποιηθεί ώστε να αποφεύγει την εύρεση συμμετρικών λύσεων, είτε όλων, είτε έστω κάποιων από αυτές; Τμήμα μίας ενδεικτικής εκτέλεσης είναι το εξής:




```
?- hexagon(List).
List = [3,19,16,17,7,2,12,18,1,5,4,10,11,6,8,13,9,14,15]   --> ;
.........................................................
```

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(f(f(a)))).
p(X) :- p(f(X)).
p(f(b)) :- p(a).
p(f(f(f(b)))) :- p(c).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(c)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(c)`.

(β′) Δώστε ένα παράδειγμα γνωστικού πεδίου όπου ενδείκνυται η χρήση σημασιολογικών δικτύων σαν μεθοδολογία αναπαράστασης γνώσης και δώστε μια μικρή βάση γνώσης από αυτό σε Prolog (15 προτάσεις), διαφορετικό από το παράδειγμα των σημειώσεων. Στη συνέχεια, επαυξήστε το με κανόνες if-then. Τι ρόλο μπορεί να παίξει η Prolog στην υλοποίηση;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2014

1. (α΄) Αν θέλουμε να αναπαραστήσουμε σε Prolog ένα συγκεκριμένο σημείο στο δισδιάστατο χώρο, μπορούμε να το κάνουμε χρησιμοποιώντας ένα συναρτησιακό σύμβολο βαθμού 2, έστω το `point/2`, για να κατασκευάσουμε έναν όρο με ορίσματα τις συντεταγμένες του σημείου. Για παράδειγμα, ο όρος Prolog `point(3,5)` ορίζει μονοσήμαντα το σημείο `(3,5)` του επιπέδου. Με παρόμοιο τρόπο, επειδή ένα τρίγωνο καθορίζεται πλήρως από τις τρεις κορυφές του, ο όρος `triangle(point(2,-1),point(0,7),point(3,10))` ορίζει μονοσήμαντα ένα τρίγωνο στο επίπεδο. Ένα κύκλος θα μπορούσε να καθορισθεί πλήρως από τον όρο `circle(point(7,1),radius(6))`, υπονοώντας ότι έχει κέντρο το σημείο `(7,1)` και ακτίνα ίση με 6.

   Στην προηγούμενη λογική, προτείνετε όρους Prolog, εξηγώντας τη σημασία των συστατικών τους, για τη μονοσήμαντη αναπαράσταση στο δισδιάστατο χώρο *i)* τετραπλεύρου, *ii)* παραλληλογράμμου, *iii)* ρόμβου, *iv)* ορθογωνίου παραλληλογράμμου και *v)* τετραγώνου.

   (β΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `compr(List1,List2)`, το οποίο για δεδομένη λίστα `List1`, να επιστρέφει στη μεταβλητή `List2` τη λίστα που προκύπτει αν αντικατασταθούν στη `List1` όλες οι **συνεχόμενες** εμφανίσεις κάθε στοιχείου από μία μόνο εμφάνιση του στοιχείου. Η σειρά των στοιχείων δεν πρέπει να μεταβληθεί. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- compr([a,a,a,b,c,c,d,d,d,d,e],List).
List = [a,b,c,d,e]
?- compr([f(1),g(2),g(2),g(1),f(1),f(1),g(2)],List).
List = [f(1),g(2),g(1),f(1),g(2)]
```

   Η υλοποίησή σας αρκεί να δουλεύει σωστά όταν τα στοιχεία της `List1` δεν περιέχουν μεταβλητές. Παρ΄ όλα αυτά, ποιες απαντήσεις θα παίρνατε αν στο πρόγραμμα που γράψατε υποβάλατε τις παρακάτω ερωτήσεις;

```
?- compr([f(1),f(X),f(2),g(X),g(1)],List).
.....................
?- compr([f(X),f(Y),g(X),g(1),g(2),g(Y)],List).
.....................
```

2. (α΄) Δίνεται το παρακάτω πρόγραμμα Prolog:

```
addroot(nil, X, t(nil, X, nil)).
addroot(t(L, Y, R), X, t(L1, X, t(L2, Y, R))) :-
   X < Y, addroot(L, X, t(L1, X, L2)).
addroot(t(L, Y, R), X, t(t(L, Y, R1), X, R2)) :-
   X > Y, addroot(R, X, t(R1, X, R2)).

addt(Tree, X, NewTree) :-
   addroot(Tree, X, NewTree).
addt(t(L, Y, R), X, t(L1, Y, R)) :-
   X < Y, addt(L, X, L1).
addt(t(L, Y, R), X, t(L, Y, R1)) :-
   X > Y, addt(R, X, R1).
```

Ποιες εναλλακτικές απαντήσεις θα δοθούν για τη μεταβλητή `D4`, αν υποβληθεί η παρακάτω ερώτηση; Προτιμήστε να απαντήσετε "σχηματικά", παρά αυστηρά συντακτικά.

```
?- addt(nil,3,D1),addt(D1,5,D2),addt(D2,1,D3),addt(D3,6,D4).
```

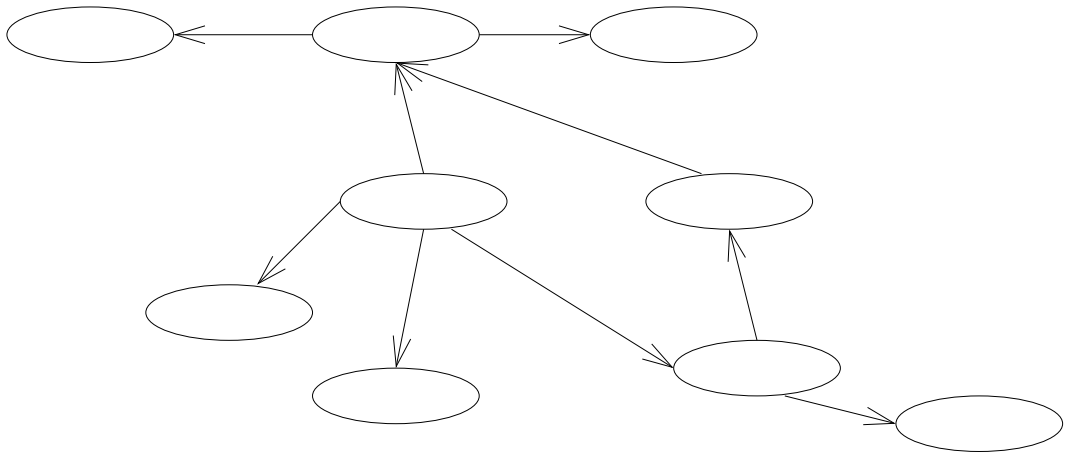
(β΄) Από τα απλά μαθηματικά που γνωρίζουμε, είναι απολύτως βέβαιο ότι είναι αδύνατο να βρούμε τρεις αριθμούς `X`, `Y` και `Z`, τέτοιους ώστε να ισχύουν `X > Y`, `Y > Z` και `Z > X`. Αν τα `X`, `Y` και `Z` μπορούσαν να πάρουν ακέραιες τιμές στο διάστημα `[0,9]`, ένας τρόπος να επιβεβαιώσουμε την προηγούμενη προφανή γνώση υπολογιστικά θα ήταν να γράψουμε ένα πρόγραμμα που θα γεννά συστηματικά όλους τους δυνατούς συνδυασμούς τιμών για τα `X`, `Y` και `Z` ($10^3 = 1000$ στο πλήθος) και θα ελέγχει αν για κάποιον απ΄ αυτούς ισχύουν οι τρεις ανισότητες. Φυσικά, δεν θα μπορούσε να βρεθεί κάποιος τέτοιος συνδυασμός. Αν όμως υποβάλλαμε σ΄ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, π.χ. στην ECL^{i}PS^e, την ερώτηση

```
?- [X,Y,Z] :: 0..9, X #> Y, Y #> Z, Z #> X.
```

όπου το κατηγόρημα `#>` εκφράζει τη σχέση του "μεγαλύτερου" σαν περιορισμό, τι απάντηση πιστεύετε ότι θα παίρναμε και πώς θα υπολογιζόταν αυτή η απάντηση, σε γενικές γραμμές, από το σύστημα; Μήπως με εξαντλητική αναζήτηση ή κάπως αλλιώς;

3. (α΄) Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Υπάρχει κάποια ενδιαφέρουσα επίπτωση όταν συμβαίνει κάτι τέτοιο; Δώστε ένα παράδειγμα οριστικού προγράμματος, το απλούστερο δυνατό που θα μπορούσατε να σκεφτείτε, το οποίο να περιλαμβάνει τουλάχιστον μία πρόταση και το οποίο να έχει σαν ελάχιστο μοντέλο το κενό σύνολο. Για το οριστικό πρόγραμμα που προτείνατε, δώστε και τουλάχιστον άλλα δύο μοντέλα, διάφορα του κενού συνόλου.

(β΄) Δώστε ονόματα όπου νομίζετε ότι χρειάζεται ώστε η παρακάτω εικόνα να αναπαριστά μέρος μιας βάσης γνώσης. Ποια μεθοδολογία ακολουθεί η αναπαράσταση; Γράψτε ένα πρόγραμμα Prolog που να υλοποιεί τη γνώση που αναπαριστάνεται στο (συμπληρωμένο) σχήμα σας.



ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2013

1. (α΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `ifthenelse(Cond,Goal1,Goal2)`, το οποίο, στην περίπτωση που ισχύει η συνθήκη `Cond`, να καλεί τον στόχο `Goal1`, αλλιώς τον στόχο `Goal2`. Θα πρέπει με τον τρόπο που ορίσατε το κατηγόρημα, να είναι δυνατόν να διατυπωθούν εμφωλευμένα if-then-else, δηλαδή οι στόχοι `Goal1` και `Goal2` να έχουν σαν κατηγόρημα το `ifthenelse/3`. Δώστε ένα παράδειγμα κλήσης αυτού του κατηγορήματος με εμφωλευμένα if-then-else και τον ισοδύναμο κώδικά του σε μία διαδικαστική γλώσσα προγραμματισμού, π.χ. την C. Πώς θα υλοποιούσατε και ένα κατηγόρημα `ifthen(Cond,Goal)`, με την προφανή σημασία, είτε μέσω του `ifthenelse/3`, είτε αυτοδύναμα;

   (β΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `maxcsl(List1,List2,Maxcsl)`, το οποίο για δεδομένες λίστες `List1` και `List2`, να επιστρέφει στη μεταβλητή `Maxcsl` τη μέγιστη (σε μήκος) κοινή υπολίστα (συνεχόμενων στοιχείων) στις `List1` και `List2`. Αν δεν υπάρχει κοινή υπολίστα, το κατηγόρημα να αποτυγχάνει, ενώ αν υπάρχουν περισσότερες από μία κοινές υπολίστες μεγίστου μήκους, να επιστρέφονται όλες μέσω οπισθοδρόμησης. Ο ορισμός που θα δώσετε για το ζητούμενο κατηγόρημα πρέπει να είναι ευανάγνωστος, αλλά η απάντησή σας δεν πρέπει να ξεπερνά τους 400 χαρακτήρες. Μπορείτε να θεωρήσετε ότι, στην περίπτωση που χρειάζεστε κάποια, τα συνήθη κατηγορήματα λιστών (`member`, `length`, `append`, και μόνο αυτά) είναι ενσωματωμένα. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- maxcsl([a,b,c,d,e,f,g],[h,f,g,i,b,c,d,j,d,e],Maxcsl).
Maxcsl = [b,c,d]
?- maxcsl([a,b,c,d,e,f,g],[h,f,g,i,b,c,d,j,d,e,f],Maxcsl).
Maxcsl = [b,c,d]  --> ;
Maxcsl = [d,e,f]
?- maxcsl([a,b,c,d],[e,f],Maxcsl).
no
?- maxcsl([b,c,b,c,b,c],[c,b,c,b,c,b], Maxcsl).
Maxcsl = [b,c,b,c,b]  --> ;
Maxcsl = [c,b,c,b,c]
```

2. (α΄) Υλοποιήστε σε Prolog ένα κατηγόρημα `diags(Matrix,DiagsDown,DiagsUp)`, το οποίο να παίρνει σαν όρισμα ένα πίνακα `Matrix`, στη μορφή μίας λίστας λιστών (οι εσωτερικές λίστες είναι οι γραμμές του πίνακα), και να επιστρέφει στα `DiagsDown` και `DiagsUp` τις λίστες των στοιχείων των κατιουσών και των ανιουσών διαγωνίων του πίνακα, αντίστοιχα. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- diags([[a,b,c,d],[e,f,g,h],[i,j,k,l]],DiagsDown,DiagsUp).
DiagsDown = [[i],[e,j],[a,f,k],[b,g,l],[c,h],[d]]
DiagsUp = [[a],[b,e],[c,f,i],[d,g,j],[h,k],[l]]
```

   (β΄) Μία διδιάστατη ασπρόμαυρη εικόνα έχει σκαναριστεί οριζόντια, κάθετα, και διαγώνια (κατά τις κατιούσες και τις ανιούσες διαγωνίους της) και έχει κωδικοποιηθεί με βάση τα πλήθη των μαύρων pixels σε κάθε γραμμή, κάθε στήλη, κάθε κατιούσα διαγώνιο και κάθε ανιούσα διαγώνιο. Ορίστε σε κάποιο περιβάλλον **λογικού προγραμματισμού με περιορισμούς** της επιλογής σας ένα κατηγόρημα `decode/4`, το οποίο να δέχεται σαν ορίσματα τα πλήθη των μαύρων pixels για όλες τις γραμμές, στήλες, κατιούσες και ανιούσες διαγωνίους

της εικόνας, να αποκωδικοποιεί την εικόνα και να την εκτυπώνει. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- decode([4,4,3,3,4,4],[2,2,2,4,2,4,2,2,2],
          [0,2,1,0,2,1,5,5,1,2,0,1,2,0],[0,2,1,0,2,1,5,5,1,2,0,1,2,0]).
 . * * . . . * * .
 * . . * . * . . *
 . . . * * * . . .
 . . . * * * . . .
 * . . * . * . . *
 . * * . . . * * .
?- decode([13,10,14,10,13],
   [5,2,2,1,0,5,2,3,2,0,3,2,2,3,0,5,1,1,1,0,3,2,2,3,0,3,2,3,2],
   [1,1,1,2,2,2,3,1,4,2,1,4,2,1,1,3,4,2,2,1,0,2,2,1,1,2,4,2,1,2,2,1,0],
   [1,2,2,2,3,1,2,2,2,3,1,2,3,1,1,3,3,1,1,1,1,3,3,1,1,2,4,2,1,1,2,2,0]).
 * * * . . * * * . . . * * . . * . . . . . * * . . . * * .
 * . . * . * . . * . * . . * . * . . . . * . . * . * . . .
 * * * . . * * * . . * . . * . * . . . . * . . * . * . * *
 * . . . . * . * . . * . . * . * . . . . * . . * . * . . *
 * . . . . * . . * . . * * . . * * * * . . * * . . . * * .
```

3. (α′) Δίνεται το εξής λογικό πρόγραμμα P:

```
p(f(f(f(a)))).
p(X) :- p(f(X)), p(f(f(X))).
```

   i. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος P;

   ii. Προτείνετε μία οριστική πρόταση C_1, έτσι ώστε το ελάχιστο μοντέλο του προγράμματος $P \cup \{C_1\}$ να περιέχει ακριβώς τέσσερα στοιχεία. Ποιο είναι αυτό;

   iii. Προτείνετε μία οριστική πρόταση C_2, έτσι ώστε το ελάχιστο μοντέλο του προγράμματος $P \cup \{C_2\}$ να περιέχει ακριβώς πέντε στοιχεία. Ποιο είναι αυτό;

   iv. Προτείνετε μία οριστική πρόταση C_3, έτσι ώστε το ελάχιστο μοντέλο του προγράμματος $P \cup \{C_3\}$ να είναι απειροσύνολο, αλλά να μην ταυτίζεται με τη βάση Herbrand. Ποιο είναι αυτό το ελάχιστο μοντέλο;

   v. Επιλέξτε μία μόνο από τις προηγούμενες περιπτώσεις και προτείνετε έναν ατομικό τύπο A που να μην ανήκει στο ελάχιστο μοντέλο. Ποια είναι η τομή όλων των μοντέλων στα οποία ανήκει ο A;

   (β′) Να υλοποιήσετε σε Prolog ένα κατηγόρημα `prove(Goal,NoOfSteps)` το οποίο να δέχεται σαν όρισμα ένα στόχο, `Goal`, και να προσπαθεί να τον ικανοποιήσει σε σχέση με μια βάση γνώσης που περιέχει if-then κανόνες και αξιώματα (γεγονότα), υλοποιημένα σε Prolog με τον εξής τρόπο. Αρχικά να εξετάζει τη βάση κατά τον εμπρόσθιο (forward) τρόπο για (το πολύ) `NoOfSteps` βήματα. Αν κατά τη διαδικασία αυτή ο στόχος δεν ικανοποιηθεί, να εξετάζει τη βάση κατά τον οπίσθιο (backward) τρόπο, λαμβάνοντας υπόψη και τα συμπεράσματα που προέκυψαν από την προηγούμενη διαδικασία.

   Να αναδείξετε την υλοποίησή σας για ένα συγκεκριμένο παράδειγμα βάσης γνώσης, δίδοντας τα βήματα της εκτέλεσης και τα αποτελέσματά τους για ένα συγκεκριμένο στόχο.

   Σημείωση: Για λόγους ευκολίας, θεωρήστε ότι συζεύξεις, διαζεύξεις και αρνήσεις δεν υποστηρίζονται.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β′ Περιόδου 2013

1. (α′) Πολλές φορές, στην καθημερινή μας ζωή λέμε "*δύο αρνήσεις ισούνται με μία κατάφαση*". Ισχύει το ίδιο και στην Prolog; Δηλαδή είναι απολύτως ισοδύναμα το να καλέσουμε τον στόχο *Goal* ή τον `not not` *Goal*; Εξηγήστε την άποψή σας, μέσω των απαντήσεων που θα πάρετε στην Prolog, αν υποβάλετε τις παρακάτω ερωτήσεις:

```
?- not not 3 = 5.          ?- 3 = 5.
?- not not 5 = 5.          ?- 5 = 5.
?- not not X = 5.          ?- X = 5.
```

(β′) Μελετήστε προσεκτικά τον ορισμό του κατηγορήματος `binrep/3` στο σχήμα δεξιά. Δεδομένου αυτού του ορισμού, ποια θα είναι η απάντηση ενός συστήματος Prolog στην παρακάτω ερώτηση;

```
?- binrep(4, [a,b,c], List).
```

```
binrep(0, List, List).
binrep(N, List, RepList) :-
   N > 0,
   append(List, List, NewList),
   N1 is N-1,
   binrep(N1, NewList, RepList).
```

Περιγράψτε, με μία πρόταση, τη λειτουργικότητα αυτού του ορισμού.
Τροποποιήστε κατάλληλα τον ορισμό του `binrep/3`, έτσι ώστε να υλοποιεί, εκτός από τη λειτουργικότητα που αναφέρατε, και την εξής:

```
?- binrep(N, List, [1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2]).
N = 0
List = [1,2,1,2,1,2,1,2,1,2,1,2,1,2,1,2]   --> ;
N = 1
List = [1,2,1,2,1,2,1,2]                   --> ;
N = 2
List = [1,2,1,2]                           --> ;
N = 3
List = [1,2]
```

2. (α′) Μελετήστε προσεκτικά τον ορισμό του κατηγορήματος `solve3x3/2` παρακάτω. Με βάση αυτόν τον ορισμό, ποια θα είναι η απάντηση ενός συστήματος Prolog στην εξής ερώτηση;

```
?- solve3x3([147,347,376,576,815,833], Sol).
```

```
solve3x3(Nums, Sol) :- Sol = [X1,X2,X3,X4,X5,X6,X7,X8,X9],
   HVSol = [X1,X4,X7,X2,X5,X8,X3,X6,X9|Sol],
   generate(Sol), checknums(Nums, HVSol).

generate([]).
generate([Dig|Sol]) :-
   member(Dig, [0,1,2,3,4,5,6,7,8,9]), generate(Sol).

checknums([], []).
checknums(Nums, [Y1,Y2,Y3|HVSol]) :- delete(Num, Nums, RestNums),
   Num =:= 100*Y1+10*Y2+Y3, checknums(RestNums, HVSol).
```

Ποια είναι η λειτουργικότητα αυτού του ορισμού; Είναι, κατά τη γνώμη σας, υλοποιημένος με αποδοτικό τρόπο; Αν όχι, δώστε μία εναλλακτική υλοποίηση του κατηγορήματος `solve3x3/2`, που να είναι εξαιρετικά αποδοτική.

(β΄) Διατυπώστε τη γενίκευση του προβλήματος που λύνει το κατηγόρημα `solve3x3/2` του προηγούμενου ερωτήματος, ενδεχομένως ορίζοντας τις προδιαγραφές για ένα κατηγόρημα `solveNxN/2`, περιγράψτε πώς θα το μοντελοποιούσατε σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και σκιαγραφήστε πώς θα το αντιμετωπίζατε σ΄ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, όπως η ECL^{i}PS^e.

3. (α΄) Δίνονται οι εξής οριστικές προτάσεις:
C_1: `p(f(g(a))).`
C_2: `p(X) :- p(f(X)).`
C_3: `p(f(X)) :- p(g(X)).`
C_4: `p(g(f(X))) :- p(f(g(X))).`
Έστω $C = \{C_1, C_2, C_3, C_4\}$.

i. Επιλέξτε ένα $P \subseteq C$, τέτοιο ώστε το ελάχιστο μοντέλο M του P να περιλαμβάνει $N = 2$ στοιχεία. Ποιο είναι το M;

ii. Το ίδιο με το προηγούμενο, αλλά για $N = 3$.

iii. Το ίδιο με το προηγούμενο, αλλά για $N = 4$.

iv. Το ίδιο με το προηγούμενο, αλλά για $N = 5$.

v. Το ίδιο με το προηγούμενο, αλλά για $N = 7$.

vi. Το ίδιο με το προηγούμενο, αλλά για $N = 1$.

vii. Το ίδιο με το προηγούμενο, αλλά για $N = 0$.

viii. Το ίδιο με το προηγούμενο, αλλά για $N = \infty$.

Σε κάποια από τα προηγούμενα ερωτήματα, ενδέχεται να υπάρχουν περισσότερες της μίας σωστές απαντήσεις. Αρκεί να δώσετε κάποια από αυτές. Επίσης, είναι ενδεχόμενο σε κάποια ερωτήματα να μην υπάρχει απάντηση. Αρκεί να το επισημάνετε.

(β΄) Σε μια αίθουσα εξέτασης έχουμε εξεταζόμενους και επιτηρητές. Τόσο οι εξεταζόμενοι, όσο και οι επιτηρητές, είναι άνθρωποι. Οι άνθρωποι, στη γενική περίπτωση, είναι καθιστοί (αναφερόμαστε στην εξέταση). Κάθε εξεταζόμενος (μπορεί να) κάθεται ακριβώς πίσω από κάποιον άλλο εξεταζόμενο, ακριβώς μπροστά, ακριβώς αριστερά και ακριβώς δεξιά από κάποιους άλλους. Ένας εξεταζόμενος Α βρίσκεται πίσω (μπροστά/αριστερά/δεξιά) από κάποιον άλλο Γ, αν κάθεται ακριβώς πίσω (μπροστά/αριστερά/δεξιά) από κάποιον Β και ο Β βρίσκεται πίσω (μπροστά/αριστερά/δεξιά) από τον Γ. Ένας επιτηρητής (μπορεί να) είναι όρθιος. Ένας εξεταζόμενος καταλαμβάνει μια θέση σε μια συγκεκριμένη γραμμή και στήλη. Ένας επιτηρητής επιτηρεί ένα χώρο που αποτελείται από ένα εύρος γραμμών και στηλών.

Ποια μεθοδολογία αναπαράστασης γνώσης ενδείκνυται να χρησιμοποιηθεί για να αναπαραστήσουμε τον παραπάνω κόσμο; Να γράψετε ένα πρόγραμμα Prolog που

i. να περιγράφει ένα συγκεκριμένο στιγμιότυπο αίθουσας εξέτασης ακολουθώντας τη μεθοδολογία που έχετε προτείνει

ii. να μπορεί να απαντά στις ερωτήσεις
- Ο Χ κάθεται
- Ο Υ είναι όρθιος
- Ο Χ βρίσκεται αριστερά από τον Υ
- Ο Χ επιτηρεί τον Υ

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2012

1. (α΄) i. Γνωρίζετε ότι η προσεταιριστικότητα των ενδοσημασμένων τελεστών στην Prolog καθορίζεται από τον τύπο τους. Τι σημαίνει ο ορισμός του τελεστή `+` ως `yfx`; Τι διαφορά θα είχε αν οριζόταν σαν `xfy` ή `xfx`; Θα μπορούσε να ορισθεί και σαν `yfy`; Διατυπώστε τις απαντήσεις σας στα προηγούμενα σε σχέση με τη σημασία του όρου `2+3+7` στην Prolog. Εν πάση περιπτώσει, γιατί ασχολούμαστε με την προσεταιριστικότητα του `+`; Δεν είναι για την Prolog τα `(2+3)+7` και `2+(3+7)` ίδια πράγματα;

   ii. Η προτεραιότητα του `+` ορίζεται ίση με 500 και του `*` ίση με 400. Δεν είναι περίεργο να έχει η πρόσθεση μεγαλύτερη προτεραιότητα από τον πολλαπλασιασμό; Δηλαδή, ισχύει το αντίστροφο απ΄ ότι στην άλγεβρα;

   (β΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `maxrep/2` έτσι ώστε όταν αυτό καλείται σαν `maxrep(List, Maxrep)`, να βρίσκει τη μέγιστη υπολίστα ίδιων συνεχόμενων στοιχείων στη λίστα `List` και να επιστρέφει το αποτέλεσμα στο `Maxrep` σαν έναν όρο της μορφής `X/N`, που σημαίνει ότι το στοιχείο `X` επαναλαμβάνεται `N` φορές. Αν υπάρχουν περισσότερες της μίας υπολίστες μεγίστου μήκους με επαναλήψεις ίδιου στοιχείου, να επιστρέφονται όλες οι δυνατές λύσεις μέσω οπισθοδρόμησης. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- maxrep([a,a,b,c,c,c,c,d,d,d,e,e,e,e], Maxrep).
Maxrep = c/4     --> ;
Maxrep = e/4
?- maxrep([a,b,c], Maxrep).
Maxrep = a/1     --> ;
Maxrep = b/1     --> ;
Maxrep = c/1
?- maxrep([x,x,x,x,x,x,x,x,x,x,x,x,x,x], Maxrep).
Maxrep = x/14
```

2. (α΄) Σε αθλητικές διοργανώσεις, όπως το Ευρωπαϊκό Πρωτάθλημα Ποδοσφαίρου, συνήθως οι ομάδες χωρίζονται αρχικά σε ομίλους, με σκοπό να δώσουν αγώνες μεταξύ τους με όλους τους δυνατούς τρόπους. Από τους αγώνες αυτούς, οι ομάδες παίρνουν βαθμούς, 3 για κάθε νίκη, 1 για κάθε ισοπαλία και 0 για τις ήττες, όπου με βάση την τελική κατάταξή τους, καθορίζεται αν θα προχωρήσουν ή όχι στις επόμενες φάσεις της διοργάνωσης. Ορίστε σε Prolog[2] ένα κατηγόρημα `guessresults/4`, το οποίο να καλείται σαν `guessresults(Teams, Matches, Points, Results)`, όπου `Teams` είναι μία λίστα με τις ομάδες ενός ομίλου, `Matches` είναι οι αγώνες που έχουν δώσει οι ομάδες μέχρι στιγμής (όχι κατ΄ ανάγκη όλοι οι δυνατοί), σαν μία λίστα από όρους της μορφής `Team1-Team2`, και `Points` είναι η λίστα των συνολικών βαθμών που έχουν συγκεντρώσει οι ομάδες της λίστας `Teams`, με 1-1 αντιστοιχία, από τους αγώνες της λίστας `Matches`. Το κατηγόρημα να επιστρέφει στο `Results` έναν συμβατό συνδυασμό αποτελεσμάτων για τους αγώνες `Matches`, σαν μία λίστα από `1` (νίκη της πρώτης ομάδας), `0` (ισοπαλία) και `-1` (νίκη της δεύτερης ομάδας).[3] Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- guessresults([pol,gre,rus,cze], [pol-gre,rus-cze,gre-cze,pol-rus,
                  gre-rus,cze-pol], [2,4,4,6], Results).
```

[2] Μπορείτε στην απάντησή σας να χρησιμοποιήσετε είτε απλή Prolog, είτε προγραμματισμό με περιορισμούς.

[3] Οι ομάδες ενός ομίλου μπορεί να είναι οσεσδήποτε, όχι πάντοτε τέσσερις, όπως συμβαίνει στην πραγματικότητα.

```
Results = [0,-1,1,0,-1,1]     --> ;
Results = [0,1,-1,0,1,1]
?- guessresults([ned,den,ger,por], [ned-den,den-por,por-ned], [0,3,0,6],
                Results).
Results = [-1,-1,1]
?- guessresults([fra,eng,ukr,swe], [fra-eng,ukr-swe,ukr-fra,swe-eng],
                [4,4,3,0], Results).
Results = [0,1,-1,-1]
```

(β′) Θεωρήστε ότι δίνονται N πόλεις και οι αποστάσεις μεταξύ τους ανά δύο. Στο πρόβλημα του πλανόδιου πωλητή (traveling salesman problem) το ζητούμενο είναι να βρεθεί η βέλτιστη σειρά επίσκεψης του πωλητή σε όλες τις πόλεις, αρχίζοντας τη διαδρομή του από κάποια πόλη και καταλήγοντας στην ίδια, έτσι ώστε η συνολική απόσταση που θα διανύσει να είναι ελάχιστη. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί, συνάρτηση κόστους) και σκιαγραφήστε πώς θα το αντιμετωπίζατε σ′ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, όπως η ECL^{i}PS^e. Αν $N = 5$, η σειρά επίσκεψης 1, 5, 3, 2, 4 διαφέρει από την 2, 4, 1, 5, 3 ή από την 1, 4, 2, 3, 5; Με βάση την απάντησή σας στην προηγούμενη ερώτηση, μπορείτε να επαυξήσετε τη μοντελοποίηση του προβλήματος με κάποιο τρόπο, ώστε ο χώρος αναζήτησης, άρα και ο χρόνος επίλυσης, να είναι μικρότερος;

3. (α′) i. Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
p(f(b)).
p(g(f(X))) :- p(f(X)).
p(f(g(X))) :- p(g(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(b)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(b)`. Επίσης, μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(a))`.

ii. Γνωρίζετε ότι το αποτέλεσμα της μεταγλώττισης ενός προγράμματος C είναι διαφορετικό μεταξύ μίας αρχιτεκτονικής Intel και μίας Sparc. Ισχύει το ίδιο και για το αποτέλεσμα της μεταγλώττισης ενός προγράμματος Prolog; Δικαιολογήστε την απάντησή σας.

(β′) Ο κανόνας συμπερασμού της απαγωγής (abduction) επιτρέπει το εξής: Στην περίπτωση που έχουμε μια συνεπαγωγή της μορφής $p \rightarrow q$, όπου το p είναι ένας ατομικός τύπος που ανήκει σε κάποιο χαρακτηριστικό σύνολο (abducibles), τότε εάν το q είναι αληθές, μας επιτρέπει να θεωρήσουμε ότι και το p είναι αληθές. Συνοπτικά: Εάν $p \rightarrow q$ και $p \in$ abducibles και q αληθές, τότε p αληθές.

Να υλοποιήσετε σε Prolog μία μηχανή συμπερασμού για αναπαράσταση γνώσης μέσω if_then κανόνων, η οποία να υποστηρίζει τόσο οπίσθια συλλογιστική, αλλά και απαγωγή, προκειμένου να αποδείξει την αλήθεια ενός ατομικού τύπου. Για λόγους ευκολίας, θεωρήστε ότι το αριστερό μέλος των κανόνων αποτελείται από το πολύ έναν ατομικό τύπο.

Θεωρώντας μια βάση γνώσης που εσείς θα επιλέξετε, να αναδείξετε τη λειτουργία του προγράμματός σας, μέσω της εκτέλεσης συγκεκριμένου παραδείγματος το οποίο απαιτεί τη χρήση και των δύο μηχανισμών.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β′ Περιόδου 2012

1. (α′) i. Γνωρίζετε ότι στην Prolog μπορούμε να χρησιμοποιήσουμε στο σώμα ενός κανόνα τον τελεστή της διάζευξης ; (ελληνικό ερωτηματικό). Δώστε ένα παράδειγμα κανόνα στον οποίο να χρησιμοποιείται η διάζευξη. Επαναδιατυπώστε τον κανόνα αυτό χωρίς χρήση διάζευξης. Πώς πιστεύετε ότι ενδέχεται να είναι υλοποιημένη η διάζευξη στην Prolog; Είναι, κατά τη γνώμη σας, η διάζευξη κάποια δυνατότητα που ήταν απολύτως απαραίτητη για τη γλώσσα; Πιστεύετε ότι είναι καλό να την χρησιμοποιούμε όσο περισσότερο γίνεται;

   ii. Όταν σε μία ερώτηση Prolog η απάντηση είναι `yes`, ποια από τις παρακάτω είναι η σωστή ερμηνεία της απάντησης;

   Α′. *Με βάση τη γνώση που έχω, αυτό που ρωτάς ίσως είναι αληθές.*

   Β′. *Με βάση τη γνώση που έχω, αυτό που ρωτάς σίγουρα είναι αληθές.*

   Γ′. *Ανεξάρτητα από τη γνώση που έχω, αυτό που ρωτάς σίγουρα είναι αληθές.*

   Σε περίπτωση αρνητικής απάντησης `no`, ποια είναι η σωστή ερμηνεία της;

   Α′. *Με βάση τη γνώση που έχω, αυτό που ρωτάς ίσως είναι ψευδές.*

   Β′. *Με βάση τη γνώση που έχω, αυτό που ρωτάς σίγουρα είναι ψευδές.*

   Γ′. *Με βάση τη γνώση που έχω, δεν μπορώ να γνωρίζω αν αυτό που ρωτάς σίγουρα είναι αληθές.*

   Δικαιολογήστε τις απαντήσεις σας.

   (β′) Να υλοποιηθεί σε Prolog το κατηγόρημα `strangey/1` έτσι ώστε όταν αυτό καλείται σαν `strangey(L)`, να επιστρέφει στο `L` μία λίστα 10 στοιχείων, με την ιδιότητα να προκύπτει η ίδια λίστα είτε όταν προστεθούν στην αρχή της `L` τα στοιχεία `a`, `b`, `c` είτε όταν προστεθούν στο τέλος της `L` τα στοιχεία `b`, `c`, `a`, με αυτήν ακριβώς τη σειρά.

   Επίσης, να υλοποιηθεί και το κατηγόρημα `strangen/1` έτσι ώστε όταν αυτό καλείται σαν `strangen(L)`, να επιστρέφει στο `L` μία λίστα 10 στοιχείων, με την ιδιότητα να προκύπτει η ίδια λίστα είτε όταν προστεθούν στην αρχή της `L` τα στοιχεία `a`, `b`, `c` είτε όταν προστεθούν στο τέλος της `L` τα στοιχεία `b`, `a`, `c`, με αυτήν ακριβώς τη σειρά. Δηλαδή:

```
?- strangey(L).
.......................
?- strangen(L).
.......................
```

   Τι απαντήσεις περιμένετε να πάρετε στα αποσιωπητικά παραπάνω;

2. (α′) Δυαδικό λεξικό είναι ένα δυαδικό δέντρο του οποίου κάθε κόμβος είναι "μεγαλύτερος" από όλους τους κόμβους του αριστερού υποδέντρου του και "μικρότερος" από όλους τους κόμβους του δεξιού υποδέντρου του. Οι όροι "μεγαλύτερος" και "μικρότερος" αναφέρονται σε μία δεδομένη σχέση διάταξης. Υιοθετώντας μία κατάλληλη αναπαράσταση δυαδικών δέντρων (άρα και δυαδικών λεξικών), ορίστε σε Prolog ένα κατηγόρημα `dictionary/1` το οποίο να επιτυγχάνει όταν το όρισμα που του δίνεται είναι δυαδικό λεξικό. Φροντίστε η υλοποίηση του κατηγορήματος `dictionary/1` να είναι τέτοια ώστε η χρονική πολυπλοκότητά του να είναι γραμμική ως προς το πλήθος των κόμβων του δυαδικού δέντρου που του δίνεται σαν όρισμα.

(β′) Το πρόβλημα της διαμέρισης αριθμών (number partitioning) είναι από τα σημαντικότερα στη θεωρητική πληροφορική. Σε μία εκδοχή του προβλήματος, το ζητούμενο είναι να διαμερίσουμε το σύνολο των ακεραίων αριθμών $\{1, 2, 3, \ldots, N\}$, για δεδομένο N, σε δύο σύνολα A και B, τέτοια ώστε:

i. Τα σύνολα A και B να έχουν το ίδιο πλήθος στοιχείων.[4]

ii. Το άθροισμα των στοιχείων του A να ισούται με το άθροισμα των στοιχείων του B.

iii. Το άθροισμα των τετραγώνων των στοιχείων του A να ισούται με το άθροισμα των τετραγώνων των στοιχείων του B.

Χρησιμοποιώντας την τεχνική του προγραμματισμού με περιορισμούς, ορίστε σε ECL^{i}PS^e ένα κατηγόρημα `numpart/3`, το οποίο όταν καλείται σαν `numpart(N, A, B)`, για δεδομένο `N`, να επιστρέφει στα `A` και `B` δύο λίστες που προκύπτουν από τη διαμέριση του συνόλου $\{1, 2, 3, \ldots, N\}$, σύμφωνα με τα παραπάνω. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- numpart(8, A, B).
A = [1, 4, 6, 7]
B = [2, 3, 5, 8]
?- numpart(20, A, B).
A = [1, 2, 4, 10, 12, 13, 14, 15, 16, 18]
B = [3, 5, 6, 7, 8, 9, 11, 17, 19, 20]      --> ;
A = [1, 2, 5, 9, 11, 13, 14, 15, 17, 18]
B = [3, 4, 6, 7, 8, 10, 12, 16, 19, 20]     --> ;
.......................................
```

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
p(f(b)).
p(g(f(X))) :- p(f(X)).
p(f(g(X))) :- p(g(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(b)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(b)`. Επίσης, μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(a))`.

(β′) Ο κανόνας συμπερασμού της απαγωγής (abduction) επιτρέπει το εξής: Στην περίπτωση που έχουμε μια συνεπαγωγή της μορφής $p \to q$, όπου το p είναι ένας ατομικός τύπος που ανήκει σε κάποιο χαρακτηριστικό σύνολο (abducibles), τότε εάν το q είναι αληθές, μας επιτρέπει να θεωρήσουμε ότι και το p είναι αληθές. Συνοπτικά: Εάν $p \to q$ και $p \in$ abducibles και q αληθές, τότε p αληθές.

Να υλοποιήσετε σε Prolog μία μηχανή συμπερασμού για αναπαράσταση γνώσης μέσω if_then κανόνων, η οποία να υποστηρίζει τόσο οπίσθια συλλογιστική, αλλά και απαγωγή, προκειμένου να αποδείξει την αλήθεια ενός ατομικού τύπου. Για λόγους ευκολίας, θεωρήστε ότι το αριστερό μέλος των κανόνων αποτελείται από το πολύ έναν ατομικό τύπο.

Θεωρώντας μια βάση γνώσης που εσείς θα επιλέξετε, να αναδείξετε τη λειτουργία του προγράμματός σας, μέσω της εκτέλεσης συγκεκριμένου παραδείγματος το οποίο απαιτεί τη χρήση και των δύο μηχανισμών.

[4] Προφανώς, το πρόβλημα αποκλείεται να έχει λύση αν το N είναι περιττός.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2011

1. (α΄) Ισχυριστήκαμε στο μάθημα ότι δύο τρόποι για να υλοποιήσουμε στην Prolog μία επαναληπτική διαδικασία είναι η *αναδρομή* και η *οπισθοδρόμηση*. Διατυπώστε ένα πολύ απλό πρόβλημα, το οποίο θα αντιμετωπίζαμε σε μία κλασική διαδικαστική γλώσσα προγραμματισμού μέσω μίας δομής επανάληψης, και δείξτε πώς θα υλοποιούσαμε τη λύση του στην Prolog μέσω των δύο προηγούμενων τεχνικών.

   (β΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `subseqs/3` έτσι ώστε όταν αυτό καλείται σαν `subseqs(N, List, Subseqs)`, να επιστρέφει στο `Subseqs` τη λίστα που περιλαμβάνει όλες τις υπολίστες συνεχόμενων `N` στοιχείων της λίστας `List`. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- subseqs(3, [a, b, c, d, e, f, g], Subseqs).
Subseqs = [[a, b, c], [b, c, d], [c, d, e], [d, e, f], [e, f, g]]
```

   Πώς (θα πρέπει να) συμπεριφέρεται το κατηγόρημα που ορίσατε όταν κληθεί με τιμή του `N` μεγαλύτερη από το πλήθος των στοιχείων της λίστας `List`; Αν κληθεί με `N` ίσο με 0;

2. (α΄) Υπάρχει μία παρέα από φίλους, κάποιοι από τους οποίους λένε πάντα ψέμματα, ενώ οι υπόλοιποι λένε πάντα την αλήθεια. Όλοι τους γνωρίζουν για τον καθένα άλλο, αν είναι ψεύτης ή όχι. Κάποια ημέρα, για να περάσουν την ώρα τους, παίζουν το εξής παιγνίδι. Το κάθε άτομο κάνει μία δήλωση της μορφής "*στην παρέα υπάρχουν τουλάχιστον Κ ψεύτες*" Για παράδειγμα, έστω ότι υπάρχουν 5 άτομα και οι δηλώσεις τους είναι οι εξής:

   i. Ο 1ος δηλώνει: "Υπάρχουν τουλάχιστον 3 ψεύτες μεταξύ μας."
   ii. Ο 2ος δηλώνει: "Υπάρχουν τουλάχιστον 2 ψεύτες μεταξύ μας."
   iii. Ο 3ος δηλώνει: "Υπάρχει τουλάχιστον 1 ψεύτης μεταξύ μας."
   iv. Ο 4ος δηλώνει: "Υπάρχουν τουλάχιστον 4 ψεύτες μεταξύ μας."
   v. Ο 5ος δηλώνει: "Υπάρχουν τουλάχιστον 2 ψεύτες μεταξύ μας."

   Σχετικά εύκολα μπορούμε να διαπιστώσουμε ότι με βάση τα παραπάνω δεδομένα, στην παρέα πρέπει το 1ο και το 4ο άτομο να είναι ψεύτες και μόνο αυτοί.

   Υλοποιήστε σε Prolog ένα κατηγόρημα `liars/2`, το οποίο όταν καλείται με πρώτο όρισμα τη λίστα των αριθμών που δηλώνονται από κάθε άτομο ως ελάχιστο πλήθος ψευτών στην παρέα, να επιστρέφει στο δεύτερο όρισμα μία λίστα που να δείχνει τι είναι το κάθε άτομο της παρέας, ψεύτης ή όχι, μέσω κατάλληλης τιμής, 1 ή 0. Δύο παραδείγματα εκτέλεσης είναι τα εξής:

```
?- liars([3, 2, 1, 4, 2], Liars).
Liars = [1, 0, 0, 1, 0]

?- liars([4, 9, 1, 12, 14, 8, 1, 17, 3, 6, 5, 6, 18,
          20, 0, 8, 7, 9, 4, 16], Liars).
Liars = [0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1]
```

   Μπορείτε στην υλοποίησή σας να χρησιμοποιήσετε είτε απλή Prolog, είτε προγραμματισμό με περιορισμούς.

(β′) Κάποιος πηγαίνει σε ένα κατάστημα ψιλικών για να αγοράσει τέσσερα συγκεκριμένα προϊόντα. Αφού ζητήσει τι θέλει ακριβώς και του τα δώσει ο καταστηματάρχης, ο τελευταίος κάνει τον λογαριασμό με τη βοήθεια ενός επιτραπέζιου calculator (όχι, δεν έχει ταμειακή μηχανή...) και του λέει "7.11 ευρώ, παρακαλώ". Πληρώνει ο πελάτης το ποσό και τη στιγμή που βρίσκεται στην έξοδο του φωνάζει ο καταστηματάρχης: "Συγνώμη, κύριε! Αντί να κάνω πρόσθεση, έκανα κατά λάθος πολλαπλασιασμό! Ελάτε πάλι, σας παρακαλώ". Αφού κάνει τώρα τη σωστή πράξη, διαπιστώνει με έκπληξη ότι το αποτέλεσμα είναι το ίδιο, 7.11 ευρώ. Ποιο μπορεί να είναι το κόστος του κάθε προϊόντος που αγόρασε ο πελάτης;[5]

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και γράψτε ένα πρόγραμμα σε ECL^{i}PS^e (είτε μέσω της βιβλιοθήκης `fd`, είτε της `ic`) που να βρίσκει το κόστος του κάθε προϊόντος. Σημειώστε ότι, παρότι το πρόβλημα προφανώς εμπλέκει αριθμούς που δεν είναι ακέραιοι, εσείς θα πρέπει να το αντιμετωπίσετε μέσω ακεραίων και μόνο.

3. (α′) i. Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(a),a).
p(X,f(Y)) :- p(X,Y).
p(f(X),X) :- p(X,X).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(a,a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(a,a)`. Επίσης, μπορεί το `p(f(f(a)),a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(f(a)),a)`.

ii. Πιστεύετε ότι η βάση Herbrand είναι μοντέλο για κάθε οριστικό πρόγραμμα; Αν ναι, εξηγήστε την άποψή σας. Αν όχι, δώστε ένα παράδειγμα οριστικού προγράμματος για το οποίο η βάση Herbrand να μην είναι μοντέλο του. Απαντήστε στην ίδια ερώτηση και για την περίπτωση μη οριστικών προγραμμάτων.

(β′) Υλοποιήστε σε Prolog μια μηχανή συμπερασμού, σε μέτα-επίπεδο, που να δέχεται μια λίστα από αληθή γεγονότα, π.χ. της μορφής `[running_nose, fever, shivering]`, και με δεδομένη μια βάση γνώσης if-then κανόνων και γεγονότων, να καταλήγει σε κάποιο συμπέρασμα. Δηλαδή, υλοποιήστε το `diagnosis(ListofFacts, Diagnosis)`, όπου δίνεται το πρώτο όρισμα και επιστρέφεται το δεύτερο.

Για να το πετύχει αυτό η μηχανή συμπερασμού, να συνδυάζει εμπρόσθια και οπίσθια συλλογιστική, λειτουργώντας ως εξής:

i. Αρχικά, κατά τον εμπρόσθιο τρόπο, να επιλέγει κανόνα για τον οποίο μέρος της συνθήκης του να ικανοποιείται από κάποι-ο/-α από τα γεγονότα που δίνονται στο πρώτο όρισμα του `diagnosis/2`.

ii. Στη συνέχεια, να προσπαθεί να επιβεβαιώσει την υπόλοιπη συνθήκη του κανόνα, κατά τον οπίσθιο τρόπο.

iii. Στην περίπτωση που κάτι δεν επιβεβαιώνεται, με οπίσθιο τρόπο, από τη βάση γνώσης, να ρωτά αν ισχύει και, σε περίπτωση καταφατικής απάντησης του χρήστη,[6] να προστίθεται σε αυτή, αλλιώς να αποτυγχάνει.

Οι συνθήκες των κανόνων μπορεί να περιλαμβάνουν συζεύξεις ή/και διαζεύξεις.

[5] Για την ιστορία, τα τέσσερα προϊόντα κοστίζουν 1.20, 1.25, 1.50 και 3.16 ευρώ, αντίστοιχα.

[6] Η ανάγνωση της εισόδου μπορεί να γίνεται μέσω του ενσωματωμένου κατηγορήματος `read/1`.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2011

1. (α΄) Αν με τον όρο "*καθαρή Prolog*" (pure Prolog) εννοούμε το τμήμα εκείνο της γλώσσας που δεν περιλαμβάνει ενσωματωμένα κατηγορήματα, ποια από τα ενσωματωμένα κατηγορήματα `=/2`, `\=/2`, `!/0`, `not/1`, `true/0`, `findall/3`, `var/1`, `write/1`, `length/2` και `member/2` **δεν** μπορούν να υλοποιηθούν σε καθαρή Prolog; Για τα υπόλοιπα, δώστε μία υλοποίησή τους σε καθαρή Prolog.

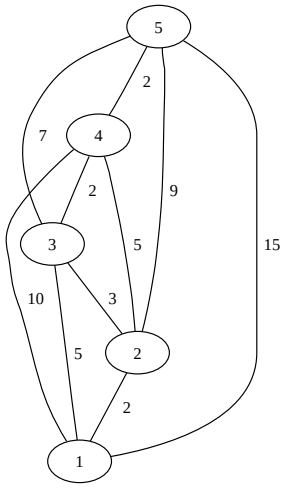
   (β΄) Στην Prolog μπορούμε να αναπαραστήσουμε σύνολα με τη βοήθεια λιστών, στις οποίες αδιαφορούμε για τη σειρά των στοιχείων τους και δεν επιτρέπουμε την ύπαρξη επαναλαμβανόμενων στοιχείων. Με βάση αυτό, δώστε σε Prolog μία υλοποίηση των

```
set_intersection(Set1, Set2, ResultSet)
set_union(Set1, Set2, ResultSet)
set_difference(Set1, Set2, ResultSet)
```

   όπου το `ResultSet` είναι η τομή, η ένωση και η διαφορά των συνόλων `Set1` και `Set2`, αντίστοιχα. Μπορείτε να θεωρήσετε δεδομένο ότι τα `Set1` και `Set2` δεν έχουν επαναλαμβανόμενα στοιχεία. Κάποια παραδείγματα εκτέλεσης:

```
?- set_intersection([a,b,c,d,e], [d,f,g,a], ResultSet).
ResultSet = [a,d]
?- set_union([a,b,c,d,e], [d,f,g,a], ResultSet).
ResultSet = [b,c,e,d,f,g,a]
?- set_difference([a,b,c,d,e], [d,f,g,a], ResultSet).
ResultSet = [b,c,e]
```

2. (α΄) Έστω ότι έχουμε έναν πλήρη γράφο[7] N κόμβων με βάρη στις ακμές του. Το βάρος μίας ακμής εκφράζει την απόσταση των κόμβων που είναι τα άκρα της. Τότε μπορούμε να τον αναπαραστήσουμε σε Prolog μέσω μίας λίστας $N-1$ λιστών, όπου η πρώτη εσωτερική λίστα περιλαμβάνει κατά σειρά τις αποστάσεις του 1ου κόμβου από τον 2ο, τον 3ο και ούτω καθεξής μέχρι και από τον N-οστό κόμβο, η δεύτερη εσωτερική λίστα περιλαμβάνει τις αποστάσεις του 2ου κόμβου από τον 3ο, τον 4ο, κλπ. και τέλος η $(N-1)$-οστή εσωτερική λίστα περιλαμβάνει μόνο την απόσταση του προτελευταίου κόμβου από τον τελευταίο. Για παράδειγμα, ο γράφος του διπλανού σχήματος μπορεί να αναπαρασταθεί από τη λίστα




```
[[2,5,10,15],[3,5,9],[2,7],[2]]
```

   Όμως, ίσως να μας ήταν χρήσιμο σε ένα πρόγραμμα Prolog να επεξεργαστούμε έναν τέτοιο γράφο σαν μία λίστα N λιστών, όπου κάθε εσωτερική λίστα περιλαμβάνει τις αποστάσεις ενός κόμβου από όλους τους κόμβους του γράφου (προηγούμενους και επόμενους, ακόμα και από τον εαυτό του, που, βέβαια, είναι 0). Δηλαδή, μπορεί να θέλαμε να επεξεργαστούμε τον εν λόγω γράφο σαν

```
[[0,2,5,10,15],[2,0,3,5,9],[5,3,0,2,7],[10,5,2,0,2],[15,9,7,2,0]]
```

[7] Σε έναν πλήρη γράφο, κάθε κόμβος συνδέεται μέσω ακμής με οποιονδήποτε άλλο.

Ορίστε σε Prolog το κατηγόρημα `alldist/2` έτσι ώστε όταν καλείται σαν `alldist(GIn, GOut)`, παίρνοντας στο `GIn` έναν γράφο με την πρώτη αναπαράστασή του, να επιστρέφει τη δεύτερη αναπαράσταση στο `GOut`. Δηλαδή:

```
?- alldist([[2,5,10,15],[3,5,9],[2,7],[2]], GOut).
GOut = [[0,2,5,10,15],[2,0,3,5,9],[5,3,0,2,7],[10,5,2,0,2],[15,9,7,2,0]]
```

(β′) Δίνεται η σχέση

$$\frac{A}{BC} + \frac{D}{EF} + \frac{G}{HI} = 1$$

στην οποία το ζητούμενο είναι να αντικατασταθούν τα εμφανιζόμενα σύμβολα με τα ψηφία από το 1 έως το 9 (διαφορετικό ψηφίο για κάθε σύμβολο), ώστε να είναι αληθής. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και γράψτε ένα πρόγραμμα σε ECL^{i}PS^e (είτε μέσω της βιβλιοθήκης `fd`, είτε της `ic`) που να το λύνει. Είναι προφανές ότι για κάθε λύση του προβλήματος, υπάρχουν και άλλες πέντε συμμετρικές με αυτήν, που προκύπτουν κάνοντας αντιμεταθέσεις στα κλάσματα της σχέσης. Μπορείτε να διατυπώσετε έτσι τους περιορισμούς ώστε να μην προκύπτουν συμμετρικές λύσεις;[8]

3. (α′) i. Υπάρχει περίπτωση το ελάχιστο μοντέλο ενός οριστικού προγράμματος να είναι το κενό σύνολο; Πότε μπορεί να συμβεί αυτό; Υπάρχει τότε κάποια ενδιαφέρουσα επίπτωση που θα μπορούσατε να σχολιάσετε;

   ii. Πιστεύετε ότι η βάση Herbrand είναι μοντέλο για κάθε οριστικό πρόγραμμα; Αν ναι, εξηγήστε την άποψή σας. Αν όχι, δώστε ένα παράδειγμα οριστικού προγράμματος για το οποίο η βάση Herbrand να μην είναι μοντέλο του. Απαντήστε στην ίδια ερώτηση και για την περίπτωση μη οριστικών προγραμμάτων.

   (β′) i. Αναφέρατε μία βασική διαφορά ανάμεσα στη μεθοδολογία των σημασιολογικών δικτύων και εκείνη των πλαισίων για την αναπαράσταση γνώσης. Πού εστιάζει η καθεμία από αυτές; Δώστε από ένα παράδειγμα γνωστικού πεδίου εφαρμογής που ταιριάζει η καθεμία.

   ii. Υλοποιήστε σε Prolog ένα μικρό παράδειγμα βάσης γνώσης σε μία από τις παραπάνω μεθοδολογίες (διαφορετικό από αυτό που χρησιμοποιήθηκε στις διαλέξεις).

[8] Ενημερωτικά, η μοναδική λύση του προβλήματος, αφαιρώντας τις συμμετρίες, είναι η

$$\frac{5}{34} + \frac{7}{68} + \frac{9}{12} = 1$$

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2010

1. (α΄) Για ποια από τα παρακάτω υπολογιστικά προβλήματα πιστεύετε ότι θα ήταν κατάλληλη η χρήση λογικού προγραμματισμού, και συγκεκριμένα της γλώσσας Prolog, για την αντιμετώπισή τους; Από εκείνα για τα οποία η απάντησή σας είναι καταφατική, για ποια θα ήταν χρήσιμη, κατά τη γνώμη σας, η εκμετάλλευση και τεχνικών προγραμματισμού με περιορισμούς; Σε κάθε περίπτωση, αιτιολογήστε στοιχειωδώς τις απαντήσεις σας (**συνολικά σε 20 γραμμές το πολύ**).

   i. Μετασχηματισμός Fourier
   ii. Κατασκευή προγράμματος εξετάσεων ενός πανεπιστημιακού τμήματος
   iii. Επίλυση του κύβου του Rubik
   iv. Υπολογισμός αορίστων ολοκληρωμάτων
   v. Υλοποίηση της εργασίας στην "Ανάπτυξη Λογισμικού"
   vi. Εύρεση συντομότερου μονοπατιού μεταξύ δεδομένων κόμβων σ΄ ένα γράφο
   vii. Έκδοση των αποτελεσμάτων των Πανελληνίων Εξετάσεων
   viii. Κατασκευή προγράμματος εργασίας των μηχανοδηγών του μετρό
   ix. Συντακτική ανάλυση προτάσεων φυσικής γλώσσας
   x. Επίλυση γραμμικού συστήματος 100 εξισώσεων με 100 αγνώστους

   (β΄) Στην πρώτη φάση του Παγκοσμίου Κυπέλλου Ποδοσφαίρου που αρχίζει αύριο στη Νότια Αφρική, οι ομάδες που συμμετέχουν έχουν χωρισθεί σε ομίλους, με σκοπό να δώσουν αγώνες μεταξύ τους, με όλους τους δυνατούς τρόπους. Ορίστε σε Prolog ένα κατηγόρημα `matches/2`, το οποίο όταν καλείται σαν `matches(Ts,Ms)`, με `Ts` να είναι η λίστα των ομάδων ενός ομίλου, να επιστρέφει στο `Ms` τη λίστα των αγώνων που πρέπει να δώσουν μεταξύ τους.[9] Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- matches([arg, nig, kor, gre], Ms).
Ms = [arg-nig, arg-kor, arg-gre, nig-kor, nig-gre, kor-gre]
```

2. (α΄) Αφού ολοκληρωθούν οι αγώνες ενός ομίλου στο Παγκόσμιο Κύπελλο Ποδοσφαίρου, κάθε ομάδα έχει συγκεντρώσει ένα πλήθος βαθμών, παίρνοντας 3 βαθμούς για κάθε νίκη της, 1 για κάθε ισοπαλία και 0 για τις ήττες. Ορίστε σε Prolog ένα κατηγόρημα `wdl/2`, το οποίο όταν καλείται σαν `wdl(P,M,WDL)`, δεδομένων των βαθμών `P` που συγκέντρωσε μία ομάδα σε `M` αγώνες που έδωσε στον όμιλό της, να επιστρέφει στο `WDL` έναν όρο της μορφής `W/D/L`, όπου τα `W`, `D` και `L` είναι πιθανοί αριθμοί των νικών, ισοπαλιών και ηττών που έκανε η ομάδα στους αγώνες του ομίλου.[10] Κάποια παραδείγματα εκτέλεσης:

```
?- wdl(5, 3, WDL).
WDL = 1/2/0
?- wdl(3, 3, WDL).
WDL = 0/3/0     --> ;
WDL = 1/0/2
?- wdl(8, 3, WDL).
no
```

[9] Οι ομάδες ενός ομίλου μπορεί να είναι οσεσδήποτε, όχι πάντοτε τέσσερις, όπως συμβαίνει στην πραγματικότητα. Επίσης, η σειρά των αγώνων στο αποτέλεσμα και η διάταξη των ομάδων σ΄ ένα αγώνα είναι αδιάφορες.

[10] Μπορείτε στην απάντησή σας να χρησιμοποιήσετε είτε απλή Prolog, είτε προγραμματισμό με περιορισμούς.

```
?- wdl(10, 9, WDL).
WDL = 1/7/1     --> ;
WDL = 2/4/3     --> ;
WDL = 3/1/5
```

(β΄) Στο πρόβλημα του σακιδίου (knapsack problem), έχουμε N αντικείμενα, όπου το i-οστό από αυτά έχει βάρος W_i και αξία V_i ($1 \leq i \leq N$). Επίσης, έχουμε ένα σακίδιο στο οποίο μπορούν να μπουν αντικείμενα μέχρι συνολικού βάρους C. Το ζητούμενο είναι ποια αντικείμενα από τα διαθέσιμα να τοποθετήσουμε στο σακίδιο, ώστε η συνολική αξία τους να είναι η μέγιστη δυνατή.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί, αντικειμενική συνάρτηση). Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σ΄ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α΄) Δίνεται το εξής λογικό πρόγραμμα:

```
p(g(X)) :- p(f(f(X))).
p(f(X)) :- p(g(g(X))).
p(f(f(g(f(g(a)))))).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(g(g(f(a))))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(g(g(f(a))))`. Επίσης, μπορεί το `p(f(g(a)))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(f(g(a)))`.

(β΄) Δίνεται η παρακάτω γνώση:

- Αν κάποιος είναι επιστήμονας, τότε διαβάζει βιβλία.
- Αν κάποιος είναι ιστορικός, τότε διαβάζει βιβλία ιστορίας.
- Αν κάποιος αγαπά την Αίγυπτο, τότε διαβάζει βιβλία τοπικής ιστορίας με τόπο την Αίγυπτο.
- Αν κάποιος διαβάζει κάτι, τότε του το κάνω δώρο.
- Τα βιβλία τοπικής ιστορίας είναι βιβλία ιστορίας.
- Τα βιβλία ιστορίας είναι βιβλία.
- Ένας ιστορικός είναι επιστήμονας.
- Η "Ιστορία των Αιγυπτιακών χρόνων" είναι βιβλίο τοπικής ιστορίας.
- Η "Ιστορία των Αιγυπτιακών χρόνων" έχει τόπο την Αίγυπτο.
- Το "Πόλεμος και Ειρήνη" είναι βιβλίο.
- Ο Βασίλης είναι ιστορικός.
- Ο Βασίλης αγαπά την Αίγυπτο.

i. Διακρίνετε κάποι-α/-ες μεθοδολογί-α/-ες αναπαράστασης γνώσης που να ταιριάζει να χρησιμοποιηθ-εί/-ούν για να αναπαραστήσουν την παραπάνω γνώση; Αιτιολογήστε την επιλογή σας.

ii. Γράψτε ένα Prolog πρόγραμμα που να αναπαριστά την παραπάνω γνώση και μέσω της Prolog να απαντά στην ερώτηση "Τι δώρο να κάνω στον Βασίλη;" Τι απαντήσεις δίνονται;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2010

1. (α΄) Η γνωστή σας υλοποίηση του κατηγορήματος `length/2` (σελ. 52 των σημειώσεων)

```
length([], 0).
length([_|Tail], N) :- length(Tail, N1), N is 1+N1.
```

μπορεί να χρησιμοποιηθεί για να υπολογισθεί το μήκος δεδομένης λίστας, π.χ.

```
?- length([a, b, [c, d], e], N).
N = 4
```

Θα μας ενδιέφερε όμως να μπορούσε να χρησιμοποιηθεί και αντίστροφα, δηλαδή, να κατασκευάζει μία λίστα από μεταβλητές δεδομένου μήκους. Για παράδειγμα:

```
?- length(L, 4).
L = [_194, _197, _200, _203]
```

Εξηγήστε γιατί η προηγούμενη υλοποίηση δεν είναι εντελώς σωστή, αν θέλουμε να χρησιμοποιήσουμε το `length/2` με τον δεύτερο τρόπο. Δώστε μία υλοποίηση του κατηγορήματος, τέτοια ώστε να μπορεί να χρησιμοποιηθεί αυτό τόσο για τον υπολογισμό του μήκους δεδομένης λίστας, όσο και για την κατασκευή λίστας από μεταβλητές δεδομένου μήκους.

(β΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `substodd/4` έτσι ώστε όταν αυτό καλείται σαν `substodd(X, L1, Y, L2)`, να επιστρέφει στο `L2` τη λίστα που προκύπτει αν αντικατασταθούν όλες οι εμφανίσεις του στοιχείου `X` σε περιττής τάξης θέσεις (1η, 3η, 5η, ...) στη λίστα `L1` με το στοιχείο `Y`. Για παράδειγμα:

```
?- substodd(c, [a, b, c, c, d, c, c, e, c, f], z, L).
L = [a, b, z, c, d, c, z, e, z, f]
?- substodd(4, [4, 4, 5, 2, 4], 0, L).
L = [0, 4, 5, 2, 0]
?- substodd(f, [a, b, c, d, e], g, L).
L = [a, b, c, d, e]
```

2. (α΄) Σε αθλητικές διοργανώσεις, όπως το Παγκόσμιο Κύπελλο Ποδοσφαίρου, συνήθως οι ομάδες χωρίζονται αρχικά σε ομίλους, με σκοπό να δώσουν αγώνες μεταξύ τους με όλους τους δυνατούς τρόπους. Από τους αγώνες αυτούς, οι ομάδες παίρνουν βαθμούς, 3 για κάθε νίκη, 1 για κάθε ισοπαλία και 0 για τις ήττες, όπου με βάση την τελική κατάταξή τους, καθορίζεται αν θα προχωρήσουν ή όχι στις επόμενες φάσεις της διοργάνωσης. Ορίστε σε Prolog ένα κατηγόρημα `guessscores/4`, το οποίο όταν καλείται σαν `guessscores(M, P, FG-AG, S)`, όπου `M` είναι το πλήθος των αγώνων που έδωσε μία ομάδα στη φάση των ομίλων, `P` οι βαθμοί που συγκέντρωσε και `FG`, `AG` συνολικά πόσα γκολ πέτυχε και δέχθηκε, αντίστοιχα, να επιστρέφει στο `S` μία λίστα με τα πιθανά σκορ της ομάδας στους αγώνες που έδωσε. Για παράδειγμα:

```
?- guessscores(3, 4, 2 - 3, S).
S = [0 - 0, 0 - 2, 2 - 1]     --> ;
S = [0 - 0, 0 - 3, 2 - 0]     --> ;
S = [0 - 0, 1 - 0, 1 - 3]     --> ;
.............................
S = [2 - 1, 0 - 2, 0 - 0]
```

(β′) Σ′ ένα πανεπιστημιακό τμήμα, κατά τη διάρκεια ενός ακαδημαϊκού εξαμήνου προσφέρθηκε ένας αριθμός μαθημάτων τα οποία πρόκειται να εξετασθούν στο τέλος του εξαμήνου. Έστω ότι υπάρχουν N_1, N_2, N_3 και N_4 μαθήματα του 1ου, 2ου, 3ου και 4ου έτους, αντίστοιχα, τα οποία πρόκειται να εξετασθούν σε D συνεχόμενες ημέρες κατά την εξεταστική περίοδο. Θεωρήστε ότι κάθε ημέρα υπάρχει η δυνατότητα εξέτασης τριών το πολύ μαθημάτων (ισχύει όμως $N_1 + N_2 + N_3 + N_4 \leq 3 \cdot D$) και ότι δεν πρέπει να εξετασθούν την ίδια ημέρα δύο μαθήματα του ίδιου έτους. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και σκιαγραφήστε πώς θα το αντιμετωπίζατε σ′ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e. Αν ενδιέφερε να βρεθεί όχι απλώς μία λύση, αλλά η καλύτερη, πώς θα ορίζατε το "καλύτερη" και πώς θα τη βρίσκατε δουλεύοντας σ′ ένα περιβάλλον λογικού προγραμματισμού με περιορισμούς;

3. (α′) Έστω μία γλώσσα πρώτης τάξης στην οποία $p/1, q/1 \in P$, $a/0, b/0 \in F$ και $x \in V$. Δώστε μία οριστική πρόταση αυτής της γλώσσας και δύο μοντέλα της πρότασης που να είναι ξένα μεταξύ τους. Αν ένα οριστικό πρόγραμμα αποτελείται μόνο από αυτή την πρόταση, ποιο είναι το ελάχιστο μοντέλο του; Έχετε να κάνετε κάποιο ενδιαφέρον σχόλιο;

   (β′) i. Δίνεται το Prolog πρόγραμμα:

```
p(a,b).
p(b,c).
p(c,d).

q(a,v1).
q(b,v2).

z(a,v3).
z(c,v4).
```

   Δώστε ονόματα στα σύμβολα `p`, `q`, `z` και στα `a`, `b`, `c`, `d`, `v1`, `v2`, `v3`, `v4`, ώστε το παραπάνω πρόγραμμα να έχει κάποια φυσική σημασία που να ταιριάζει σε κάποια μεθοδολογία αναπαράστασης γνώσης, αναφέροντας ποια.

   ii. Σε τι διαφέρει συντακτικά η Datalog από την Prolog; Για τα θέματα σημασιολογίας (semantics), τι διαφορές/ομοιότητες έχετε να παρατηρήσετε;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2009

1. (α΄) Πώς νοείται η χρήση *συσσωρευτή* στον προγραμματισμό σε Prolog; Ορίστε σε Prolog ένα κατηγόρημα για τον υπολογισμό του παραγοντικού ενός δεδομένου θετικού ακεραίου με δύο τρόπους, έναν με χρήση συσσωρευτή και έναν χωρίς. Ποια είναι τα πλεονεκτήματα/μειονεκτήματα της μίας και ποια της άλλης προσέγγισης; Πώς σχετίζεται η χρήση συσσωρευτή με αυτό που ονομάζουμε *αναδρομή ουράς*;

   (β΄) Ορίστε σε Prolog ένα κατηγόρημα `emo/2`, το οποίο όταν καλείται σαν `emo(L1,L2)`,[11] με δεδομένη τη λίστα `L1`, να επιστρέφει στη μεταβλητή `L2` μία αναδιάταξη της `L1`, στην οποία πρώτο να είναι το πρώτο στοιχείο της `L1`, δεύτερο το τελευταίο στοιχείο της `L1`, μετά το δεύτερο στοιχείο της `L1`, μετά το προτελευταίο της, κ.ο.κ. Τα παρακάτω παραδείγματα εκτέλεσης αποσαφηνίζουν πλήρως τι ζητείται.

```
?- emo([1,2,3,4,5,6,7], L).
L = [1,7,2,6,3,5,4]
?- emo([a,b,c,d,e,f,g,h,i,j], L).
L = [a,j,b,i,c,h,d,g,e,f]
```

2. (α΄) Θεωρήστε την αναπαράσταση των θετικών ακεραίων σαν μία λίστα των ψηφίων τους. Για παράδειγμα, ο αριθμός 287 παριστάνεται με τη λίστα `[2,8,7]`. Ορίστε σε Prolog ένα κατηγόρημα `sum_numbs/3`, το οποίο όταν καλείται σαν `sum_numbs(L1,L2,L3)`, με δεδομένες τις λίστες `L1` και `L2`, που αναπαριστούν δύο θετικούς ακεραίους σύμφωνα με το προηγούμενο, να επιστρέφει στη μεταβλητή `L3` τη λίστα που αναπαριστά το άθροισμα των δύο αυτών ακεραίων. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- sum_numbs([3,4,1], [5,7,8], L).
L = [9,1,9]
?- sum_numbs([5,3,8,0,7], [6,4,9], L).
L = [5,4,4,5,6]
?- sum_numbs([8,7], [9,9,9,9,9,9,9,9,2,8], L).
L = [1,0,0,0,0,0,0,0,0,1,5]
```

   Πιθανότατα, στην υλοποίησή σας έχετε χρησιμοποιήσει το ενσωματωμένο κατηγόρημα `is/2`. Αν ναι, θα μπορούσατε να δώσετε λύση χωρίς αυτό; Πώς θα επιτυγχάνατε κάτι τέτοιο;

   (β΄) Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο N πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς 1, 2, 3, …, N. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει M συνδυασμούς πτήσεων P_i ($1 \leq i \leq M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις 1, 2, 3, …, N και, επίσης, τα P_i δεν είναι κατ΄ ανάγκη ξένα μεταξύ τους. Οι συνδυασμοί αυτοί είναι έτσι κατασκευασμένοι ώστε να είναι δυνατόν λόγω κανονισμών, συμβάσεων κλπ. να πραγματοποιηθούν ο καθένας, δηλαδή όλες οι πτήσεις που περιλαμβάνει, με έναν από τους διαθέσιμους κυβερνήτες της εταιρείας. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί ξένοι μεταξύ τους, που καλύπτουν ακριβώς τις πτήσεις της εταιρείας, με σκοπό να ανατεθούν σε συγκεκριμένους κυβερνήτες. Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος (έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.)

[11] Το όνομα του κατηγορήματος προέρχεται από τα αρχικά των "**e**nds to **m**iddle **o**rder".

στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος.

Για να γίνει περισσότερο σαφές το ζητούμενο, δείτε το εξής παράδειγμα:

Έστω $N = 5$, $M = 7$ και

i	P_i	C_i
1	$\{1, 2\}$	6
2	$\{1, 2, 3\}$	7
3	$\{1, 5\}$	5
4	$\{2\}$	3
5	$\{3, 4\}$	4
6	$\{4, 5\}$	8
7	$\{5\}$	6

Τότε, η βέλτιστη λύση είναι η $\{\{1, 5\}, \{2\}, \{3, 4\}\}$, με κόστος 12 (=5+3+4), ενώ οι άλλες πιθανές λύσεις $\{\{1, 2\}, \{3, 4\}, \{5\}\}$ και $\{\{1, 2, 3\}, \{4, 5\}\}$ έχουν κόστη 16 (=6+4+6) και 15 (=7+8), αντίστοιχα.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(X,f(Y)) :- p(f(X),Y).
p(f(Y),X) :- p(X,f(Y)).
p(f(a),b).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `p(a,a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(a,a)`. Επίσης, μπορεί το `p(b,f(b))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `p(b,f(b))`.

(β′) i. Περιγράψτε επιγραμματικά (**σε 5 γραμμές το πολύ**) τη μεθοδολογία αναπαράστασης γνώσης με χρήση κανόνων *if_then* και τη μεθοδολογία αναπαράστασης γνώσης με χρήση πλαισίων. Σε ποιές περιοχές εφαρμογών ενδείκνυται να χρησιμοποιηθεί η μεν και σε ποιές η δε;

ii. Πιστεύετε ότι θα μπορούσε και ότι θα είχε νόημα να συνδυασθεί η μεθοδολογία αναπαράστασης γνώσης με χρήση κανόνων *if_then* με αυτή μέσω πλαισίων, και, αν ναι, με ποιον τρόπο; Δώστε ένα παράδειγμα υλοποίησης μίας τέτοιας υβριδικής βάσης γνώσης σε Prolog και ένα περίγραμμα/σκελετό υλοποίησης του συμπερασμού επάνω σ′ αυτή.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2009

1. (α΄) Στη λογική, ο τύπος

$$(P \longrightarrow Q) \quad \longleftrightarrow \quad (\neg P \vee Q)$$

είναι ταυτολογία, δηλαδή είναι πάντοτε αληθής. Ορίστε, μέσω του ενσωματωμένου κατηγορήματος `op/3`, κατάλληλους τελεστές Prolog, ώστε να μπορεί να διατυπωθεί ο παραπάνω τύπος σαν ένα γεγονός Prolog, για παράδειγμα ως εξής:

```
P ==> Q <==> ^P \/ Q.
```

(β΄) Έστω ότι έχετε στη διάθεσή σας ένα "ελλιπές" σύστημα Prolog, στο οποίο δεν υπάρχει το γνωστό σας ενσωματωμένο κατηγόρημα `is/2` για την αποτίμηση αριθμητικών εκφράσεων. Αντ΄ αυτού, υπάρχει μία απλή εκδοχή του, το `smplis/2`, το οποίο είναι σε θέση να κάνει τις τέσσερις στοιχειώδεις πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση), αλλά μόνο μεταξύ δύο αριθμών. Δηλαδή, δεν είναι σε θέση να υπολογίσει την τιμή πιο πολύπλοκων αριθμητικών εκφράσεων. Όταν του δοθούν τέτοιες εκφράσεις, απλά αποτυγχάνει. Κάποια παραδείγματα χρήσης του `smplis/2` είναι τα εξής:

```
?- X smplis 5 + 4.2.
X = 9.2
?- X smplis 14 - 3 * 4.
no
```

Ορίστε σε Prolog ένα κατηγόρημα `newis/2`, το οποίο, με τη βοήθεια του `smplis/2`, να μπορεί να υπολογίζει την τιμή και αριθμητικών εκφράσεων που εμπλέκουν τις τέσσερις βασικές πράξεις. Δηλαδή, να συμπεριφέρεται στο θέμα αυτό, όπως το γνωστό `is/2`, το οποίο όμως δεν είναι διαθέσιμο. Κάποια παραδείγματα εκτέλεσης:

```
?- X newis 17 - 11.
X = 6
?- X newis 5 + 3 * 2.
X = 11
?- X newis (20 - 4) / 5 + 7 - 2 * 4.
X = 2.2
```

2. (α΄) Ορίστε σε Prolog ένα κατηγόρημα `repeated/3` με την εξής συμπεριφορά: Όταν καλείται σαν `repeated(L1,L2,N)`, να επιτυγχάνει, αν η λίστα `L2` είναι το αποτέλεσμα της συνένωσης `N` φορών της λίστας `L1` με τον εαυτό της. Για παράδειγμα:

```
?- repeated([a,b,c,d], [a,b,c,d,a,b,c,d,a,b,c,d], 3).
yes
```

Υλοποιήστε με τέτοιο τρόπο το κατηγόρημα `repeated/3`, ώστε να είναι δυνατόν να κληθεί και όπως φαίνεται στα ακόλουθα παραδείγματα:

```
?- repeated(L1, [a,b,c,d,a,b,c,d,a,b,c,d], 3).
L1 = [a,b,c,d]
?- repeated([a,b,c,d], [a,b,c,d,a,b,c,d,a,b,c,d], N).
N = 3
```

```
?- repeated(L1, [a,b,c,d,a,b,c,d,a,b,c,d], N).
L1 = [a,b,c,d]
N = 3 -> ;
L1 = [a,b,c,d,a,b,c,d,a,b,c,d]
N = 1
?- repeated([a,b,c,d], L2, 3).
L2 = [a,b,c,d,a,b,c,d,a,b,c,d]
```

(β′) Μία αεροπορική εταιρεία έχει προγραμματίσει να εκτελέσει για μία προκαθορισμένη χρονική περίοδο N πτήσεις, στις οποίες μπορεί να αναφέρεται κανείς με τους κωδικούς 1, 2, 3, ..., N. Επιπλέον, μέσω κάποιας μεθόδου που εφάρμοσε, έχει δημιουργήσει M συνδυασμούς πτήσεων P_i ($1 \leq i \leq M$). Δηλαδή, κάθε P_i περιλαμβάνει κάποιες από τις πτήσεις 1, 2, 3, ..., N και, επίσης, τα P_i δεν είναι κατ′ ανάγκη ξένα μεταξύ τους. Το ζητούμενο είναι να επιλεγούν κάποιοι συνδυασμοί, όχι κατ′ ανάγκη ξένοι μεταξύ τους, τέτοιοι ώστε κάθε πτήση να περιλαμβάνεται σε τουλάχιστον έναν από τους επιλεγμένους συνδυασμούς.[12] Τέλος, αν ένας συνδυασμός πτήσεων P_i έχει ένα κόστος[13] στην εταιρεία C_i, οι συνδυασμοί που θα επιλεγούν πρέπει να είναι αυτοί που προκαλούν το ελάχιστο συνολικό κόστος.

Για να γίνει περισσότερο σαφές το ζητούμενο, δείτε το εξής παράδειγμα:

Έστω $N = 5$, $M = 7$ και

i	P_i	C_i
1	$\{1, 2\}$	4
2	$\{1, 2, 3\}$	5
3	$\{2, 4, 5\}$	6
4	$\{2, 5\}$	2
5	$\{3, 4\}$	7
6	$\{3, 4, 5\}$	8
7	$\{5\}$	5

Τότε, η βέλτιστη λύση είναι η $\{\{1, 2, 3\}, \{2, 4, 5\}\}$, με κόστος 11 (=5+6), ενώ υπάρχουν πολλές άλλες πιθανές λύσης, καμία με κόστος μικρότερο του 11, για παράδειγμα η $\{\{1, 2\}, \{3, 4, 5\}\}$ με κόστος 12 (=4+8).

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και διατυπώστε τη συνάρτηση κόστους βάσει της οποίας θα πρέπει να βρεθεί η βέλτιστη λύση.

3. (α′)
   i. Πότε η βάση Herbrand που αντιστοιχεί σ′ ένα λογικό πρόγραμμα είναι απειροσύνολο; Δώστε δύο πολύ απλά παραδείγματα οριστικών προγραμμάτων με βάσεις Herbrand που είναι απειροσύνολα και στο μεν ένα το ελάχιστο μοντέλο να είναι πεπερασμένο, στο δε άλλο απειροσύνολο.
   ii. Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Τι επίπτωση έχει αυτό το γεγονός;
   iii. Γιατί κατά τη γνώμη σας ενώ η έννοια του μοντέλου ορίζεται για οποιοδήποτε λογικό πρόγραμμα, αυτή του ελάχιστου μοντέλου ορίζεται μόνο για οριστικά προγράμματα;

   (β′) Περιγράψτε επιγραμματικά (**σε 5 γραμμές το πολύ**) τη μεθοδολογία αναπαράστασης γνώσης με χρήση κανόνων *if_then* και τη μεθοδολογία αναπαράστασης γνώσης με χρήση σημασιολογικών δικτύων. Δώστε ένα παράδειγμα περιοχής εφαρμογών που ενδείκνυται να χρησιμοποιηθεί η πρώτη μεθοδολογία για την αναπαράσταση γνώσης και ένα παράδειγμα για τη δεύτερη. Έχετε υπόψη σας μία υλοποίηση *if_then* κανόνων και μία υλοποίηση σημασιολογικών δικτύων σε Prolog. Αιτιολογήστε τις επιλογές που έγιναν στις αναπαραστάσεις. (Αν δεν έχετε κάτι υπόψη σας, προτείνετε από μία υλοποίηση, αιτιολογώντας τις επιλογές σας.)

[12] Η εταιρεία, στη συνέχεια, θα αντιστοιχήσει σε κάθε έναν από τους επιλεγμένους συνδυασμούς έναν κυβερνήτη και κάθε πτήση θα εκτελεσθεί από έναν από τους υπεύθυνους κυβερνήτες των συνδυασμών στους οποίους ανήκει (αυτό, όμως, δεν είναι κάτι που αφορά το πρόβλημα που συζητάμε).

[13] έξοδα διανυκτερεύσεων, αποζημιώσεις εκτός έδρας, πληρωμή υπερωριών κλπ.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2008

1. (α΄) Όταν το ενσωματωμένο κατηγόρημα `findall/3` της Prolog, που περιλαμβάνεται στο πρότυπο της γλώσσας, καλείται σαν `findall(X,Goal,L)` αυτό που κάνει είναι να επιστρέψει στο `L` μία λίστα με όλες τις δυνατές τιμές της μεταβλητής `X` που προκύπτουν από όλες τις δυνατές λύσεις του στόχου `Goal`, όπως αυτές μπορούν να βρεθούν μέσω οπισθοδρόμησης. Πιστεύετε ότι το κατηγόρημα αυτό θα μπορούσε να υλοποιηθεί σε "καθαρή" Prolog, δηλαδή χωρίς τη χρήση "εκτός λογικής" ενσωματωμένων κατηγορημάτων (π.χ. `assert`/`retract`); Αν ναι, δώστε μία τέτοια υλοποίηση. Αν όχι, εξηγήστε γιατί.

   (β΄) Ορίστε σε Prolog ένα κατηγόρημα `multi_split/3`, το οποίο όταν καλείται με πρώτο όρισμα μία λίστα `L` και δεύτερο όρισμα ένα θετικό ακέραιο `N`, να επιστρέφει στο τρίτο όρισμα μία λίστα με `N` ακριβώς στοιχεία, τα οποία να είναι **μη κενές** λίστες, τέτοιες ώστε αν συνενωθούν, με τη σειρά που εμφανίζονται, να προκύπτει η λίστα `L`. Το κατηγόρημα που θα γράψετε να επιστρέφει μέσω οπισθοδρόμησης όλες τις εναλλακτικές λύσεις του προβλήματος. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- multi_split([a,b,c,d,e], 3, SL).
SL = [[a],[b],[c,d,e]] -> ;
SL = [[a],[b,c],[d,e]] -> ;
SL = [[a],[b,c,d],[e]] -> ;
SL = [[a,b],[c],[d,e]] -> ;
SL = [[a,b],[c,d],[e]] -> ;
SL = [[a,b,c],[d],[e]] -> ;
no
?- multi_split([a,b,c,d,e,f,g,h], 8, SL).
SL = [[a],[b],[c],[d],[e],[f],[g],[h]] -> ;
no
?- multi_split([a,b,c,d,e,f,g,h], 9, SL).
no
```

2. (α΄) Ας δούμε αν μπορούμε να βάλουμε στην Prolog ιδέες από την Logo. Έχουμε μία χελώνα η οποία μπορεί να κινείται σ΄ ένα διδιάστατο σύστημα αξόνων. Κάθε χρονική στιγμή, η χελώνα βρίσκεται σ΄ ένα σημείο του επιπέδου, που ορίζεται από τις συντεταγμένες του, και έχει ένα συγκεκριμένο προσανατολισμό, σύμφωνα με τα τέσσερα σημεία του ορίζοντα. Η χελώνα μπορεί να εκτελέσει δύο μόνο εντολές, την `step`, οπότε μετακινείται κατά μία μονάδα μήκους προς την κατεύθυνση που είναι προσανατολισμένη, και την `rotate`, οπότε στρέφεται κατά 90^o δεξιά. Ορίστε σε Prolog ένα κατηγόρημα `closed_path/2`, το οποίο να καλείται με πρώτο όρισμα ένα "πρόγραμμα" για εκτέλεση από τη χελώνα, υπό τη μορφή μίας λίστας από τις εντολές που αναγνωρίζει, και το οποίο να επιτυγχάνει στην περίπτωση που, εκτελώντας το πρόγραμμα, η χελώνα επιστρέφει στο σημείο από το οποίο ξεκίνησε. Σε περίπτωση επιτυχίας, στο δεύτερο όρισμα να επιστρέφεται η απόσταση που διάνυσε η χελώνα. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- closed_path([step,step,step,rotate,step,step,rotate,
                step,step,step,rotate,step,step], D).
D = 10
?- closed_path([step,step,rotate,rotate,step], D).
no
```

(β′) Μία εταιρεία που λειτουργεί 24 ώρες την ημέρα και 7 ημέρες την εβδομάδα απασχολεί 50 υπαλλήλους. Προφανώς, οι υπάλληλοι πρέπει να εργάζονται σε βάρδιες. Κάθε υπάλληλος μπορεί, για κάποια ημέρα, να εργάζεται σε πρωινή βάρδια (M), ή απογευματινή (E), ή νυχτερινή (N), ή μπορεί να έχει ανάπαυση (R). Η εταιρεία θέλει να δημιουργήσει ένα πρόγραμμα εργασίας των υπαλλήλων της για ένα διάστημα 30 συνεχόμενων ημερών, στο οποίο όμως να ισχύουν οι παρακάτω κανόνες:

- Κάθε ημέρα πρέπει να εργάζονται τουλάχιστον 15 υπάλληλοι στην πρωινή βάρδια, τουλάχιστον 10 στην απογευματινή και τουλάχιστον 8 στην νυχτερινή.
- Κανένας υπάλληλος δεν επιτρέπεται αν μία ημέρα είχε νυχτερινή βάρδια, την επόμενη να έχει πρωινή.
- Κάθε υπάλληλος πρέπει σε οποιεσδήποτε 7 συνεχόμενες ημέρες του προγράμματος εργασίας (από την 1η έως την 7η, από την 2η έως την 8η, κ.ο.κ.) να έχει τουλάχιστον δύο ημέρες ανάπαυσης.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και επειδή, προφανώς, μπορεί να έχει πολλές λύσεις, προτείνετε μία εύλογη αντικειμενική συνάρτηση που να ποσοτικοποιεί την ποιότητα (ή το κόστος) μίας λύσης, οπότε πλέον έχουμε ένα πρόβλημα βελτιστοποίησης. Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(f(h(a))).
p(h(X)) :- p(X).
p(X) :- q(h(X)).
q(X) :- p(f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Βρείτε το ελάχιστο μοντέλο του προγράμματος, εφαρμόζοντας την κατάλληλη μέθοδο για το σκοπό αυτό. Μπορεί το `q(h(f(a)))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε την τομή όλων των μοντέλων που περιλαμβάνουν και το `q(h(f(a)))`.

(β′) Δίνονται οι παρακάτω κανόνες:

```
p(X) :- q(X).      q(X) :- z(X).
p(X) :- v(X,X).    v(X,Y) :- w1(X) & w2(Y) & w3(X,Y).
```

και τα γεγονότα:

```
p(a).      p(b).      p(c).
z(1).      z(2).
w1(d).     w1(e).     w1(f).
w2(d).     w2(e).     w2(g).
w3(d,e).   w3(e,e).   w3(e,f).
```

i. Κάντε τις κατάλληλες διορθώσεις ώστε το παραπάνω να αποτελεί ένα Datalog πρόγραμμα (δηλαδή, να γίνεται διαχωρισμός των IDB και EDB κατηγορημάτων).

ii. Σχεδιάστε τον γράφο εξαρτήσεων των κατηγορημάτων του νέου προγράμματος.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2008

1. (α΄) Σχολιάστε σε **15 γραμμές το πολύ** τα παρακάτω:

   i. "Οι τελεστές είναι βασικό συστατικό της Prolog. Χωρίς αυτούς δεν θα μπορούσαν να λυθούν πολλά προβλήματα."

   ii. "Είναι περιττό που έχουμε τα `=/2` και `\=/2` σαν ενσωματωμένα κατηγορήματα στην Prolog. Θα μπορούσαν να υλοποιηθούν πολύ απλά απ΄ ευθείας στη γλώσσα."

   iii. "Στα προγράμματα Prolog πρέπει να αποφεύγουμε όσο είναι δυνατόν την οπισθοδρόμηση, γιατί καθυστερεί πολύ η εκτέλεσή τους."

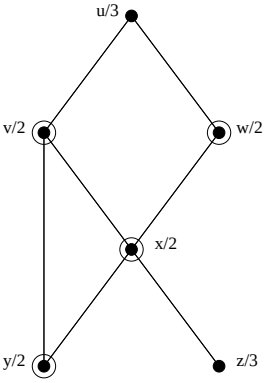
   iv. "Αν θέλουμε να εισάγουμε διάφορα στοιχεία σε μία λίστα, είναι προτιμότερο αυτό να γίνεται στο τέλος της, παρά στην αρχή της."

   v. "Η αποκοπή (`!/2`) μπορεί να υλοποιηθεί πολύ εύκολα σε απλή Prolog."

   (β΄) Ορίστε σε Prolog ένα κατηγόρημα `cartprod/2`, το οποίο όταν καλείται με ένα σύνολο συνόλων σαν πρώτο όρισμα (στη μορφή μίας λίστας λιστών), να επιστρέφει στο δεύτερο όρισμα το καρτεσιανό γινόμενο των συνόλων, σαν ένα σύνολο (λίστα) πλειάδων, όπου κάθε πλειάδα είναι μία λίστα στοιχείων. Ένα παράδειγμα εκτέλεσης είναι το εξής:

```
?- cartprod([[a,b,c],[1,2],[x,y,z,w]], CP).
CP = [[a,1,x],[a,1,y],[a,1,z],[a,1,w],[a,2,x],[a,2,y],[a,2,z],[a,2,w],
      [b,1,x],[b,1,y],[b,1,z],[b,1,w],[b,2,x],[b,2,y],[b,2,z],[b,2,w],
      [c,1,x],[c,1,y],[c,1,z],[c,1,w],[c,2,x],[c,2,y],[c,2,z],[c,2,w]]
```

2. (α΄) Ας θεωρήσουμε τους μη κατευθυνόμενους γράφους με **μοναδιαία** κόστη στις ακμές. Ένας τέτοιος γράφος μπορεί να αναπαρασταθεί από έναν όρο Prolog της μορφής `graph(Nodes,Edges)`, όπου `Nodes` είναι η λίστα των κόμβων του γράφου και `Edges` η λίστα των ακμών του. Κάθε ακμή είναι της μορφής `e(Node1,Node2)`. *Απόσταση* δύο κόμβων του γράφου είναι το μήκος (πλήθος ακμών) του συντομότερου μονοπατιού που τους συνδέει. *Εκκεντρικότητα* ενός κόμβου του γράφου λέγεται η απόσταση που έχει ο κόμβος από τον πιο απομακρυσμένο κόμβο απ΄ αυτόν. *Κέντρο* του γράφου είναι το σύνολο των κόμβων του με ελάχιστη εκκεντρικότητα. Για παράδειγμα, στον γράφο του διπλανού σχήματος, σε κάθε κόμβο φαίνεται και η εκκεντρικότητά του και οι κυκλωμένοι κόμβοι (`v`, `w`, `x` και `y`) αποτελούν το κέντρο του. Ορίστε σε Prolog ένα κατηγόρημα `grcenter/2`, το οποίο όταν καλείται με έναν γράφο στο πρώτο όρισμα, να επιστρέφει στο δεύτερο όρισμα το κέντρο του γράφου. Ένα παράδειγμα εκτέλεσης είναι το εξής:




```
?- grcenter(graph([u,v,w,x,y,z], [e(u,v),e(u,w),e(v,x),e(v,y),
                                  e(x,y),e(x,z),e(x,w)]), Center).
Center = [v,w,x,y]
```

   Βλέπετε κάποια πρακτική χρησιμότητα στο να θέλουμε να βρούμε το κέντρο ενός γράφου;

   (β΄) Σε μία συνδυαστική δημοπρασία (combinatorial auction), ο δημοπράτης διαθέτει προς πώληση N αντικείμενα, έστω αυτά του συνόλου $O = \{o_1, o_2, \ldots, o_N\}$, και οι υποψήφιοι αγοραστές μπορούν να καταθέσουν συγκεκριμένες προσφορές για ομάδες από αντικείμενα που επιλέγουν αυτοί. Έστω ότι τελικά έχουν κατατεθεί M προσφορές, αυτές του συνόλου

$B = \{b_1, b_2, \ldots, b_M\}$, όπου για κάθε b_i υπάρχει μία προσφερόμενη τιμή v_i και ισχύει $b_i \subseteq O$. Το πρόβλημα του δημοπράτη έγκειται στο να βρει ποιες προσφορές να κάνει αποδεκτές, έτσι ώστε να μεγιστοποιήσει το ποσό που θα εισπράξει. Δεν είναι απαραίτητο να πουληθούν όλα τα προϊόντα, αλλά, φυσικά, δεν μπορεί να πουληθεί ένα προϊόν δύο φορές. Δηλαδή, αν δύο ή περισσότερες προσφορές περιλαμβάνουν το ίδιο προϊόν, το πολύ μία από αυτές μπορεί να γίνει αποδεκτή.

Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί, αντικειμενική συνάρτηση). Τέλος, σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

3. (α′) i. Γιατί την έννοια του ελάχιστου μοντέλου την ορίζουμε μόνο για οριστικά προγράμματα; Δείξτε μέσω ενός παραδείγματος ότι δεν είναι πάντοτε δυνατόν να έχουμε ελάχιστο μοντέλο για ένα πρόγραμμα που δεν είναι οριστικό.

   ii. Ποια είναι η πρακτική σημασία της σημασιολογίας σταθερού σημείου και ποια αυτή της λειτουργικής σημασιολογίας;

   iii. Το τελικό συμπέρασμα της λειτουργικής σημασιολογίας, ότι, δηλαδή, το σύνολο επιτυχίας ενός οριστικού προγράμματος ταυτίζεται με το ελάχιστο μοντέλο του προγράμματος, αναφέρεται σε οριστικά προγράμματα. Πιστεύετε ότι θα ήταν χρήσιμο και δυνατό να εφαρμόσουμε τις αρχές της λειτουργικής σημασιολογίας και σε μη οριστικά προγράμματα;

   (β′) Περιγράψτε επιγραμματικά (σε **5 γραμμές το πολύ**) *i)* τα σημασιολογικά δίκτυα και *ii)* τα πλαίσια. Δώστε ένα παράδειγμα περιοχής εφαρμογών που ενδείκνυται να χρησιμοποιηθούν τα πρώτα για την αναπαράσταση γνώσης και ένα παράδειγμα για τα δεύτερα. Έχετε υπόψη σας μία υλοποίηση σημασιολογικών δικτύων και μία υλοποίηση πλαισίων σε Prolog. Αιτιολογήστε τις επιλογές που έγιναν στις αναπαραστάσεις. (Αν δεν έχετε κάτι υπόψη σας, προτείνετε από μία υλοποίηση, αιτιολογώντας τις επιλογές σας.)

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2007

1. (α΄) Γιατί στην Prolog το θέμα της άρνησης είναι δύσκολα διαχειρίσιμο, ενώ στη λογική πρώτης τάξης δεν τίθεται τέτοιο ζήτημα; Τι προβλήματα υπάρχουν με την υλοποίηση της άρνησης στην Prolog σύμφωνα με τη λογική της *άρνησης ως αποτυχία* (negation as failure); Δώστε ένα συγκεκριμένο παράδειγμα που να επιδεικνύει την προβληματική συμπεριφορά της άρνησης ως αποτυχία στην Prolog και αναφέρατε τι λύση υιοθετούν τα συστήματα Prolog για να απαλύνουν κάπως το πρόβλημα αυτό.

   (β΄) Ορίστε σε Prolog τα κατηγορήματα `unify/2` και `unifiable/2`, τα οποία όταν καλούνται σαν `unify(T1,T2)` και `unifiable(T1,T2)`, αντίστοιχα, να επιτυγχάνουν αν οι όροι `T1` και `T2` είναι ενοποιήσιμοι, αλλά το μεν `unify/2` να προκαλεί και την ενοποίηση των όρων, ενώ το `unifiable/2` όχι.

2. Ορίστε σε Prolog δύο κατηγορήματα `longestdecsub/2` και `longestincsub/2` τα οποία να καλούνται σαν `longestdecsub(L1,L2)` και `longestincsub(L1,L2)`, αντίστοιχα, δίνοντας στη μεταβλητή `L1` μία λίστα από αριθμούς. Το `longestdecsub/2` να επιστρέφει στη μεταβλητή `L2`, μέσω οπισθοδρόμησης, όλες τις λίστες μεγίστου μήκους με στοιχεία από τη λίστα `L1`, που είναι γνησίως φθίνουσες, και τα στοιχεία τους είναι με τη σειρά που εμφανίζονται και στη λίστα `L1`, όχι όμως κατ΄ ανάγκη συνεχόμενα σ΄ αυτήν. Το `longestincsub/2` διαφέρει από το `longestdecsub/2` μόνο ως προς το ότι η λίστα που επιστρέφει πρέπει να είναι γνησίως αύξουσα. Θα εκτιμηθεί ιδιαιτέρως η επαναχρησιμοποίηση ορισμών για την υλοποίηση των δύο κατηγορημάτων. Παραδείγματα εκτέλεσης είναι τα εξής:[14]

```
?- longestdecsub([3,2,7,4,6,1,5], Sub).
Sub = [3,2,1]  -> ;
Sub = [7,4,1]  -> ;
Sub = [7,6,1]  -> ;
Sub = [7,6,5]  -> ;
no
?- longestincsub([2,4,6,1,3,5,7], Sub).
Sub = [2,4,6,7]  -> ;
Sub = [2,4,5,7]  -> ;
Sub = [2,3,5,7]  -> ;
Sub = [1,3,5,7]  -> ;
no
```

3. (α΄) Έστω το οριστικό πρόγραμμα P:

```
p(a,c)  ←
p(b,c)  ←
p(X,X)  ←
p(X,Y)  ←  p(Y,X)
p(X,Z)  ←  p(X,Y), p(Y,Z)
```

[14] Το θεώρημα των Erdös-Szekeres (1935) λέει ότι κάθε ακολουθία από $m \cdot n+1$ **διαφορετικούς** αριθμούς περιέχει οπωσδήποτε είτε κάποια γνησίως φθίνουσα ακολουθία από $m+1$ αριθμούς, είτε κάποια γνησίως αύξουσα ακολουθία από $n+1$ αριθμούς. Στα παραδείγματα της εκφώνησης, για $m = 2$ και $n = 3$, βρίσκουμε, στο μεν πρώτο, γνησίως φθίνουσες ακολουθίες μήκους $2+1 = 3$, στο δε δεύτερο, γνησίως αύξουσες ακολουθίες μήκους $3+1 = 4$, άρα το θεώρημα επαληθεύεται για τα δεδομένα αυτά.

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Κατασκευάστε, σύμφωνα με το αποτέλεσμα της σημασιολογίας σταθερού σημείου, το ελάχιστο μοντέλο αυτού του προγράμματος. Δώστε μία SLD-απόρριψη του $P \cup \{\leftarrow \texttt{p(a,b)}\}$. Θα γράφατε ποτέ αυτό το οριστικό πρόγραμμα σε Prolog, φυσικά μετά τις κατάλληλες συντακτικές τροποποιήσεις; Αν όχι, γιατί;

(β΄) Η βιβλιοθήκη `ic` της ECL^{i}PS^e υποστηρίζει τον περιορισμό `alldifferent/1`, ο οποίος όταν κληθεί με όρισμα μία λίστα από μεταβλητές πεδίων, επιβάλλει αυτές οι μεταβλητές να πάρουν τελικά διαφορετικές τιμές. Δώστε μία πιθανή υλοποίηση του περιορισμού αυτού, χρησιμοποιώντας άλλους περιορισμούς της ECL^{i}PS^e που γνωρίζετε.

Εκτός από την `ic`, η ECL^{i}PS^e παρέχει και τη βιβλιοθήκη `ic_global`, στην οποία ορίζονται κάποιοι νέοι περιορισμοί και επαναορίζονται, με διαφορετικό τρόπο, κάποιοι περιορισμοί που είναι υλοποιημένοι στην `ic`. Μεταξύ αυτών των περιορισμών είναι και ο `alldifferent/1`. Αν έχουμε ταυτόχρονα φορτωμένες και τις δύο βιβλιοθήκες, μπορούμε να αναφερθούμε στις διαφορετικής υλοποιήσεις του περιορισμού αυτού βάζοντας μπροστά από το όνομά του το πρόθεμα "`ic:`" ή "`ic_global:`". Δείτε την παρακάτω αλληλεπίδραση με την ECL^{i}PS^e:

```
?- lib(ic), lib(ic_global).
Yes (0.16s cpu)
?- [X,Y,Z] :: [0,1], ic:alldifferent([X,Y,Z]).
X = X{[0, 1]}
Y = Y{[0, 1]}
Z = Z{[0, 1]}
There are 3 delayed goals.
Yes (0.00s cpu)
?- [X,Y,Z] :: [0,1], ic_global:alldifferent([X,Y,Z]).
No (0.00s cpu)
```

Ποια συμπεριφορά του `alldifferent/1` πιστεύετε είναι ορθότερη, αυτή της `ic` ή της `ic_global`; Είναι ο ορισμός που δώσατε για τον `ic:alldifferent/1` συμβατός με την παραπάνω συμπεριφορά; Προτείνετε κάποιες ιδέες για το πώς θα μπορούσε να είναι υλοποιημένος, όχι κατ΄ ανάγκη σε ECL^{i}PS^e, ο `ic_global:alldifferent/1`, ώστε να συμπεριφέρεται όπως προηγουμένως.

(γ΄) Πώς πιστεύετε ότι είναι εύλογο και αποδοτικό να υλοποιείται η αποκοπή (κατηγόρημα `!/0`) σε συστήματα Prolog που είναι διερμηνείς;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2007

1. (α΄) Η Prolog χρησιμοποιεί, ως γνωστόν, μία *πρώτα-κατά-βάθος* προσέγγιση για την εύρεση των απαντήσεων στις υποβαλλόμενες ερωτήσεις. Πιστεύετε ότι θα μπορούσαμε να είχαμε μία γλώσσα προγραμματισμού που να βασίζεται στη λογική, όπως και η Prolog, η οποία να εφαρμόζει τη διαδικασία της ανάλυσης με έναν *πρώτα-κατά-πλάτος* τρόπο, δηλαδή με διάσχιση του δέντρου ανάλυσης κατά επίπεδα; Αν ναι, δώστε ένα απλό παράδειγμα προγράμματος αυτής της γλώσσας και δείξτε πώς θα υπολογιζόταν η απάντηση σε μία συγκεκριμένη ερώτηση με την *πρώτα-κατά-πλάτος* προσέγγιση. Για λόγους απλότητας, χρησιμοποιήστε στο πρόγραμμα αυτό κατηγορήματα βαθμού 0, ώστε να μην εμπλακείτε και με τη διαδικασία της ενοποίησης. Αναφέρατε ένα μειονέκτημα και ένα πλεονέκτημα της χρήσης μίας *πρώτα-κατά-πλάτος* διαδικασίας ανάλυσης, σε σχέση με την εφαρμοζόμενη στην Prolog *πρώτα-κατά-βάθος* διαδικασία.

   (β΄) Ένα τουριστικό γραφείο στη Βενετία έχει προσλάβει έναν γονδολιέρη για να κάνει βόλτες στα κανάλια με τη γόνδολά του διάφορους πελάτες του γραφείου. Το γραφείο εφοδιάζει τους τουρίστες που έχουν πληρώσει για τη βόλτα με ένα αναγνωριστικό, το οποίο, όταν το φορούν, αποτελεί την απόδειξη για τον γονδολιέρη ότι το συγκεκριμένο άτομο δικαιούται τη βόλτα. Κάθε άτομο δικαιούται να κάνει μία φορά μόνο τη βόλτα. Επίσης, ποτέ στη βόλτα δεν υπάρχει μόνο ένα άτομο, εκτός του γονδολιέρη, φυσικά. Στο τέλος της ημέρας ο γονδολιέρης, έχοντας μετρήσει πόσα άτομα έκανε βόλτα, πηγαίνει στο τουριστικό γραφείο για να πληρωθεί ανάλογα με το πλήθος των ατόμων.

   Μία συγκεκριμένη ημέρα όμως, ενώ το γραφείο είχε προμηθεύσει N αναγνωριστικά σε ισάριθμους πελάτες του, ο γονδολιέρης ζήτησε να πληρωθεί για $N + 1$ άτομα. Κάποιος κορόιδεψε, λοιπόν. Οι πιθανές περιπτώσεις είναι είτε κάποιο άτομο να πήγε δύο φορές τη βόλτα, είτε ο γονδολιέρης να είπε ψέματα. Το τουριστικό γραφείο αποφασίζει, για να διαλευκάνει την απάτη, να ρωτήσει τους πελάτες του, ποιους άλλους πελάτες είδαν στη βόλτα που έκαναν. Θεωρήστε ότι όλοι απαντούν έντιμα και ότι στην περίπτωση που κάποιος έκανε δύο βόλτες, στην απάντησή του δίνει όλους αυτούς που συνάντησε και στις δύο βόλτες.

   Έστω ότι οι απαντήσεις των πελατών είναι καταχωρημένες σε ένα σύνολο γεγονότων Prolog με κατηγόρημα `saw/2`, όπου το `saw(X,Y)` σημαίνει ότι ο `X` δήλωσε ότι είδε τον `Y` και ότι ο `Y` δήλωσε ότι είδε τον `X` στις βόλτες τους. Ορίστε σε Prolog ένα κατηγόρημα `cheater/1`, το οποίο όταν καλείται σαν `cheater(X)` να επιστρέφει στη μεταβλητή `X` είτε τον απατεώνα-τουρίστα είτε τον γονδολιέρη (`gondola_driver`).

   Δύο παραδείγματα εκτέλεσης φαίνονται στη συνέχεια.

Αν έχουμε

```
saw(george,john).
saw(george,jim).
saw(john,jack).
saw(jack,george).
```

τότε

```
?- cheater(X).
X = george
?-
```

Αν έχουμε

```
saw(anna,beata).
saw(anna,christa).
saw(beata,christa).
saw(donna,eva).
```

τότε

```
?- cheater(X).
X = gondola_driver
?-
```

2. Θεωρήστε ότι ένας πίνακας μπορεί να αναπαρασταθεί στην Prolog σαν μία λίστα από τις γραμμές του, όπου κάθε γραμμή είναι μία λίστα από τα στοιχεία της. Έχοντας αυτό σαν δεδομένο, ορίστε το κατηγόρημα `matr_det/2` έτσι ώστε το `matr_det(M,D)` να επιστρέφει στο `D` την ορίζουσα του πίνακα `M`. Για παράδειγμα:

```
?- matr_det([[5,3],
             [8,7]],D).
   D = 11
?- matr_det([[4,3,2,1],
             [3,2,1,2],
             [1,4,1,3],
             [2,1,3,2]],D).
   D = 51
?- matr_det([[2,0,0,0,0,0],
             [0,3,0,0,0,0],
             [0,0,4,0,0,0],
             [0,0,0,5,0,0],
             [0,0,0,0,6,0],
             [0,0,0,0,0,7]],D).
   D = 5040
```

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(a,f(b)).
p(X,f(f(Y))) :- p(Y,f(X)).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Μπορεί το `p(a,b)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε ένα τέτοιο μοντέλο. Ομοίως για το `p(b,f(a))`. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος; Πώς το βρήκατε;

(β′) Έστω ότι θέλουμε να αγοράσουμε από ένα κατάστημα N συγκεκριμένα προϊόντα p_i, $i = 1, 2, \ldots, N$. Το κατάστημα προσφέρει τα προϊόντα αυτά τόσο μεμονωμένα, φυσικά σε κάποια τιμή το καθένα, αλλά και σαν μέρος προσφορών που είναι σύνολα προϊόντων. Έστω ότι το κατάστημα έχει M προσφορές. Κάθε προσφορά o_j, $j = 1, 2, \ldots, M$, ενδέχεται να περιλαμβάνει και προϊόντα που μας ενδιαφέρουν, και, φυσικά, έχει την τιμή της c_j. Για λόγους ενιαίας αντιμετώπισης, θεωρήστε ότι τα μεμονωμένα προϊόντα είναι ισοδύναμα με προσφορές, που αποτελούνται από ένα μόνο προϊόν. Το πρόβλημα που τίθεται είναι ποιες προσφορές να αγοράσουμε ώστε να αποκτήσουμε κάθε προϊόν p_i που μας ενδιαφέρει τουλάχιστον μία φορά, αλλά με το ελάχιστο δυνατό συνολικό κόστος. Μοντελοποιήστε το πρόβλημα αυτό σαν πρόβλημα ικανοποίησης περιορισμών (μεταβλητές, πεδία, περιορισμοί) και σκιαγραφήστε πώς θα το αντιμετωπίζατε σε ένα σύστημα λογικού προγραμματισμού με περιορισμούς, για παράδειγμα την ECL^{i}PS^e.

(γ′) Σε τι διαφέρει ένας διερμηνέας της Prolog από ένα μεταγλωττιστή της Prolog, σε τι ένας διερμηνέας της Basic από ένα διερμηνέα της Prolog και σε τι ένας μεταγλωττιστής της C από ένα μεταγλωττιστή της Prolog;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2006

1. (α΄) Γνωρίζετε ότι τη γνώση "αν ισχύουν το `a` και το `b`, τότε θα ισχύει το `c`" τη διατυπώνουμε σε Prolog ως εξής:

   ```
   c :- a, b.
   ```

   Διατυπώστε, ή σχολιάστε τη δυνατότητα διατύπωσής τους, σε Prolog των παρακάτω δηλώσεων. Αν κάπου υπάρχει δυνατότητα διατύπωσης με περισσότερους από έναν τρόπους, συντακτικά διαφορετικούς, αλλά επί της ουσίας ισοδύναμους, επισημάνετέ το.

   i. αν ισχύει το `c`, τότε θα ισχύουν το `a` και το `b`
   ii. αν ισχύει το `a` ή το `b`, τότε θα ισχύει το `c`
   iii. αν ισχύει το `c`, τότε θα ισχύει το `a` ή το `b`

   (β΄) Ορίστε σε Prolog ένα κατηγόρημα `binnumbs/2`, που όταν καλείται σαν `binnumbs(N, L)`, με δεδομένο θετικό ακέραιο `N`, να επιστρέφει στη μεταβλητή `L` ένα `N`-ψήφιο δυαδικό αριθμό, σε μορφή λίστας των δυαδικών ψηφίων του. Για την ακρίβεια, το κατηγόρημα αυτό θα πρέπει να επιστρέφει μέσω οπισθοδρόμησης όλους τους `N`-ψήφιους δυαδικούς αριθμούς, και μάλιστα κατ΄ αύξουσα σειρά μεγέθους. Ένα παράδειγμα φαίνεται στο Σχήμα 1 παρακάτω. Ποια ερώτηση, στην οποία να εμπλέκεται το κατηγόρημα `binnumbs/2`, θα έπρεπε να δοθεί στη θέση των αποσιωπητικών του Σχήματος 2 για να πάρουμε το αποτέλεσμα που φαίνεται;

```
?- binnumbs(4, L).
L = [0,0,0,0]  -> ;
L = [0,0,0,1]  -> ;
L = [0,0,1,0]  -> ;
L = [0,0,1,1]  -> ;
L = [0,1,0,0]  -> ;
.............................
L = [1,1,0,1]  -> ;
L = [1,1,1,0]  -> ;
L = [1,1,1,1]
```

Σχήμα 1

```
?- ........................
[0,0,0]
[0,0,1]
[0,1,0]
[0,1,1]
[1,0,0]
[1,0,1]
[1,1,0]
[1,1,1]
no
```

Σχήμα 2

2. Ο ήρωας της μικρής μας ιστορίας διηγείται: "*Παντρεύτηκα μία χήρα που είχε μία κόρη. Ο πατέρας μου μας επισκεπτόταν συχνά, γνωρίστηκε με την κόρη της γυναίκας μου, και παντρεύτηκαν. Μήπως είμαι παππούς του εαυτού μου;*" Γράψτε πρόγραμμα Prolog και διατυπώστε, επίσης σε Prolog, την κατάλληλη ερώτηση που θα απαντά στο θεμελιώδες υπαρξιακό ερώτημα του φίλου μας. Στο πρόγραμμά σας να διατυπώσετε τη βασική γνώση του κόσμου που δόθηκε προηγουμένως, καθώς και κοινή γενική γνώση που ισχύει στην καθημερινή ζωή. Σ΄ αυτή την κοινή γνώση, περιλάβετε και ότι το παιδί της/του συζύγου κάποι-ου/-ας είναι και παιδί του/της ίδι-ου/-ας (παρότι μπορεί και να μην είναι βιολογικό του/της παιδί). Αφού αριθμήσετε τις προτάσεις Prolog που γράψατε στο πρόγραμμά σας, δώστε την ακολουθία των αριθμών προτάσεων που θα χρησιμοποιούντο από ένα σύστημα Prolog για να βρει την απάντηση στην ερώτηση που διατυπώσατε. Διευκρινίστε αν στην ακολουθία αυτή περιλαμβάνετε και τους αριθμούς προτάσεων των οποίων η χρησιμοποίηση κατά την εύρεση λύσης αναιρείται λόγω οπισθοδρόμησης, οπότε σημειώστε ποιοι είναι αυτοί οι αριθμοί, ή όχι.

3. (α′) i. Πότε η βάση Herbrand που αντιστοιχεί σ′ ένα λογικό πρόγραμμα είναι απειροσύνολο; Δώστε δύο πολύ απλά παραδείγματα οριστικών προγραμμάτων με βάσεις Herbrand που είναι απειροσύνολα και στο μεν ένα το ελάχιστο μοντέλο να είναι πεπερασμένο, στο δε άλλο απειροσύνολο.

ii. Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Τι επίπτωση έχει αυτό το γεγονός;

iii. Γιατί κατά τη γνώμη σας ενώ η έννοια του μοντέλου ορίζεται για οποιοδήποτε λογικό πρόγραμμα, αυτή του ελάχιστου μοντέλου ορίζεται μόνο για οριστικά προγράμματα;

(β′) Στα συστήματα προγραμματισμού με περιορισμούς επάνω σε πεπερασμένα πεδία (σύνολα), μερικές φορές, για κάθε μεταβλητή πεδίου φυλάσσεται (από το σύστημα) αναλυτικά το πεδίο της, σαν ένα σύνολο στοιχείων, και ανάλογα με τους εμπλεκόμενους περιορισμούς, διαγράφονται κάποια στοιχεία που δεν είναι δυνατόν να συμμετέχουν σε λύση. Άλλες φορές, για λόγους οικονομίας χώρου και απλούστευσης της υλοποίησης, για κάθε μεταβλητή πεδίου V (με πιθανές τιμές που είναι ακέραιοι αριθμοί), φυλάσσονται μόνο η ελάχιστη D_V^{min} και η μέγιστη D_V^{max} τιμή του πεδίου της D_V και όχι αναλυτικά όλο το πεδίο.

Έστω ότι έχουμε ένα πρόβλημα ικανοποίησης περιορισμών στο οποίο συμμετέχουν δύο μεταβλητές (πεπερασμένων πεδίων ακεραίων αριθμών) X και Y, με πεδία D_X και D_Y, αντίστοιχα, το οποίο το τροφοδοτούμε σ′ ένα σύστημα προγραμματισμού με περιορισμούς που ακολουθεί για την αναπαράσταση των πεδίων τη δεύτερη από τις πρακτικές που περιγράφηκαν προηγουμένως. Δηλαδή, για τις μεταβλητές X και Y, το σύστημα φυλάσσει τις ελάχιστες (D_X^{min}, D_Y^{min}) και μέγιστες (D_X^{max}, D_Y^{max}) τιμές των πεδίων τους μόνο. Θα πρέπει να μεταβληθούν από το σύστημα κάποιες από τις τιμές αυτές και πώς, αν μεταξύ των μεταβλητών X και Y δηλωθεί ο περιορισμός $a \cdot X + b \cdot Y \leq c$, όπου a, b και c είναι θετικοί ακέραιοι; Εφαρμόστε τα συμπεράσματά σας για την περίπτωση $D_X = D_Y = \{0, 1, 2, 3, 4, 5, 6\}$ και $a = 2$, $b = 3$, $c = 11$.

(γ′) Σε μία γλώσσα λογικού προγραμματισμού που υποστηρίζει Ή-παραλληλία, με την οδηγία "`:- parallel p/1.`" δηλώνεται ότι οι προτάσεις που ορίζουν το κατηγόρημα `p/1` μπορούν να εξετασθούν παράλληλα για την ικανοποίηση ενός στόχου με αυτό το κατηγόρημα. Αν το κατηγόρημα `p/1` ορίζεται μέσω των δύο γεγονότων "`p(a).`" και "`p(b).`" και κατά τη διάρκεια ικανοποίησης ενός στόχου προκύψει ο υποστόχος `p(X)`, πιστεύετε ότι έχει κάποια πρακτική αξία να βρεθούν οι τιμές του `X` (δηλαδή `X=a` και `X=b`) παράλληλα; Εξηγήστε την άποψή σας.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2005

1. (α΄) Κάποιος ήθελε να ορίσει ένα κατηγόρημα `add_nth`, για να προσθέτει ένα στοιχείο στην `n`-οστή θέση μίας λίστας και το έγραψε έτσι:

```
add_nth(1, X, [X]).
add_nth(N, X, [Y|L]) :-
   N is N-1,
   add_nth(N, X, L).
```

Φιλοδοξούσε, δηλαδή, στην ερώτηση

```
?- L = [a,b,c,d,e], add_nth(3, new, L), write(L), nl, fail.
```

να πάρει σαν απάντηση την `[a,b,new,c,d,e]`. Ικανοποιήθηκε η φιλοδοξία του; Αν όχι, τι απάντηση πήρε; Ποια λάθη έκανε στο πρόγραμμα που έγραψε; Είναι πιθανόν κάποια απ΄ αυτά να οφείλονται στην εμπειρία του από το διαδικαστικό προγραμματισμό; Εξηγήστε την άποψη σας, κάνοντας και ένα γενικότερο σχόλιο για την Prolog σε σχέση με τις γλώσσες διαδικαστικού προγραμματισμού. Τέλος, γράψτε και τη σωστή εκδοχή του κατηγορήματος `add_nth`, ώστε αυτό να συμπεριφέρεται με τον επιθυμητό τρόπο.

(β΄) Ορίστε σε Prolog το κατηγόρημα `subst_one/4`, με τέτοιο τρόπο ώστε όταν αυτό καλείται σαν `subst_one(X, L1, Y, L2)`, να επιστρέφει στο `L2` τη λίστα που προκύπτει αν αντικατασταθεί μία εμφάνιση του στοιχείου `X` στη λίστα `L1` με το στοιχείο `Y`. Φυσικά, αν υπάρχουν πολλές εμφανίσεις του στοιχείου `X` στη λίστα `L1`, το `subst_one/4` να επιστρέφει μέσω οπισθοδρόμησης όλες τις πιθανές λύσεις, όπου σε κάθε μία έχει γίνει μία δυνατή αντικατάσταση. Επίσης, ορίστε και το κατηγόρημα `subst_all/4`, έτσι ώστε όταν αυτό καλείται με τον ίδιο τρόπο, να επιστρέφει στο `L2` τη λίστα που προκύπτει αν αντικατασταθούν όλες οι εμφανίσεις του στοιχείου `X` στη λίστα `L1` με το στοιχείο `Y`. Παραδείγματα ερωτήσεων:

```
?- subst_one(b, [a,b,c,b,b,d,e], w, L).
L = [a,w,c,b,b,d,e]  -> ;
L = [a,b,c,w,b,d,e]  -> ;
L = [a,b,c,b,w,d,e]
no
?- subst_all(b, [a,b,c,b,b,d,e], w, L).
L = [a,w,c,w,w,d,e]
```

Πώς θα έπρεπε, κατά τη γνώμη σας, να συμπεριφέρονται τα κατηγορήματα `subst_one/4` και `subst_all/4` όταν το στοιχείο `X` δεν εμφανίζεται στη λίστα `L1`; Υπάρχει αυτή η επιθυμητή συμπεριφορά στους ορισμούς που δώσατε;

2. (α΄) Με το ενσωματωμένο κατηγόρημα `current_op` μπορούμε να πάρουμε πληροφορίες για τους τελεστές που έχουν ορισθεί, τις προτεραιότητες και τους τύπους τους. Για παράδειγμα, στην ερώτηση

```
?- current_op(P1,yfx,+), current_op(P2,yfx,*), current_op(P3,xfx,mod).
```

παίρνουμε την απάντηση `P1 = 500`, `P2 = 400` και `P3 = 300`. Σημειώνεται ότι το `mod` είναι ο τελεστής για το υπόλοιπο διαίρεσης. Πώς θα έπρεπε να ορισθεί το συναρτησιακό σύμβολο ^ βαθμού 2 σαν τελεστής σ΄ ένα σύστημα Prolog, αν δεν είναι ήδη ορισμένο,

έτσι ώστε να παριστάνουμε μέσω αυτού την ύψωση σε δύναμη, λαμβάνοντας υπόψη ότι θα θέλαμε ο όρος `x + y ^ z mod u * w` να αντιπροσωπεύει την έκφραση $x+(y^z \ mod \ u)\times w$; Α, και το 2^{2^3} ισούται με 256, όχι με 64.

(β΄) Να ορίσετε σε Prolog ένα κατηγόρημα `reduce/2`, το οποίο να δέχεται στο πρώτο όρισμά του μία αλγεβρική έκφραση που είναι γινόμενο θετικών ακεραίων αριθμών και μονωνύμων, δηλαδή, μεταβλητών που πιθανώς έχουν υψωθεί και σε κάποια θετική ακέραια δύναμη, οι οποίες μεταβλητές παριστάνονται σαν σταθερές (άτομα) Prolog. Το κατηγόρημα αυτό να απλοποιεί την έκφραση που του δόθηκε, κάνοντας όλες τις απαραίτητες αναγωγές, επιστρέφοντας το αποτέλεσμα στο δεύτερο όρισμα. Για παράδειγμα:

```
?- reduce(3 * x * y ^ 2 * z * 5 * y ^ 3 * x ^ 5 * 2, E).
E = 30 * x ^ 6 * y ^ 5 * z
```

3. (α΄) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το λογικό πρόγραμμα του διπλανού σχήματος; Γιατί η ερμηνεία $I = \{$`p(a,a)`, `p(a,b)`, `p(c,b)`, `q(a,a)`, `q(b,c)`, `q(c,c)`$\}$ <u>δεν</u> είναι μοντέλο για το πρόγραμμα; Είναι δυνατόν το `p(b,a)` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε ένα τέτοιο μοντέλο. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος; Με ποιο τρόπο το κατασκευάσατε;

```
p(Y,X) :- q(X,Y).
q(X,X) :- p(X,Y).
p(a,b).
q(b,c).
```

(β΄) Σ΄ ένα πανεπιστημιακό τμήμα, κατά τη διάρκεια ενός ακαδημαϊκού εξαμήνου προσφέρθηκε ένας αριθμός μαθημάτων τα οποία πρόκειται να εξετασθούν στο τέλος του εξαμήνου. Έστω ότι υπάρχουν N_1, N_2, N_3 και N_4 μαθήματα του 1ου, 2ου, 3ου και 4ου έτους, αντίστοιχα, τα οποία πρόκειται να εξετασθούν σε D συνεχόμενες ημέρες κατά την εξεταστική περίοδο. Θεωρήστε ότι κάθε ημέρα υπάρχει η δυνατότητα εξέτασης 3 το πολύ μαθημάτων (ισχύει όμως $N_1 + N_2 + N_3 + N_4 \leq 3D$) και ότι δεν πρέπει να εξετασθούν την ίδια ημέρα δύο μαθήματα του ίδιου έτους. Θα ήταν καλή ιδέα να χρησιμοποιούσε ο υπεύθυνος για την κατάρτιση του προγράμματος εξετάσεων λογικό προγραμματισμό με περιορισμούς για να λύσει το πρόβλημα; Αν ναι, εξηγήστε το λόγο και βοηθήστε τον σκιαγραφώντας τον τρόπο με τον οποίο θα πρέπει να αντιμετωπισθεί το πρόβλημα σε μία γλώσσα λογικού προγραμματισμού με περιορισμούς (π.χ. την ECL^{i}PS^e), δίνοντας τα βασικά σημεία του σχετικού προγράμματος, χωρίς, κατ΄ ανάγκη, να ασχοληθείτε με τεχνικές λεπτομέρειες. Αρκεί να προτείνετε μία μοντελοποίηση του προβλήματος σαν ένα πρόβλημα ικανοποίησης περιορισμών και να δώσετε την αλληλουχία των ενεργειών που πρέπει να υλοποιηθούν στο σχετικό πρόγραμμα που επιλύει το πρόβλημα. Αν ενδιέφερε να βρεθεί όχι απλώς μία λύση, αλλά η καλύτερη, πώς θα ορίζατε το "καλύτερη" και πώς θα την βρίσκατε δουλεύοντας σ΄ ένα περιβάλλον λογικού προγραμματισμού με περιορισμούς;

(γ΄) Γνωρίζετε ότι ένας διερμηνέας Prolog χρησιμοποιεί μία στοίβα για να διαχειριστεί το δέντρο ανάλυσης που κατασκευάζεται κατά τη διαδικασία υπολογισμού της απάντησης σε μία ερώτηση. Ένας από τους βασικούς λόγους ύπαρξης αυτής της στοίβας είναι και η υποστήριξη της οπισθοδρόμησης. Δώστε ένα πολύ απλό πρόγραμμα Prolog, ίσως το απλούστερο δυνατό, καθώς και μία ερώτηση και δείξτε κάποια στιγμιότυπα της στοίβας, όπως αυτή διαμορφώνεται κατά τον υπολογισμό της απάντησης στην ερώτηση. Τι "παίζει" με την αποκοπή (cut – `!`) και τη στοίβα αυτή;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2005

1. (α΄) Προφανώς έχετε γράψει αρκετά προγράμματα Prolog στον υπολογιστή, είτε με αφορμή τις εργασίες του μαθήματος είτε για άλλο λόγο. Είναι απολύτως βέβαιο ότι τα προγράμματα αυτά δεν δούλευαν σωστά με την πρώτη δοκιμή τους. Συνεπώς χρειαζόντουσαν αποσφαλμάτωση (debugging). Πώς βρήκατε και διορθώσατε τα λάθη του τους; Τι άλλους τρόπους θα μπορούσατε να εφαρμόσετε για την αποσφαλμάτωση Prolog προγραμμάτων; Σχολιάστε τα πλεονεκτήματα και τα μειονεκτήματά τους.

   (β΄) Ορίστε σε Prolog το κατηγόρημα `mult_subst/4` έτσι ώστε όταν υποβάλλεται η ερώτηση `mult_subst(Is,Xs,Ls,Ys)`, να αντικαθιστά τα στοιχεία της λίστας `Xs` που βρίσκονται στις θέσεις που καθορίζονται από τους δείκτες που περιέχονται στη λίστα `Is` με τα στοιχεία που περιέχονται στις λίστες που ανήκουν στη λίστα `Ls` (τα στοιχεία-δείκτες της `Is` αντιστοιχούν ένα προς ένα με τα στοιχεία-λίστες της `Ls`). Το αποτέλεσμα να επιστρέφεται στη μεταβλητή `Ys`. Ένα παράδειγμα:

```
?- mult_subst([2,4,5,7], [a,b,c,d,e,f,g,h], [[k,l],[m],[],[n,o,p]],R).
R = [a,k,l,c,m,f,n,o,p,h]
?-
```

   Σημειώνεται ότι τα στοιχεία της λίστας `Xs`, καθώς και τα στοιχεία των λιστών που ανήκουν στη λίστα `Ls` μπορεί να είναι αυθαίρετοι όροι Prolog.

2. Ορίστε σε Prolog το κατηγόρημα `cycle/3`, το οποίο όταν καλείται δίνοντας στα δύο πρώτα ορίσματά του δύο θετικούς ακεραίους `M` και `N`, να επιστρέφει στο τρίτο όρισμά του μία λίστα από δεκαδικά ψηφία τα οποία αποτελούν την περίοδο του δεκαδικού αριθμού που προκύπτει από τη διαίρεση `M/N`. Κάποια παραδείγματα εκτέλεσης είναι τα εξής:

```
?- cycle(3,4,C).
C = [0]      %  3/4 = 0.75000...
?- cycle(4,3,C).
C = [3]      %  4/3 = 1.3333...
?- cycle(1,7,C).
C = [1,4,2,8,5,7]
             %  1/7 = 0.142857142857...
?- cycle(129,55,C).
C = [4,5]    %  129/55 = 2.3454545...
?- cycle(13,43,C).
C = [3,0,2,3,2,5,5,8,1,3,9,5,3,4,8,8,3,7,2,0,9]
             %  13/43 = 0.3023255813953488372093023 25...
?-
```

3. (α΄) i. Πότε η βάση Herbrand που αντιστοιχεί σ΄ ένα λογικό πρόγραμμα είναι απειροσύνολο; Δώστε δύο πολύ απλά παραδείγματα οριστικών προγραμμάτων με βάσεις Herbrand που είναι απειροσύνολα και στο μεν ένα το ελάχιστο μοντέλο να είναι πεπερασμένο, στο δε άλλο απειροσύνολο.

   ii. Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Τι επίπτωση έχει αυτό το γεγονός;

iii. Γιατί κατά τη γνώμη σας ενώ η έννοια του μοντέλου ορίζεται για οποιοδήποτε λογικό πρόγραμμα, αυτή του ελάχιστου μοντέλου ορίζεται μόνο για οριστικά προγράμματα;

(β′) Παρατηρήστε την παρακάτω ερώτηση που υποβάλλουμε στην ECL^{i}PS^e, αφού πρώτα έχουμε φορτώσει τη βιβλιοθήκη πεπερασμένων πεδίων `fd`, και την απάντηση που παίρνουμε:

```
?- lib(fd).
yes.
?- [X,Y] :: 1..10, X #> Y, Y #> X.
no.
?-
```

Υπενθυμίζεται ότι το κατηγόρημα `#>` εκφράζει τη σχέση του "μεγαλύτερου" σαν περιορισμό. Με ποιο τρόπο πιστεύετε ότι η ECL^{i}PS^e κατέληξε στην απάντηση "`no.`"; Είδε ότι προφανώς δεν μπορεί να είναι το `X` μεγαλύτερο του `Y` και ταυτόχρονα το `Y` μεγαλύτερο του `X`, ή κάπως αλλιώς;

(γ′) Γνωρίζετε ότι στη στοίβα που χρησιμοποιεί ένας διερμηνέας Prolog για να διαχειριστεί το δέντρο ανάλυσης που κατασκευάζεται κατά τη διαδικασία υπολογισμού της απάντησης σε μία ερώτηση, οι εγγραφές που περιλαμβάνονται αντιστοιχούν σ′ ένα κόμβο του δέντρου. Ο κόμβος αυτός κωδικοποιεί την ενοποίηση ενός στόχου με την κεφαλή μίας πρότασης του προγράμματος και στην αντίστοιχη εγγραφή της στοίβας δημιουργείται ένα περιβάλλον για τις μεταβλητές της πρότασης αυτής. Πιστεύετε ότι σε περίπτωση οπισθοδρόμησης αρκεί απλώς να μετακινηθεί ο δείκτης της στοίβας ώστε να ακυρωθεί η εγγραφή που αντιστοιχεί στον κόμβο του δέντρου πίσω από τον οποίο οπισθοδρομούμε, ή πρέπει να γίνουν και άλλες ενέργειες; Αν ναι, ποιες; Δώστε και ένα παράδειγμα προγράμματος που να επιδεικνύει τις απόψεις σας.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2004

1. (α΄) Έστω ότι για τους τελεστές `op1`, `op2`, `op3`, `op4` και `op5` έχουν δηλωθεί τα παρακάτω:

```
:- op(120, fy,  op1).
:- op(150, yf,  op2).
:- op(250, xfx, op3).
:- op(200, fx,  op4).
:- op(100, xfy, op5).
```

Δώστε τον όρο Prolog, στην κλασική γραφή, με τον οποίο αντιστοιχεί η έκφραση:

```
op1 x op2 op3 op4 y op5 z op5 w
```

(β΄) Το πρόβλημα της κατάστρωσης σχεδίου (planning) στην Τεχνητή Νοημοσύνη συνίσταται στην εύρεση μίας ακολουθίας ή ενός συνόλου από ενέργειες, οι οποίες, αν εκτελεσθούν, οδηγούν στην επίτευξη ενός στόχου. Υπάρχουν μεθοδολογίες γραμμικής κατάστρωσης σχεδίου (linear planning), στις οποίες το αποτέλεσμα είναι μία συγκεκριμένη ακολουθία από ενέργειες, δηλαδή ένα γραμμικό πλάνο (linear plan), που πρέπει να εκτελεσθούν με τη δεδομένη σειρά για να επιτευχθεί ο επιθυμητός στόχος. Υπάρχουν όμως και μεθοδολογίες μη-γραμμικής κατάστρωσης σχεδίου (non-linear planning), στις οποίες το αποτέλεσμα είναι ένα σύνολο από ζευγάρια διατεταγμένων ενεργειών, δηλαδή ένα μη-γραμμικό πλάνο (non-linear plan), όπου η απαίτηση, για την επίτευξη του στόχου, είναι η πρώτη ενέργεια κάθε ζευγαριού να πρέπει να εκτελεσθεί πριν από τη δεύτερη ενέργεια. Ένα μη-γραμμικό πλάνο δεν ορίζει κάποια συγκεκριμένη ακολουθία εκτέλεσης των ενεργειών, απλώς περιγράφει εμμέσως όλες τις ακολουθίες που σέβονται τις επί μέρους διατάξεις ενεργειών που περιέχονται στο πλάνο.

Σας ζητείται να υλοποιήστε σε Prolog ένα κατηγόρημα `linearize/2`, το οποίο, όταν καλείται σαν `linearize(NonLinearPlan, LinearPlan)`, με δεδομένο ένα μη-γραμμικό πλάνο `NonLinearPlan` που αναπαρίσταται σαν μία λίστα από διατάξεις ενεργειών, για παράδειγμα όρους της μορφής `bef(Act1, Act2)`, όπου `Act1` και `Act2` είναι ενέργειες, να κατασκευάζει μέσω οπισθοδρόμησης όλα τα δυνατά γραμμικά πλάνα που αντιστοιχούν στο δοθέν μη-γραμμικό πλάνο. Υποθέστε ότι δεν υπάρχουν ενέργειες που δεν συνδέονται με τουλάχιστον άλλη μία ενέργεια με κάποια συγκεκριμένη διάταξη στο δοθέν μη-γραμμικό πλάνο. Παραδείγματα εκτέλεσης του `linearize/2` είναι τα εξής:

```
?- linearize([bef(a,b), bef(b,c), bef(c,d),
              bef(a,e), bef(e,d), bef(d,f)], LinearPlan).
LinearPlan = [a, e, b, c, d, f]   -> ;
LinearPlan = [a, b, e, c, d, f]   -> ;
LinearPlan = [a, b, c, e, d, f]   -> ;
no
?- linearize([bef(a,b), bef(b,c), bef(c,a)], LinearPlan).
no
```

Δεδομένου ότι σε πραγματικά προβλήματα το πλήθος των ενεργειών ενός πλάνου μπορεί να είναι ιδιαίτερα μεγάλο, θα ληφθεί σοβαρά υπόψη αν η υλοποίηση που θα προτείνετε για το `linearize/2` δεν έχει εκθετική πολυπλοκότητα ως προς το πλήθος των ενεργειών.

2. Μία ακολουθία Lanford είναι μία 27-μελής διάταξη των αριθμών 1, 2, ..., 9 (από τρεις φορές ο καθένας), στην οποία για κάθε i ($i = 1, 2, \ldots, 9$), μεταξύ της πρώτης και της δεύτερης εμφάνισης

του i, όπως και μεταξύ της δεύτερης και τρίτης, να υπάρχουν, σε κάθε περίπτωση, ακριβώς i στο πλήθος άλλοι αριθμοί. Συμπληρώστε τον ελλιπή ορισμό του κατηγορήματος `lanford/1` που ακολουθεί, με το οποίο μέσω οπισθοδρόμησης μπορούμε να βρούμε όλες τις ακολουθίες Lanford.

```
lanford(L) :-
   L = ..........................................................,
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........),
   mypred(.........., .........., .........).
```

Ορίστε επίσης και το κατηγόρημα `mypred/3`. Η απάντησή σας να είναι συμβατή με το αποτέλεσμα:

```
?- lanford(L).
L = [1,9,1,6,1,8,2,5,7,2,6,9,2,5,8,4,7,6,3,5,4,9,3,8,7,4,3]   -> ;
L = [1,9,1,2,1,8,2,4,6,2,7,9,4,5,8,6,3,4,7,5,3,9,6,8,3,5,7]   -> ;
L = [1,8,1,9,1,5,2,6,7,2,8,5,2,9,6,4,7,5,3,8,4,6,3,9,7,4,3]   -> ;
L = [3,4,7,9,3,6,4,8,3,5,7,4,6,9,2,5,8,2,7,6,2,5,1,9,1,8,1]   -> ;
L = [7,5,3,8,6,9,3,5,7,4,3,6,8,5,4,9,7,2,6,4,2,8,1,2,1,9,1]   -> ;
L = [3,4,7,8,3,9,4,5,3,6,7,4,8,5,2,9,6,2,7,5,2,8,1,6,1,9,1]   -> ;
no
```

3. (α′) Έστω μία γλώσσα πρώτης τάξης στην οποία $p/1, q/1 \in P$, $a/0, b/0 \in F$ και $x \in V$. Δώστε μία οριστική πρόταση αυτής της γλώσσας και δύο μοντέλα της πρότασης που να είναι ξένα μεταξύ τους. Αν ένα οριστικό πρόγραμμα αποτελείται μόνο από αυτή την πρόταση, ποιο είναι το ελάχιστο μοντέλο του; Έχετε να κάνετε κάποιο ενδιαφέρον σχόλιο;

   (β′) Ένας κρυπτάριθμος είναι ένας γρίφος στον οποίο διατυπώνεται κάποια αριθμητική πράξη μεταξύ αριθμών, τα ψηφία των οποίων παριστάνονται με γράμματα του αλφαβήτου. Το ζητούμενο είναι να βρεθεί με ποια ψηφία από το 0 έως το 9 πρέπει να αντικατασταθούν τα γράμματα στην αριθμητική πράξη, έτσι ώστε οι διαφορετικές εμφανίσεις του ίδιου γράμματος να αντικατασταθούν με το ίδιο ψηφίο, διαφορετικά γράμματα να αντικατασταθούν με διαφορετικά ψηφία και η αριθμητική πράξη, μετά την αντικατάσταση των γραμμάτων με ψηφία, να είναι σωστή. Σκιαγραφήστε τον τρόπο με τον οποίο θα αντιμετωπίζατε την επίλυση ενός συγκεκριμένου κρυπτάριθμου, για παράδειγμα του `SEND + MORE = MONEY`, σε μία γλώσσα λογικού προγραμματισμού με περιορισμούς (π.χ. την $\mathrm{ECL^iPS^e}$), δίνοντας τα βασικά σημεία του σχετικού προγράμματος, χωρίς, κατ΄ ανάγκη, να υπεισέλθετε σε δευτερεύουσες τεχνικές λεπτομέρειες. Αρκεί να προτείνετε μία μοντελοποίηση του προβλήματος σαν ένα πρόβλημα ικανοποίησης περιορισμών και να δώσετε την αλληλουχία των ενεργειών που πρέπει να υλοποιηθούν στο σχετικό πρόγραμμα που επιλύει το πρόβλημα.

   (γ′) Σε τι διαφέρει ένας διερμηνέας της Prolog από ένα μεταγλωττιστή της Prolog, σε τι ένας διερμηνέας της Basic από ένα διερμηνέα της Prolog και σε τι ένας μεταγλωττιστής της C από ένα μεταγλωττιστή της Prolog;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2003

1. (α΄) Τι ακριβώς σημαίνει ότι η άρνηση στην Prolog ορίζεται *μέσω αποτυχίας* (negation as failure); Πώς σχετίζεται αυτό με την *υπόθεση του κλειστού κόσμου*;

   Θεωρήστε ότι έχετε στη διάθεσή σας ένα πρόγραμμα Prolog στο οποίο έχουν οριστεί γεγονότα με κατηγορήματα `man/1` και `woman/1`, μέσω των οποίων είναι καταχωρημένα ένα σύνολο από άνδρες και ένα σύνολο από γυναίκες. Για παράδειγμα:

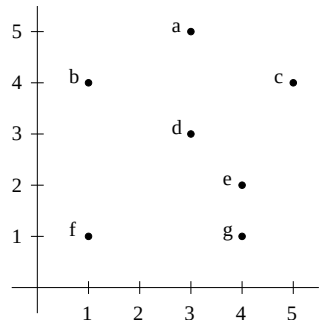
   ```
   man(john).   man(george).   woman(mary).   woman(jane).   woman(helen).
   ```

   Σας είναι γνωστό ότι αν θέλετε να υποβάλετε στο πρόγραμμα αυτό την ερώτηση "*υπάρχει κάποιος που είναι άνδρας;*", αυτό διατυπώνεται σε Prolog σαν "`?- man(X).`" (ομοίως και για τις γυναίκες). Δηλαδή, αν η ερώτηση που θέλατε να κάνετε ήταν "*υπάρχει κάποιος που δεν είναι άνδρας;*", αυτό θα το διατυπώνατε σε Prolog σαν "`?- not man(X).`" (ομοίως και για τις γυναίκες); Εξηγήστε την απάντησή σας, εφ΄ όσον είναι αρνητική, και προτείνετε μία σωστή αντιμετώπιση του προβλήματος της υποβολής ερωτήσεων με άρνηση.

   (β΄) Θεωρήστε ότι ένας πίνακας μπορεί να αναπαρασταθεί στην Prolog σαν μία λίστα από τις γραμμές του, όπου κάθε γραμμή είναι μία λίστα από τα στοιχεία της. Έχοντας αυτό σαν δεδομένο, ορίστε το κατηγόρημα `matr_transp/2` έτσι ώστε το `matr_transp(M1,M2)` να επιστρέφει στο `M2` τον ανάστροφο πίνακα (δηλαδή αυτόν που προκύπτει με εναλλαγή γραμμών και στηλών) του `M1`. Για παράδειγμα:

   ```
   ?- matr_transp([[5,8,9,7,2],[3,6,1,1,4],[2,4,2,8,0]],M).
      M = [[5,3,2],[8,6,4],[9,1,2],[7,1,8],[2,4,0]]
   ```

2. Θεωρήστε ότι έχετε ένα σύνολο από σημεία στο επίπεδο, των οποίων η αναπαράσταση σ΄ ένα πρόγραμμα Prolog γίνεται μέσω γεγονότων με κατηγόρημα `point/2`, όπου το πρώτο όρισμα είναι μία σταθερά (το όνομα του σημείου) και το δεύτερο είναι ένας όρος της μορφής `p(X,Y)`, όπου `X` και `Y` είναι οι συντεταγμένες του σημείου. Για παράδειγμα, το σημείο `a`, στο διπλανό σχήμα, παριστάνεται από το γεγονός "`point(a,p(3,5)).`", το `b` από το "`point(b,p(1,4)).`", το `c` από το "`point(c,p(5,4)).`", κ.ο.κ. Να ορίσετε σε Prolog δύο κατηγορήματα, έστω `max_dist/2` και `min_dist/2`, τα οποία να επιστρέφουν στα ορίσματά τους δύο σημεία, απ΄ αυτά που έχουν δοθεί, που απέχουν τη μεγαλύτερη (αντίστοιχα, μικρότερη) δυνατή απόσταση. Για παράδειγμα, με βάση τα σημεία του σχήματος, θα παίρναμε:




   ```
   ?- max_dist(X,Y).
      X = c
      Y = f
   ?- min_dist(X,Y).
      X = e
      Y = g
   ```

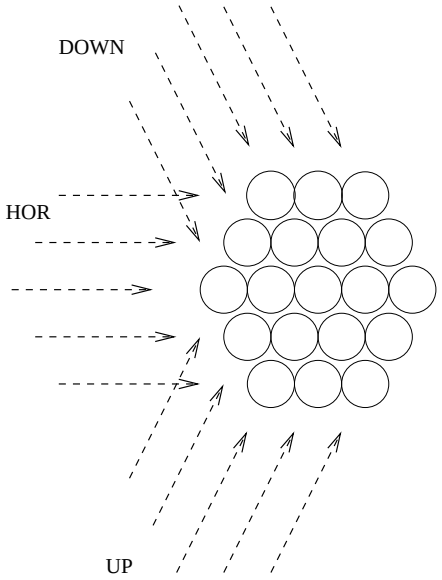
   Αν υπάρχουν περισσότερα του ενός ζευγάρια σημείων με μέγιστη (αντίστοιχα, ελάχιστη) απόσταση, το πρόγραμμά σας να τα βρίσκει όλα μέσω οπισθοδρόμησης. Φυσικά, δεν πρέπει να δίνει σαν εναλλακτικές λύσεις τα ίδια ζευγάρια σημείων, αλλά με διαφορετική διάταξη. Δηλαδή να μην δίνει σαν σημεία μέγιστης απόστασης τα `c`, `f`, αλλά και τα `f`, `c`.

3. (α′) Δίνεται το εξής λογικό πρόγραμμα:

```
p(a).
p(f(b)).
p(f(f(X))) :- p(X).
```

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε ένα τέτοιο μοντέλο. Ομοίως για το `p(f(f(f(b))))`. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος;

(β′) Έστω ότι θέλουμε να συμπληρώσουμε στους κύκλους του διπλανού σχήματος τους αριθμούς από το 1 έως το 19 έτσι ώστε στις πέντε γραμμές της οριζόντιας κατεύθυνσης (HOR), στις πέντε ανιούσες γραμμές (UP) και στις πέντε κατιούσες (DOWN) να έχουμε το ίδιο άθροισμα αριθμών (πόσο;). Σκιαγραφήστε τον τρόπο με τον οποίο θα αντιμετωπίζατε το πρόβλημα αυτό σε μία γλώσσα λογικού προγραμματισμού με περιορισμούς (π.χ. την ECL^{i}PS^e), δίνοντας τα βασικά σημεία του σχετικού προγράμματος, χωρίς, κατ΄ ανάγκη, να υπεισέλθετε σε δευτερεύουσες τεχνικές λεπτομέρειες. Αρκεί να προτείνετε μία μοντελοποίηση του προβλήματος σαν ένα πρόβλημα ικανοποίησης περιορισμών και να δώσετε την αλληλουχία των ενεργειών που πρέπει να υλοποιηθούν στο σχετικό πρόγραμμα που επιλύει το πρόβλημα. Υπάρχει η δυνατότητα να αποφύγετε την εύρεση συμμετρικών λύσεων, π.χ. αυτών που προκύπτουν με περιστροφές του εξαγώνου;



(γ′) Σε τι διαφέρει ένας διερμηνέας της Prolog από ένα μεταγλωττιστή της Prolog, σε τι ένας διερμηνέας της Basic από ένα διερμηνέα της Prolog και σε τι ένας μεταγλωττιστής της C από ένα μεταγλωττιστή της Prolog;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2003

1. (α΄) Ποια βασικά κριτήρια θα έπρεπε να λάβει υπόψη του κάποιος αν έπρεπε να επιλέξει αν θα υλοποιήσει ένα κατηγόρημα Prolog μέσω συσσωρευτή ή όχι, εφόσον είναι εφικτές και οι δύο δυνατότητες; Δώστε ένα συγκεκριμένο παράδειγμα υλοποίησης ενός κατηγορήματος Prolog και με τους δύο τρόπους και εξηγήστε κάτω από ποιες συνθήκες θα υιοθετούσατε τον ένα ή τον άλλο.

   (β΄) Ορίστε σε Prolog το κατηγόρημα `rem_ind/3` έτσι ώστε όταν υποβάλλεται η ερώτηση `rem_ind(Is,Xs,Rs)`, να επιστρέφει στη λίστα `Rs` τα στοιχεία που απομένουν όταν από τη λίστα `Xs` διαγραφούν εκείνα που βρίσκονται στις θέσεις που καθορίζονται από τους δείκτες που περιέχονται στη λίστα `Is`. Για παράδειγμα, η ερώτηση

   ```
   ?- rem_ind([1,3,4,6],[a,b,c,d,e,f,g],L).
   ```

   να επιστρέφει στο `L` τη λίστα `[b,e,g]`.

2. Δυαδικό λεξικό είναι ένα δυαδικό δέντρο του οποίου κάθε κόμβος είναι "μεγαλύτερος" από όλους τους κόμβους του αριστερού υποδέντρου του και "μικρότερος" από όλους τους κόμβους του δεξιού υποδέντρου του. Οι όροι "μεγαλύτερος" και "μικρότερος" αναφέρονται σε μία δεδομένη σχέση διάταξης. Υιοθετώντας μία κατάλληλη αναπαράσταση δυαδικών δέντρων (άρα και δυαδικών λεξικών), ορίστε σε Prolog ένα κατηγόρημα `dictionary/1` το οποίο να επιτυγχάνει όταν το όρισμα που του δίνεται είναι δυαδικό λεξικό. Φροντίστε η υλοποίηση του κατηγορήματος `dictionary/1` να είναι τέτοια ώστε η χρονική πολυπλοκότητά του να είναι γραμμική ως προς το πλήθος των κόμβων του δυαδικού δέντρου που του δίνεται σαν όρισμα.

3. (α΄) Δίνεται το εξής λογικό πρόγραμμα:

   ```
   p(a).
   p(f(b)).
   p(f(f(X))) :- p(X).
   ```

   Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Μπορεί το `p(f(a))` να ανήκει σε κάποιο μοντέλο του προγράμματος; Αν όχι, γιατί; Αν ναι, δώστε ένα τέτοιο μοντέλο. Ομοίως για το `p(f(f(f(b))))`. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος;

   (β΄) Έστω ότι σας δίνεται ένα σύνολο S από ακεραίους αριθμούς, ένας συγκεκριμένος ακέραιος N και σας ζητούν να βρείτε όλα τα υποσύνολα του S των οποίων τα στοιχεία έχουν άθροισμα N. Προτείνετε ένα μοντέλο (μεταβλητές, πεδία, περιορισμοί) για το πρόβλημα αυτό σαν ένα πρόβλημα ικανοποίησης περιορισμών και σκιαγραφήστε τον τρόπο με τον οποίο θα το υλοποιούσατε στη γλώσσα ECL^{i}PS^e.

   (γ΄) Γνωρίζετε ότι η μεταγλώττιση ενός προγράμματος Prolog συνίσταται στο μετασχηματισμό του σ΄ ένα πρόγραμμα χαμηλού επιπέδου, για τη λεγόμενη μηχανή WAM. Στη συνέχεια, η εκτέλεση του προγράμματος γίνεται με τη βοήθεια ενός εξομοιωτή της WAM. Γιατί, κατά τη γνώμη σας, δεν δουλεύουμε με την Prolog όπως και με τις περισσότερες κλασικές γλώσσες προγραμματισμού; Δηλαδή, γιατί δεν κατασκευάζουμε μεταγλωττιστές που να μεταφράζουν ένα πρόγραμμα Prolog απ΄ ευθείας στη γλώσσα μηχανής του μικροεπεξεργαστή μας, έτσι ώστε η εκτέλεσή του να γίνει παραδοσιακά από το υλικό (hardware); Θα είχε νόημα να έχουμε μεταγλωττιστές που μεταφράζουν προγράμματα C ή Java σε WAM;

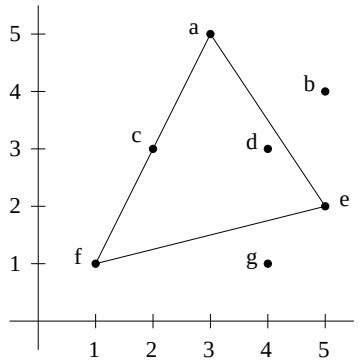
(δ΄) Κάποιος χρησιμοποιεί ένα σύστημα Prolog που υποστηρίζει Ή-παραλληλία και έστω ότι για ένα κατηγόρημα, στο οποίο υπάρχουν αποκοπές (`!`) στα σώματα ενός ή περισσοτέρων κανόνων στον ορισμό του, έχει δηλώσει (π.χ. μέσω μίας οδηγίας `parallel`) ότι επιθυμεί να εξετάζονται οι προτάσεις του ορισμού του παράλληλα. Πώς θα χαρακτηρίζατε τη δήλωσή του αυτή και γιατί;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2002

1. (α΄) Γνωρίζετε διάφορους τρόπους με τους οποίους μπορεί ένα πρόγραμμα γραμμένο σε κάποια διαδικαστική γλώσσα προγραμματισμού (π.χ. C, C++, Pascal, κ.λ.π.) να πάρει τα δεδομένα του, όπως: i) να είναι καταχωρημένα τα δεδομένα απ΄ ευθείας μέσα στο πρόγραμμα, για παράδειγμα μέσω αναθέσεων τιμών σε μεταβλητές με εντολές αντικατάστασης, ii) να δίνονται τα δεδομένα στη γραμμή εντολής όταν εκτελούμε το πρόγραμμα, ή iii) να διαβάζονται τα δεδομένα με κατάλληλες εντολές ή συναρτήσεις εισόδου κατά την εκτέλεση του προγράμματος. Πιστεύετε ότι αυτοί οι τρόποι εισαγωγής δεδομένων στα διαδικαστικά προγράμματα αντιστοιχούν σε κάποιους παρόμοιους τρόπους για τα προγράμματα Prolog και, αν ναι, ποιους; Σχολιάστε επίσης και το αν τα σχετικά πλεονεκτήματα και μειονεκτήματα των προηγούμενων τρόπων εισαγωγής δεδομένων σε διαδικαστικά προγράμματα ισχύουν και στους αντίστοιχους τρόπους, αν αυτοί υφίστανται, για τα προγράμματα Prolog.

   (β΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `strange/1` έτσι ώστε το `strange(L)` να επιστρέφει στο `L` μία λίστα με 10 στοιχεία, με την ιδιότητα να προκύπτει η ίδια λίστα είτε όταν προστεθούν στην αρχή της `L` τα στοιχεία `a`, `b`, `c` είτε όταν προστεθούν στο τέλος της `L` τα στοιχεία `b`, `c`, `a` με αυτήν ακριβώς τη σειρά. Ποια είναι η λίστα `L`;

2. Θεωρήστε ότι έχετε ένα σύνολο από σημεία στο επίπεδο, των οποίων η αναπαράσταση σ΄ ένα πρόγραμμα Prolog γίνεται μέσω γεγονότων με κατηγόρημα `point/2`, όπου το πρώτο όρισμα είναι μία σταθερά (το όνομα του σημείου) και το δεύτερο είναι ένας όρος της μορφής `p(X,Y)`, όπου `X` και `Y` είναι οι συντεταγμένες του σημείου. Για παράδειγμα, το σημείο `a`, στο διπλανό σχήμα, παριστάνεται από το γεγονός "`point(a,p(3,5)).`", το `b` από το "`point(b,p(5,4)).`"κ.ο.κ. Να ορίσετε σε Prolog ένα κατηγόρημα `max_triangle/2`, το οποίο όταν καλείται σαν `max_triangle(Tr,Ar)`, να επιστρέφει στο `Tr`, σαν έναν όρο `t(V1,V2,V3)`, το τρίγωνο μεγίστου εμβαδού (=`Ar`), με κορυφές τα σημεία με ονόματα `V1`, `V2` και `V3` απ΄ αυτά που έχουν δοθεί. Για παράδειγμα, με βάση τα σημεία του σχήματος, θα παίρναμε:




```
?- max_triangle(Tr,Ar).
   Tr = t(a,e,f)
   Ar = 7
```

   Αν υπάρχουν περισσότερα του ενός τρίγωνα μεγίστου εμβαδού, αρκεί το πρόγραμμά σας να βρίσκει ένα απ΄ αυτά. Φυσικά, δεν πρέπει να δίνει σαν εναλλακτικές λύσεις το ίδιο τρίγωνο, με διαφορετική διάταξη στις κορυφές του. Το εμβαδόν E ενός τριγώνου με μήκη πλευρών α, β και γ δίνεται από τον τύπο $E = \sqrt{\tau(\tau-\alpha)(\tau-\beta)(\tau-\gamma)}$, όπου $\tau = (\alpha+\beta+\gamma)/2$. Για τον υπολογισμό της τετραγωνικής ρίζας, χρησιμοποιήστε την ενσωματωμένη συνάρτηση `sqrt/1`. Σε ποιο σημείο στην κλίμακα από 0 έως 10 θα κατατάσσατε το πρόγραμμα που γράψατε, όπου στο σημείο 0 βρίσκεται το απόλυτο διαδικαστικό πρόγραμμα, ενώ στο σημείο 10 το απόλυτο δηλωτικό πρόγραμμα; Θα ήταν καλό η υλοποίησή σας να βρίσκεται πολύ κοντά προς το 10, αδιαφορώντας αν αυτό μπορεί να έχει επίπτωση στην αποδοτικότητα του προγράμματός σας.

3. (α΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `mm/4` έτσι ώστε όταν αυτό καλείται σαν `mm(L1, L2, CS, CD)`, όπου `L1` και `L2` είναι δύο δεδομένες ισομήκεις λίστες, να επιστρέφει στο `CS`

το πλήθος των κοινών στοιχείων των δύο λιστών που βρίσκονται στην ίδια θέση, και στο `CD` το πλήθος των κοινών στοιχείων των δύο λιστών, αφού εξαιρεθούν τα προηγούμενα, ανεξάρτητα από τη θέση στην οποία βρίσκονται. Για παράδειγμα:

```
?- mm([a,b,c,d,e],[f,b,g,d,c],CS,CD).
   CS = 2
   CD = 1
?- mm([a,b,b,c,d],[e,a,b,b,b],CS,CD).
   CS = 1
   CD = 2
```

Έχετε καμία ιδέα γιατί ονομάσαμε το συγκεκριμένο κατηγόρημα `mm`;

(β΄) Θεωρήστε ότι ένας πίνακας μπορεί να αναπαρασταθεί στην Prolog σαν μία λίστα από τις γραμμές του, όπου κάθε γραμμή είναι μία λίστα από τα στοιχεία της. Έχοντας αυτό σαν δεδομένο, ορίστε το κατηγόρημα `matr_mult/3` έτσι ώστε το `matr_mult(M1,M2,M3)` να επιστρέφει στο `M3` το γινόμενο των πινάκων `M1` και `M2`. Για παράδειγμα:

```
?- matr_mult([[1,2,3],[4,5,6],[6,5,4],[3,2,1]],
             [[3,4],[5,6],[7,8]],M).
   M = [[34,40],[79,94],[71,86],[26,32]]
```

4. (α΄) Δώστε ένα απλό παράδειγμα οριστικού προγράμματος, τη βάση Herbrand που αντιστοιχεί σ΄ αυτό και μία ερμηνεία I, η οποία όμως να μην είναι μοντέλο για το πρόγραμμα, αλλά τέτοια ώστε είτε αν διαγράψουμε το στοιχείο A_1 απ΄ αυτήν είτε αν της προσθέσουμε το στοιχείο A_2, οι ερμηνείες που προκύπτουν, δηλαδή οι $I_1 = I - \{A_1\}$ και $I_2 = I \cup \{A_2\}$ αντίστοιχα, να είναι μοντέλα του προγράμματος. Ποια είναι τα A_1 και A_2 για το παράδειγμά σας; Υπάρχει περίπτωση κάποια, ή και οι δύο, από τις ερμηνείες I_1 και I_2 να είναι το ελάχιστο μοντέλο του προγράμματος;

   (β΄) Από τα απλά μαθηματικά που γνωρίζουμε, είναι απολύτως βέβαιο ότι είναι αδύνατο να βρούμε τρεις αριθμούς `X`, `Y` και `Z`, τέτοιους ώστε να ισχύουν `X > Y`, `Y > Z` και `Z > X`. Αν τα `X`, `Y` και `Z` μπορούσαν να πάρουν ακέραιες τιμές στο διάστημα `[0,9]`, ένας τρόπος να επιβεβαιώσουμε την προηγούμενη προφανή γνώση υπολογιστικά θα ήταν να γράψουμε ένα πρόγραμμα που θα γεννά συστηματικά όλους τους δυνατούς συνδυασμούς τιμών για τα `X`, `Y` και `Z` ($10^3 = 1000$ στο πλήθος) και θα ελέγχει αν για κάποιον απ΄ αυτούς ισχύουν οι τρεις ανισότητες. Φυσικά, δεν θα μπορούσε να βρεθεί κάποιος τέτοιος συνδυασμός. Αν όμως υποβάλλαμε σ΄ ένα σύστημα λογικού προγραμματισμού με περιορισμούς, π.χ. στην ECL^{i}PS^e, την ερώτηση

   ```
   ?- [X,Y,Z] :: 0..9, X #> Y, Y #> Z, Z #> X.
   ```

   όπου το κατηγόρημα `#>` εκφράζει τη σχέση του "μεγαλύτερου" σαν περιορισμό, τι απάντηση πιστεύετε ότι θα παίρναμε και πώς θα υπολογιζόταν αυτή η απάντηση, σε γενικές γραμμές, από το σύστημα; Μήπως με εξαντλητική αναζήτηση ή κάπως αλλιώς;

   (γ΄) Αν σας ζητούσαν να υλοποιήσετε ένα μεταφραστή/μεταγλωττιστή της Prolog, ο οποίος να παράγει εκτελέσιμο πρόγραμμα για τον επεξεργαστή του συστήματος που χρησιμοποιείτε, όπως γίνεται για τις περισσότερες κλασικές γλώσσες προγραμματισμού, πώς θα αντιμετωπίζατε το πρόβλημα αυτό; Θα είχαν, κατά τη γνώμη σας, κάποια ιδιορρυθμία τα εκτελέσιμα προγράμματα που θα παρήγαγε ο μεταγλωττιστής σας, σε σχέση με αυτά που παράγουν οι μεταγλωττιστές στις κλασικές γλώσσες;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2002

1. (α΄) Έστω ότι έχετε στη διάθεσή σας ένα σύστημα Prolog το οποίο όμως δεν σας παρέχει καθόλου τη δυνατότητα αριθμητικής. Με άλλα λόγια, το σύστημα αυτό δεν διαθέτει το ενσωματωμένο κατηγόρημα `is/2`, ούτε κανένα άλλο ενσωματωμένο κατηγόρημα που προκαλεί έμμεσα ή άμεσα τον υπολογισμό τιμών αριθμητικών εκφράσεων, όπως τα `</2`, `>=/2`, κ.λ.π. Αυτό έχει σαν πρακτικό αποτέλεσμα, για παράδειγμα, να μην είναι δυνατόν να συσχετισθεί, απ΄ ευθείας από το σύστημα, ο σύνθετος όρος `5+3` με τον αριθμό `8`. Αν όμως πρέπει να χρησιμοποιήσετε, για κάποιο λόγο, το συγκεκριμένο σύστημα για να υλοποιήσετε μία εφαρμογή σε Prolog, στην οποία απαιτούνται και κάποιες στοιχειώδεις αριθμητικές δυνατότητες, όπως πρόσθεση, αφαίρεση ή άλλες πράξεις, μεταξύ σχετικά μικρών θετικών ακεραίων αριθμών, προτείνετε, σε γενικές γραμμές, δύο τουλάχιστον τρόπους μέσω των οποίων θα μπορούσατε να αντιμετωπίσετε το συγκεκριμένο πρόβλημα.

   (β΄) Να υλοποιηθούν σε Prolog τα κατηγορήματα `delete_one/3` και `delete_all/3` με τις εξής προδιαγραφές: Το `delete_one(X,L1,L2)` επιστρέφει στο `L2` τη λίστα που προκύπτει αν διαγραφεί η πρώτη εμφάνιση του στοιχείου `X` στη λίστα `L1`. Αν η `L1` δεν περιέχει το `X`, τότε η `L2` είναι η ίδια με την `L1`. Το `delete_all(X,L1,L2)` επιστρέφει στο `L2` τη λίστα που προκύπτει αν διαγραφούν όλες οι εμφανίσεις του στοιχείου `X` στη λίστα `L1`. Αν η `L1` δεν περιέχει το `X`, τότε η `L2` είναι η ίδια με την `L1`. Παραδείγματα:

```
?- delete_one(a,[b,c,d,e],L).
   L = [b,c,d,e]
?- delete_one(a,[b,c,a,d,a,e,a,a,f,g,a],L).
   L = [b,c,d,a,e,a,a,f,g,a]
?- delete_all(a,[b,c,d,e],L).
   L = [b,c,d,e]
?- delete_all(a,[b,c,a,d,a,e,a,a,f,g,a],L).
   L = [b,c,d,e,f,g]
```

2. Υλοποιήστε σε Prolog ένα κατηγόρημα `triangle/1` το οποίο όταν παίρνει σαν όρισμα μία λίστα από χαρακτήρες, να εκτυπώνει ένα σύνολο ισοπλεύρων τριγώνων, που οι πλευρές τους καθενός κατασκευάζονται από έναν από τους χαρακτήρες και το κάθε τρίγωνο περικλείει τα τρίγωνα που κατασκευάζονται με τους χαρακτήρες που ακολουθούν το δικό του χαρακτήρα στη λίστα εισόδου. Ο τελευταίος χαρακτήρας της λίστας συνθέτει το μικρότερο κεντρικό τρίγωνο που δεν είναι τίποτα άλλο από ένα εκφυλισμένο σημείο. Ένα παράδειγμα εκτέλεσης του προγράμματος είναι το εξής:

```
?- triangle([a,b,c,d]).
         a
        aba
       abcba
      abcdcba
     abccccba
    abbbbbbbba
   aaaaaaaaaaaaa
```

3. (α΄) Θεωρήστε το πρόγραμμα Prolog που ακολουθεί:

```
p(a(X),[b(X),c(X)]). p(a(X),[d(X)]). p(b(1),[]).  p(b(2),[]).
p(c(2),[]).  p(c(1),[]).  p(d(3),[]).

e1([]). e1([G|Gs]) :- p(G,B), append(B,Gs,NGs), e1(NGs). e2([]).
e2([G|Gs]) :- p(G,B), reverse(B,RB), append(Gs,RB,NGs), e2(NGs).
e3([]). e3([G|Gs]) :- findall(q(G,B),p(G,B),Cs), reverse(Cs,RCs),
              member(q(G,RB),RCs), append(RB,Gs,NGs), e3(NGs).
```

Ποιες είναι οι απαντήσεις που θα έδινε ένα σύστημα Prolog, και με ποια σειρά, στις ερωτήσεις "`?- e1([a(X)]).`", "`?- e2([a(X)]).`" και "`?- e3([a(X)]).`"; Τι ακριβώς υλοποιεί καθένα από τα κατηγορήματα `e1/1`, `e2/1` και `e3/1`; Τα κατηγορήματα `member/2`, `append/3` και `reverse/2` έχουν τη γνωστή σας λειτουργικότητα.

(β′) Θεωρήστε ότι ένας πίνακας μπορεί να αναπαρασταθεί στην Prolog σαν μία λίστα από τις γραμμές του, όπου κάθε γραμμή είναι μία λίστα από τα στοιχεία της. Έχοντας αυτό σαν δεδομένο, ορίστε το κατηγόρημα `matr_det/2` έτσι ώστε το το `matr_det(M,D)` να επιστρέφει στο `D` την ορίζουσα του πίνακα `M`. Για παράδειγμα:

```
?- matr_det([[4,3,2,1],[3,2,1,2],[1,4,1,3],[2,1,3,2]],D).
   D = 51
```

4. (α′) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το λογικό πρόγραμμα του διπλανού σχήματος; Γιατί οι ερμηνείες
$I_1 = \{$`p(d)`, `q(a)`, `q(b)`, `r(b)`, `r(c)`, `s(d)`$\}$
$I_2 = \{$`p(b)`, `p(d)`, `q(a)`, `q(b)`, `r(b)`, `r(d)`, `s(d)`$\}$
$I_3 = \{$`q(a)`, `q(b)`, `r(c)`, `r(d)`, `s(d)`$\}$
$I_4 = \{$`p(b)`, `p(d)`, `q(a)`, `q(b)`, `q(c)`, `r(b)`, `r(c)`, `r(d)`, `s(d)`$\}$
$I_5 = \{$`p(b)`, `q(a)`, `q(b)`, `r(b)`, `r(c)`, `r(d)`, `s(a)`, `s(d)`$\}$
δεν είναι μοντέλα Herbrand για το πρόγραμμα; Ποιο είναι το ελάχιστο μοντέλο Herbrand; Με ποιο τρόπο το κατασκευάσατε;

```
p(X) :- q(X),
        r(X).
r(X) :- s(X).
p(X) :- s(X).
q(a).
q(b).
r(b).
r(c).
s(d).
```

(β′) Εύκολα μπορεί να δει κανείς ότι το πρόβλημα των 3-βασιλισσών δεν έχει λύση. Θα όφειλε συνεπώς, κατά τη γνώμη σας, αν διατυπωθεί η ερώτηση

```
?- [X,Y,Z] :: 1..3, X ## Y, X ## Y+1, X ## Y-1,
    X ## Z, X ## Z+2, X ## Z-2, Y ## Z, Y ## Z+1, Y ## Z-1.
```

στην ECL^{i}PS^e να πάρει απ′ ευθείας αρνητική απάντηση; Είναι μοντελοποιημένο σωστά το πρόβλημα σαν σύνολο μεταβλητών πεδίων και περιορισμών επάνω σ′ αυτές;

(γ′) Σε μία γλώσσα λογικού προγραμματισμού που υποστηρίζει Ή-παραλληλία, με την οδηγία "`:- parallel p/1.`" δηλώνεται ότι οι προτάσεις που ορίζουν το κατηγόρημα `p/1` μπορούν να εξετασθούν παράλληλα για την ικανοποίηση ενός στόχου με αυτό το κατηγόρημα. Αν το κατηγόρημα `p/1` ορίζεται μέσω των δύο γεγονότων "`p(a).`" και "`p(b).`" και κατά τη διάρκεια ικανοποίησης ενός στόχου προκύψει ο υποστόχος `p(X)`, πιστεύετε ότι έχει κάποια πρακτική αξία να βρεθούν οι τιμές του `X` (δηλαδή `X=a` και `X=b`) παράλληλα; Εξηγήστε την άποψή σας.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α΄ Περιόδου 2001

1. (α΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `substitute/4` έτσι ώστε όταν αυτό καλείται σαν `substitute(X, L1, Y, L2)`, να επιστρέφει στο `L2` τη λίστα που προκύπτει αν αντικατασταθούν όλες οι εμφανίσεις του στοιχείου `X` στη λίστα `L1` με το στοιχείο `Y`. Αν η `L1` δεν περιέχει το `X`, τότε η `L2` να είναι η ίδια με την `L1`. Για παράδειγμα:

```
?- substitute(a, [b,c,d,e], s, L).
L = [b,c,d,e]
?- substitute(a, [b,c,a,d,a,e,a,a,f,g,a], s, L).
L = [b,c,s,d,s,e,s,s,f,g,s]
```

   (β΄) Ορίστε σε Prolog το κατηγόρημα `n_th/3` με τέτοιο τρόπο που η ερώτηση `n_th(I, L, X)` να αντιστοιχεί, για δεδομένη λίστα `L`, ένα στοιχείο `X` της `L` με τη θέση του `I` μέσα στην `L`. Για παράδειγμα:

```
?- n_th(5, [a,b,c,d,b,e], X).
X = b
?- n_th(I, [a,b,c,d,b,e], b).
I = 2     -> ;
I = 5
?- n_th(I, [a,b,c,d,b,e], X).
I = 1
X = a     -> ;
I = 2
X = b     -> ;
...................
```

2. Να επιλυθεί σε Prolog η εξής παραλλαγή του προβλήματος των N βασιλισσών: Να τοποθετηθούν σ΄ ένα $N \times N$ πλαίσιο M βασίλισσες έτσι ώστε όλες οι υπόλοιπες θέσεις να απειλούνται απ΄ αυτές, ενώ αυτές να μην απειλούνται μεταξύ τους (μία βασίλισσα απειλεί όλες τις θέσεις που βρίσκονται στην ίδια γραμμή, στήλη ή διαγώνιο μ΄ αυτήν). Είναι δυνατόν η λύση αυτού του προβλήματος να αντιμετωπίσει και το κλασικό πρόβλημα των N βασιλισσών; Δικαιολογήστε την απάντησή σας.

3. Δίνονται οι παρακάτω ορισμοί των κατηγορημάτων `intrvn/3` και `intrvl/3`:

```
intrvn(X, Y, X) :- X =< Y.
intrvn(X, Y, N) :-
     X < Y,
     X1 is X+1,
     intrvn(X1, Y, N).
```

```
intrvl(X, X, [X]).
intrvl(X, Y, [X|L]) :-
     X < Y,
     X1 is X+1,
     intrvl(X1, Y, L).
```

   Τι απαντήσεις δίνει ένα σύστημα Prolog, που έχει εφοδιαστεί με τους ορισμούς αυτούς, στις ερωτήσεις `?- intrvn(3, 7, N)` και `?- intrvl(3, 7, L)`; Ορίστε σε Prolog δύο νέα κατηγορήματα, το `new_intrvn/3` και το `new_intrvl/3`, το πρώτο με τη βοήθεια του `intrvl/3`, αλλά ακριβώς με την ίδια λειτουργικότητα με το `intrvn/3`, και το δεύτερο με τη βοήθεια του `intrvn/3`, αλλά ακριβώς με την ίδια λειτουργικότητα με το `intrvl/3`. Σχολιάστε ό,τι νομίζετε ότι έχει ενδιαφέρον σχετικά με το θέμα αυτό.

4. (α′) i. Πότε η βάση Herbrand που αντιστοιχεί σ′ ένα λογικό πρόγραμμα είναι απειροσύνολο; Δώστε δύο πολύ απλά παραδείγματα οριστικών προγραμμάτων με βάσεις Herbrand που είναι απειροσύνολα και στο μεν ένα το ελάχιστο μοντέλο να είναι πεπερασμένο, στο δε άλλο απειροσύνολο.

   ii. Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Τι επίπτωση έχει αυτό το γεγονός;

   iii. Γιατί κατά τη γνώμη σας ενώ η έννοια του μοντέλου ορίζεται για οποιοδήποτε λογικό πρόγραμμα, αυτή του ελάχιστου μοντέλου ορίζεται μόνο για οριστικά προγράμματα;

   (β′) Γιατί πρέπει να αποφεύγεται η χρήση του ενσωματωμένου κατηγορήματος `assert/1` στον προγραμματισμό σε Prolog; Πότε καταστρατηγείται η αρχή αυτή στην υλοποίηση εμπείρων συστημάτων και γιατί;

   (γ′) Δεδομένου ενός χάρτη (π.χ. του πολιτικού χάρτη της Ευρώπης), το πρόβλημα του χρωματισμού του συνίσταται στην αντιστοίχιση σε κάθε χώρα ενός από τέσσερα διαθέσιμα χρώματα, έτσι ώστε να μην υπάρχουν δύο γειτονικές χώρες με το ίδιο χρώμα. Σκιαγραφήστε τον τρόπο με τον οποίο θα αντιμετωπίζατε το πρόβλημα αυτό σε μία γλώσσα λογικού προγραμματισμού με περιορισμούς (π.χ. την ECL^{i}PS^e), δίνοντας τα βασικά σημεία του σχετικού προγράμματος, χωρίς, κατ′ ανάγκη, να υπεισέλθετε σε δευτερεύουσες τεχνικές λεπτομέρειες. Γνωρίζετε αν το πρόβλημα αυτό έχει πάντοτε λύση;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β΄ Περιόδου 2001

1. (α΄) Δύο τρόποι ορισμού σε Prolog της αντιστροφής μίας λίστας είναι αυτοί που φαίνονται παρακάτω με τα κατηγορήματα `reverse1/2` και `reverse2/2`.

```
reverse1([], []).
reverse1([X|L], R) :-
   reverse1(L, RL),
   append(RL, [X], R).

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

```
reverse2(L, R) :-
   rev_app(L, [], R).

rev_app([], R, R).
rev_app([X|L], A, R) :-
   rev_app(L, [X|A], R).
```

Σχεδιάστε τα δέντρα ανάλυσης που προκύπτουν κατά την αντιστροφή της λίστας `[1,2,3]` με καθένα απ΄ αυτά τα κατηγορήματα. Έχετε να σχολιάσετε κάτι; Πιστεύετε ότι έχει κάποια πλεονεκτήματα η κάθε μία από τις δύο υλοποιήσεις έναντι της άλλης; Αν η χρονική απόκριση των κατηγορημάτων αυτών ήταν αποδεκτή, αλλά όχι ιδιαίτερα γρήγορη, κατά την αντιστροφή μίας λίστας με 1000 στοιχεία, ποιο κατηγόρημα θα επιλέγατε για την αντιστροφή μίας λίστας με 100000 στοιχεία και γιατί;

(β΄) Γνωρίζετε ότι ο κλασικός τρόπος επεξεργασίας μίας λίστας στην Prolog είναι να δουλέψουμε αρχικά με το πρώτο στοιχείο της και αναδρομικά να επεξεργαστούμε τα υπόλοιπα στοιχεία. Γιατί πιστεύετε ότι δεν κάνουμε αυτή τη διαδικασία και αντίστροφα, δηλαδή πρώτα να ασχοληθούμε με το τελευταίο στοιχείο της λίστας και αναδρομικά να επεξεργαστούμε όλα τα άλλα πλην του τελευταίου; Μπορείτε να φανταστείτε κάποια συγκεκριμένη λειτουργία επάνω σε λίστες που θα ήταν βολικό κάτι τέτοιο; Τί είναι, κατά τη γνώμη σας, πιο εύκολο να υλοποιήσετε, ένα κατηγόρημα που να βρίσκει τα 3 πρώτα στοιχεία μίας λίστας ή ένα που να βρίσκει τα 3 τελευταία στοιχεία της και γιατί;

2. Ο Γιάννης αποφάσισε μία ημέρα να επισκεφθεί τα καταστήματα για την αγορά ενός υπολογιστικού συστήματος αποτελούμενου από την κεντρική υπολογιστική μονάδα, μία οθόνη, έναν εκτυπωτή και ένα σαρωτή. Με σκοπό να βρει τις πιο συμφέρουσες τιμές, επισκέφθηκε τέσσερα γνωστά καταστήματα, το Πλαίσιο, το Multirama, την Microland και το e-motion, όχι κατ΄ ανάγκη με αυτή τη σειρά. Συμπτωματικά, αγόρασε τελικά από καθένα απ΄ αυτά τα καταστήματα ένα τμήμα του υπολογιστικού του συστήματος. Όταν γύρισε σπίτι του, ο πατέρας του τον ρώτησε τι έκανε τελικά με τις αγορές του, αλλά ο Γιάννης του απάντησε με γρίφους. Συγκεκριμένα, του είπε ότι το σαρωτή τον αγόρασε από το Multirama, ότι αμέσως μετά την αγορά της κεντρικής μονάδας δεν επισκέφθηκε το e-motion, ότι το Πλαίσιο ήταν ο δεύτερος σταθμός του και ότι στη μεθεπόμενη επίσκεψη μετά την Microland αγόρασε την οθόνη. Γράψτε ένα πρόγραμμα Prolog για να βοηθήσετε τον πατέρα του Γιάννη να μάθει τι αγόρασε ο γιος του από κάθε κατάστημα και με ποια σειρά τα επισκέφθηκε.

3. (α΄) Να υλοποιηθεί σε Prolog το κατηγόρημα `givensum/3` έτσι ώστε το `givensum(L1, S, L2)` να επιστρέφει στο `L2` όλες τις δυνατές λίστες, μέσω οπισθοδρόμησης, με στοιχεία από τη λίστα `L1` τα οποία έχουν άθροισμα `S`. Για παράδειγμα:

```
?- givensum([5,2,-1,2,-3], 4, L).
L = [5,2,-3]       -> ;
L = [5,-1]         -> ;
L = [5,2,-3]       -> ;
L = [2,2]
yes.
```

Αν γνωρίζατε ότι τα στοιχεία της λίστας `L1` είναι πάντα θετικά, θα μπορούσατε να βελτιώσετε την υλοποίησή σας, ή δεν σας βοηθά αυτή η επιπλέον γνώση;

(β′) Ως γνωστόν, στην Prolog υπάρχουν τρία είδη ενδοσημασμένων (infix) τελεστών, οι `xfx`, `yfx` και `xfy` τελεστές. Γιατί, κατά τη γνώμη σας, δεν υπάρχουν και `yfy` τελεστές;

4. (α′) i. Πότε η βάση Herbrand που αντιστοιχεί σ′ ένα λογικό πρόγραμμα είναι απειροσύνολο; Δώστε δύο πολύ απλά παραδείγματα οριστικών προγραμμάτων με βάσεις Herbrand που είναι απειροσύνολα και στο μεν ένα το ελάχιστο μοντέλο να είναι πεπερασμένο, στο δε άλλο απειροσύνολο.

ii. Πότε το ελάχιστο μοντέλο ενός οριστικού προγράμματος είναι το κενό σύνολο; Τι επίπτωση έχει αυτό το γεγονός;

iii. Γιατί κατά τη γνώμη σας ενώ η έννοια του μοντέλου ορίζεται για οποιοδήποτε λογικό πρόγραμμα, αυτή του ελάχιστου μοντέλου ορίζεται μόνο για οριστικά προγράμματα;

(β′) Σε μία γλώσσα λογικού προγραμματισμού που υποστηρίζει Ή-παραλληλία, με την οδηγία "`:- parallel p/1.`" δηλώνεται ότι οι προτάσεις που ορίζουν το κατηγόρημα `p/1` μπορούν να εξετασθούν παράλληλα για την ικανοποίηση ενός στόχου με αυτό το κατηγόρημα. Αν το κατηγόρημα `p/1` ορίζεται μέσω των δύο γεγονότων "`p(a).`" και "`p(b).`" και κατά τη διάρκεια ικανοποίησης ενός στόχου προκύψει ο υποστόχος `p(X)`, πιστεύετε ότι έχει κάποια πρακτική αξία να βρεθούν οι τιμές του `X` (δηλαδή `X=a` και `X=b`) παράλληλα; Εξηγήστε την άποψή σας.

(γ′) Θα ήταν κατά τη γνώμη σας πιο συμφέρον να επιλυθεί κάποιο από τα προβλήματα των θεμάτων 2. ή 3.(α′) προηγουμένως (ή ίσως και τα δύο) με τη βοήθεια της τεχνολογίας του λογικού προγραμματισμού με περιορισμούς; Αν ναι, περιγράψτε συνοπτικά τον τρόπο με τον οποίο θα το (τα) επιλύνατε.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α' Περιόδου 2000

1. (α) Έστω το εξής πρόγραμμα Prolog:

```
proc_list(L1, L2) :- findall(Y, (member(X, L1), proc_elem(X, Y)), L2).
proc_elem(X, Y) :- Y is 4*X+1.
```

Τι επιστρέφει στη μεταβλητή `L` η ερώτηση `?- proc_list([1,2,3], L)`; Προτείνετε κάποιον άλλο ορισμό του `proc_list/2` ίδιο λειτουργικά με αυτόν που δίνεται, αλλά διαφορετικό σε φιλοσοφία. Σχολιάστε τους δύο ορισμούς.

(β) Να υλοποιηθεί σε Prolog το `disjoint/2` έτσι ώστε το `disjoint(List1, List2)` να είναι αληθές όταν οι λίστες `List1` και `List2` είναι ξένες μεταξύ τους. Για παράδειγμα, το `disjoint([1,5], [2,3,8])` είναι αληθές και το `disjoint([1,7,9], [2,7])` είναι ψευδές.

2. "*Το Βέλγιο συνορεύει με τη Γαλλία, τη Γερμανία, το Λουξεμβούργο και την Ολλανδία. Η Γαλλία συνορεύει με τη Γερμανία. Το Λουξεμβούργο συνορεύει με τη Γαλλία. Επίσης συνορεύει με τη Γερμανία. Η Ολλανδία συνορεύει με τη Γερμανία. Όταν μία χώρα Α συνορεύει με μία χώρα Β, τότε και η Β συνορεύει με την Α. Συνορεύει η Γερμανία με το Λουξεμβούργο; Ποιες χώρες συνορεύουν με τη Γαλλία; Με ποιες χώρες συνορεύουν και η Γερμανία και το Βέλγιο; Ποιες χώρες συνορεύουν με δύο ακριβώς χώρες; Είναι δυνατόν δύο χώρες να συνορεύουν με μία τρίτη χώρα αλλά να μην συνορεύουν μεταξύ τους;*". Να περιγραφεί σε Prolog ο κόσμος του κειμένου που προηγήθηκε (τόσο η καταφατική γνώση, όσο και οι ερωτήσεις που τέθηκαν). Μπορείτε, με την αναπαράσταση που επιλέξατε για τον προηγούμενο κόσμο, να διατυπώσετε σε Prolog και την ερώτηση "*Ποιες χώρες δεν συνορεύουν με την Ολλανδία;*"; Αν ναι, κάντε το. Αν όχι, τι αλλαγή ή προσθήκη πρέπει να κάνετε στην προηγούμενη αναπαράσταση για να μπορέσετε να διατυπώσετε σωστά την ερώτηση αυτή;

3. (α) Έστω το πρόγραμμα Prolog:

```
unmarried_student(X) :- not married(X), student(X).
student(bill).
married(joe).
```

Να δοθούν οι απαντήσεις στις ερωτήσεις:

i. `?- unmarried_student(bill).`
ii. `?- unmarried_student(joe).`
iii. `?- unmarried_student(X).`

Είναι οι απαντήσεις αυτές που θα περίμενε κάποιος λογικά να πάρει; Εξηγήστε την άποψή σας.

(β) Ένα υποθετικό πρόγραμμα Prolog αποτελείται από 500000 γεγονότα με κατηγόρημα `pub_sect_emp/1`, που δηλώνουν τα ονόματα των δημοσίων υπαλλήλων μίας χώρας, 300 γεγονότα με κατηγόρημα `memb_of_parl/1`, που δηλώνουν τα ονόματα των βουλευτών της χώρας αυτής, και 10000000 γεγονότα με κατηγόρημα `fath/2`, που δηλώνουν όλα τα ζευγάρια "πατέρας-παιδί" των κατοίκων της χώρας. Κάποιος θέλει να μάθει όλα τα ζευγάρια `(X, Y)`, όπου ο/η `X` είναι δημόσιος υπάλληλος με πατέρα το βουλευτή `Y`, και υποβάλλει την εξής ερώτηση:

```
?- pub_sect_emp(X), memb_of_parl(Y), fath(Y, X).
```

Είναι σωστή αυτή η ερώτηση; Αν ναι, είναι ο αποδοτικότερος τρόπος για να βρεθεί το ζητούμενο; Αν όχι, τι θα προτείνατε εσείς και γιατί; (Μπορείτε να υποθέσετε ότι το χρησιμοποιούμενο σύστημα Prolog έχει τη δυνατότητα, μέσω κατάλληλων μηχανισμών δεικτοδότησης, να ελέγχει σε πολύ λίγο χρόνο αν συγκεκριμένο γεγονός βρίσκεται μέσα στο πρόγραμμα που έχει φορτωθεί, ο οποίος χρόνος είναι και σταθερός, δηλαδή ανεξάρτητος του πλήθους των γεγονότων με το ίδιο κατηγόρημα).

4. (α) Έστω το εξής οριστικό πρόγραμμα:

```
p(X) :- q(a).
q(b) :- r(X).
```

Δώστε δύο μοντέλα του προγράμματος διάφορα από το ελάχιστο μοντέλο του. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος; Μπορείτε, με βάση την απάντησή σας, να κάνετε κάποιο γενικότερο σχόλιο, ανεξάρτητο από το συγκεκριμένο πρόγραμμα;

(β) Ως γνωστόν, η σημασιολογία σταθερού σημείου και η λειτουργική σημασιολογία είναι δύο μέθοδοι μελέτης της σημασίας των λογικών προγραμμάτων. Πιστεύετε ότι υπάρχει κάποια σχέση μεταξύ των μεθόδων αυτών και των συλλογιστικών που χρησιμοποιούμε στα έμπειρα συστήματα;

(γ) Κάποιος χρησιμοποιεί ένα σύστημα Prolog που υποστηρίζει Η-παραλληλία και έστω ότι για ένα κατηγόρημα, στο οποίο υπάρχουν αποκοπές (`!`) στα σώματα ενός ή περισσοτέρων κανόνων στον ορισμό του, έχει δηλώσει (π.χ. μέσω μίας οδηγίας `parallel`) ότι επιθυμεί να εξετάζονται οι προτάσεις του ορισμού του παράλληλα. Πώς θα χαρακτηρίζατε τη δήλωσή του αυτή και γιατί;

(δ) Υπάρχει, κατά τη γνώμη σας, καμία ομοιότητα μεταξύ του εξομοιωτή WAM σ' ένα σύστημα Prolog και του Pentium στη μητρική πλακέτα του προσωπικού μας υπολογιστή; Εξηγήστε την άποψή σας.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β' Περιόδου 2000

1. (α) Ζητήσατε από κάποιους να σας ορίσουν σε Prolog τη γνωστή διαδικασία για τον υπολογισμό του μήκους μίας λίστας και πήρατε δύο ειδών απαντήσεις, την `length1/2` και `length2/2`:

```
length1([],0).
length1([_|L],N) :-
        length1(L,N1),
        N is N1 + 1.
```

```
length2(L,N) :- length2(L,0,N).
length2([],N,N).
length2([_|L],N1,N) :-
   N2 is N1 + 1, length2(L,N2,N).
```

Μπορείτε να τις σχολιάσετε; Τι πρόβλημα παρουσιάζεται αν χρησιμοποιηθούν οι ορισμοί αυτοί και για την κατασκευή λίστας (από μεταβλητές) δεδομένου μήκους; Μπορείτε να προτείνετε κάποια εναλλακτική υλοποίηση για αυτόν το σκοπό; Ο ορισμός που δώσατε μπορεί να χρησιμοποιηθεί και για τον υπολογισμό του μήκους δεδομένης λίστας; Αν όχι, μπορείτε να δώσετε έναν ορισμό που να δουλεύει σωστά και με τους δύο τρόπους;

(β) Να υλοποιηθεί σε Prolog το κατηγόρημα `primes/2` έτσι ώστε το `primes(N, L)` να επιστρέφει στο `L` τη λίστα όλων των πρώτων αριθμών που είναι μικρότεροι ή ίσοι του `N`. Για παράδειγμα:

```
?- primes(80,L).
L = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79]
```

2. Το φίδι των ΤΥΠΑ έχει στο σώμα του ένα επαναλαμβανόμενο σχήμα (όχι, κατ' ανάγκη ακέραιο αριθμό φορών), το οποίο αποτελείται από σχέδια καθένα από τα οποία μπορεί να παρασταθεί από ένα χαρακτήρα. Επίσης, είναι γνωστό ότι το φίδι των ΤΥΠΑ αρέσκεται να ξεκουράζεται σε μία "φιδοειδή" διευθέτηση καταλαμβάνοντας το χώρο ενός ορθογωνίου παραλληλογράμμου δεδομένων διαστάσεων (σε μονάδα μέτρησης το μέγεθος του σχεδίου/χαρακτήρα που συνθέτει τα επαναλαμβανόμενα σχήματα). Γράψτε ένα κατηγόρημα `typa_snake/3`, το οποίο όταν καλείται σαν `typa_snake(Pattern, Cols, Rows)` να εκτυπώνει τη στάση ξεκούρασης του φιδιού των ΤΥΠΑ, με επαναλαμβανόμενο σχήμα τους χαρακτήρες της λίστας `Pattern` και πλάτος (αντίστοιχα, μήκος) του ορθογωνίου που προαναφέρθηκε το μήκος της λίστας `Cols` (αντίστοιχα, `Rows`). Το πρόγραμμά σας δεν πρέπει να χρησιμοποιήσει καθόλου αριθμητική, δηλαδή δεν θα περιλαμβάνει κανένα από τα σύμβολα `is`, `<`, `>`, `>=`, `=<`, `+`, `-`, `*`. Ένα παράδειγμα εκτέλεσης του προγράμματος είναι το εξής:

```
?- typa_snake([a,b,c,d,e,f,g], [_,_,_,_,_,_,_,_,_,_,_,_,_], [_,_,_,_,_,_,_]).
abcdefgabcd
agfedcbagfe
bcdefgabcde
bagfedcbagf
cdefgabcdef
cbagfedcbag
```

3. (α) Σ' ένα πρόγραμμα Prolog έχουμε δηλώσει τα εξής:

```
man(paul).      man(nick).      woman(mary).      woman(joan).
```

Γνωρίζουμε ότι αν υποβάλουμε την ερώτηση `?- man(X)` θα πάρουμε σαν απαντήσεις τις `X = paul` και `X = nick`, ενώ η ερώτηση `?- woman(X)` θα μας δώσει `X = mary` και

`X = joan`. Γιατί αν ρωτήσουμε `?- not man(X)` δεν θα πάρουμε `X = mary` και `X = joan` και όταν ρωτήσουμε `?- not woman(X)` οι απαντήσεις δεν θα είναι αυτές που ίσως θα επιθυμούσαμε, δηλαδή `X = paul` και `X = nick`; Δώστε τόσο την τεχνική (διαδικασία υπολογισμού μίας απάντησης) όσο και τη θεωρητική (σημασία των ερωτήσεων με άρνηση) αιτιολόγηση.

(β) Η συνάρτηση Ackermann ορίζεται ως εξής:

$$ackermann(M,N) = \begin{cases} N+1 & \text{αν } M = 0 \text{ και } N \geq 0 \\ ackermann(M-1,1) & \text{αν } M > 0 \text{ και } N = 0 \\ ackermann(M-1, ackermann(M, N-1)) & \text{αν } M > 0 \text{ και } N > 0 \end{cases}$$

Να υλοποιηθεί σε Prolog το κατηγόρημα `ack/3` έτσι ώστε ο στόχος `ack(M, N, A)` να επιστρέφει στο `A` την τιμή της συνάρτησης Ackermann για δεδομένες τιμές `M` και `N`.

4. (α) Έστω το εξής οριστικό πρόγραμμα:

```
p(X) :- q(a).
q(b) :- r(X).
```

Δώστε δύο μοντέλα του προγράμματος διάφορα από το ελάχιστο μοντέλο του. Ποιο είναι το ελάχιστο μοντέλο του προγράμματος; Μπορείτε, με βάση την απάντησή σας, να κάνετε κάποιο γενικότερο σχόλιο, ανεξάρτητο από το συγκεκριμένο πρόγραμμα;

(β) Γιατί, κατά τη γνώμη σας, όταν επιλύουμε ένα πρόβλημα ικανοποίησης περιορισμών με την τεχνολογία του λογικού προγραμματισμού με περιορισμούς, μετά από τη διατύπωση των περιορισμών που μοντελοποιούν το πρόβλημα γεννάμε μη-ντετερμινιστικά τιμές για τις μεταβλητές; Δεν αρκεί μόνο η διατύπωση των περιορισμών για να βρούμε τις επιθυμητές λύσεις;

(γ) Γνωρίζετε ότι η μετάφραση/μεταγλώττιση (compilation) ενός προγράμματος Prolog συνίσταται στο μετασχηματισμό του σ' ένα πρόγραμμα χαμηλού επιπέδου, για τη λεγόμενη μηχανή WAM. Στη συνέχεια, η εκτέλεση του προγράμματος γίνεται με τη βοήθεια ενός εξομοιωτή της WAM. Γιατί, κατά τη γνώμη σας, δεν δουλεύουμε με την Prolog όπως και με τις περισσότερες κλασικές γλώσσες προγραμματισμού; Δηλαδή, γιατί δεν μεταγλωττίζουμε ένα πρόγραμμα Prolog απ' ευθείας στη γλώσσα μηχανής του μικροεπεξεργαστή μας, έτσι ώστε η εκτέλεσή του να γίνει παραδοσιακά από το υλικό (hardware); Εν πάση περιπτώσει, αν μας υποχρέωνε κάποιος να δουλέψουμε με αυτόν τον τρόπο, τι θα μπορούσαμε να κάνουμε για να κατασκευάσουμε ένα εκτελέσιμο πρόγραμμα σε γλώσσα μηχανής που να αντιστοιχεί σε κάποιο δεδομένο πρόγραμμα Prolog;

(δ) Αν ένα σύστημα Prolog υποστηρίζει Η-παραλληλία, υπάρχει περίπτωση να έχουμε δηλώσει σε κάποιο πρόγραμμα ότι θέλουμε ένα συγκεκριμένο κατηγόρημα να το επεξεργαζόμαστε με Η-παράλληλο τρόπο και η απόδοση του προγράμματός μας να είναι χειρότερη από το να μην εκμεταλλευόμασταν καθόλου την Η-παραλληλία, δηλαδή να χειριζόμασταν το κατηγόρημα αυτό με τον κλασικό σειριακό τρόπο;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Εξετάσεις Α' Περιόδου 1999

1. (α) Υποθέστε ότι κάνετε γνωστό σε κάποιους ότι υπάρχει ορισμένο ένα κατηγόρημα `prop/1` το οποίο επιτυγχάνει όταν του δοθεί σαν όρισμα ένας όρος που έχει κάποια συγκεκριμένη ιδιότητα. Στη συνέχεια, τους ζητάτε να υλοποιήσουν ένα άλλο κατηγόρημα `prop_all/1` το οποίο να επιτυγχάνει όταν του δίνεται σαν όρισμα μία λίστα από όρους όπου όλοι έχουν την ιδιότητα που αναφέρθηκε προηγουμένως (αυτήν που ελέγχει το `prop/1`). Οι απαντήσεις που παίρνετε από δύο άτομα είναι αυτές που φαίνονται στη συνέχεια αριστερά και δεξιά. Δεν είναι οι καλύτερες δυνατές απαντήσεις γιατί έχουν λάθη, κάπου είναι μη αποδοτικές και αλλού δυσανάγνωστες. Αφού κάνετε τις απαραίτητες διορθώσεις, σχολιάστε τις σωστές απαντήσεις από όποιες απόψεις κρίνετε ενδιαφέρουσες.

```
prop_all([_]).                  |       prop_all(L) :-
                                |          not one_not_prop(L).
                                |
prop_all([X|L]) :-              |       one_not_prop(list) :-
   prop_all(L),                 |          member(X,list),
   prop(X).                     |          $@!|&*#@#@!|^&
```

   (β) Ορίστε σε Prolog τα κατηγορήματα `intersection/3` και `union/3`, τα οποία όταν καλούνται σαν `intersection(S1,S2,S3)` και `union(S1,S2,S3)` να επιστρέφουν στη μεταβλητή `S3` την τομή και την ένωση, αντίστοιχα, των συνόλων `S1` και `S2`. Στην υλοποίηση των παραπάνω κατηγορημάτων μπορείτε να θεωρήσετε ότι τα σύνολα παριστάνονται σαν λίστες που δεν περιέχουν επαναλαμβανόμενα στοιχεία, χωρίς να απαιτείται να το ελέγχετε αυτό.

2. Έστω ο εξής κόσμος: "*Ο Κώστας και η Ελένη έχουν παιδιά το Νίκο και την Κατερίνα. Ο Νίκος και η Μαρία έχουν το Βασίλη και το Γιάννη, ενώ ο Χρήστος και η Κατερίνα την Ευγενία. Ο Βασίλης και η Δήμητρα έχουν παιδιά την Ηρώ, το Γιώργο και τη Νίκη και, τέλος, ο Γιάννης και η Αγγελική έχουν τον Πέτρο, ενώ ο Παύλος και η Ευγενία είναι παντρεμένοι, αλλά δεν έχουν παιδιά*". Αναπαραστήστε τον κόσμο αυτό σε Prolog. Ορίστε τα κατηγορήματα `married_woman/1`, `unknown_parents/1`, `one_child/2`, `predecessor/2` και `no_of_cousins/2` έτσι ώστε η ερώτηση `?-married_woman(X)` να αληθεύει όταν το `X` παριστάνει κάποια παντρεμένη γυναίκα, η ερώτηση `?-unknown_parents(X)` να αληθεύει όταν το `X` παριστάνει κάποιο άτομο του οποίου δεν είναι γνωστοί οι γονείς, η ερώτηση `?-one_child(X,Y)` να αληθεύει όταν ο `X` και η `Y` έχουν ακριβώς ένα παιδί, η ερώτηση `?-predecessor(X,Y)` να αληθεύει όταν το `X` είναι πρόγονος του `Y` και, τέλος, η ερώτηση `?-no_of_cousins(X,N)` να αληθεύει όταν το πλήθος των πρώτων εξαδέλφων του `X` είναι `N`.

3. (α) Ορίστε σε Prolog το κατηγόρημα `rotated/2`, έτσι ώστε η ερώτηση `rotated(L1,L2)` με δεδομένες λίστες `L1` και `L2` να επιτυγχάνει όταν οι λίστες αυτές ταυτίζονται, έστω και μετά από περιστροφή κάποιας από αυτές αυθαίρετο αριθμό θέσεων, και να αποτυγχάνει στην αντίθετη περίπτωση. Για να γίνει περισσότερο κατανοητή η ζητούμενη λειτουργικότητα, ένα παράδειγμα επιτυχούς ερώτησης είναι το `?-rotated([a,b,c,d,e],[d,e,a,b,c])`.

   (β) Υλοποιήστε σε Prolog το βασικό κορμό του μηχανισμού οπίσθιας συλλογιστικής ενός εμπείρου συστήματος θεωρώντας ότι δεν είναι απαραίτητο να είναι γνωστές εξ αρχής οι βασικές πληροφορίες που πρέπει να δοθούν από το χρήστη για την εξαγωγή κάποιου συμπεράσματος. Υποθέστε ότι σας είναι γνωστό ποιες πληροφορίες αποτελούν παρατηρήσιμα

ενδεχόμενα και δεν είναι αποδείξιμες από if-then κανόνες, αλλά πρέπει να επιβεβαιωθεί ή να απορριφθεί η ισχύς τους από το χρήστη του εμπείρου συστήματος. Όταν χρειαστείτε κάποια κατηγορήματα εισόδου, μπορείτε, αν δεν γνωρίζετε τα πραγματικά, να χρησιμοποιήσετε άλλα δικής σας έμπνευσης.

4. (α) Σε μία γλώσσα λογικού προγραμματισμού που υποστηρίζει Η-παραλληλία, με την οδηγία "`:- parallel p/1.`" δηλώνεται ότι οι προτάσεις που ορίζουν το κατηγόρημα `p/1` μπορούν να εξετασθούν παράλληλα για την ικανοποίηση ενός στόχου με αυτό το κατηγόρημα. Αν το κατηγόρημα `p/1` ορίζεται μέσω των δύο γεγονότων "`p(a).`" και "`p(b).`" και κατά τη διάρκεια ικανοποίησης ενός στόχου προκύψει ο υποστόχος `p(X)`, πιστεύετε ότι έχει κάποια πρακτική αξία να βρεθούν οι τιμές του `X` (δηλαδή `X=a` και `X=b`) παράλληλα; Εξηγήστε την άποψή σας.

   (β) Τι είναι μία οριστική πρόταση και τι ένα οριστικό πρόγραμμα στο λογικό προγραμματισμό; Δώστε το απλούστερο δυνατό πρόγραμμα που δεν είναι οριστικό. Γνωρίζετε ότι η τομή δύο μοντέλων ενός οριστικού προγράμματος είναι επίσης μοντέλο του προγράμματος. Δείξτε μέσω ενός αντιπαραδείγματος ότι αυτό δεν ισχύει για προγράμματα που δεν είναι οριστικά.

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Α' Περιόδου 1998

1. (α) Ως γνωστόν, το "," με το οποίο συνδέονται οι στόχοι στο σώμα ενός κανόνα Prolog συμβολίζει την πράξη της σύζευξης μεταξύ των στόχων αυτών, με την έννοια ότι πρέπει όλοι οι στόχοι να είναι αληθείς για να είναι αληθής και η κεφαλή του κανόνα. Υπάρχει κάποιος ανάλογος τρόπος για να εκφράσουμε διάζευξη μεταξύ στόχων; Προσφέρει κάτι ουσιαστικό αυτός ο τρόπος στον προγραμματισμό με την Prolog; Δηλαδή, μας είναι απολύτως απαραίτητος ή μπορούμε να ζήσουμε και χωρίς αυτόν; Τι γνώμη έχετε για την κατάχρηση της διάζευξης στα προγράμματα Prolog;

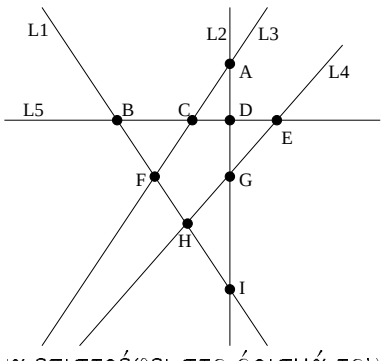
   (β) Έστω ότι έχετε στη διάθεσή σας ένα γενεαλογικό δέντρο ορισμένο μέσω του κατηγορήματος `parent/2` (το πρώτο όρισμα είναι γονιός του δεύτερου), για παράδειγμα το εξής:

```
parent(eva,bob).  parent(bob,ann).  parent(ann,jim).  parent(jim,kim).
parent(jim,ben).  parent(bob,pam).  parent(pam,sam).  parent(eva,pat).
parent(pat,joe).  parent(joe,gus).  parent(joe,leo).  parent(leo,liz).
```

   Ορίστε σε Prolog το κατηγόρημα `same_generation/2` με τέτοιο τρόπο ώστε ο στόχος `same_generation(X,Y)` να επιτυγχάνει όταν τα άτομα `X` και `Y` είναι της ίδιας γενιάς, δηλαδή θα βρισκόντουσαν στο ίδιο επίπεδο σε μία γραφική αναπαράσταση του γενεαλογικού δέντρου. Για παράδειγμα, με τα παραπάνω η `kim` και η `liz` είναι της ίδιας γενιάς.

2. (α) Ορίστε σε Prolog το κατηγόρημα `sel/3` έτσι ώστε ο στόχος `sel(X,L,R)` να αληθεύει όταν το `X` είναι στοιχείο της λίστας `L` και `R` είναι η λίστα των στοιχείων της `L` που βρίσκονται μετά το `X`. Το κατηγόρημα αυτό να δουλεύει και μη-ντετερμινιστικά, δηλαδή να μπορεί να καλείται και με μεταβλητά `X` και `R`. Για παράδειγμα, ο στόχος `sel(X,[a,b,c],R)` να έχει μέσω οπισθοδρόμησης τρεις λύσεις (`a`, `b` και `c` για το `X` και `[b,c]`, `[c]` και `[]` για το `R`, αντίστοιχα).

   (β) Δίνεται ένα σύνολο από ευθείες και τα σημεία στα οποία αυτές τέμνονται, όπως φαίνεται στο διπλανό σχήμα. Θεωρήστε, επίσης, την αναπαράσταση σε Prolog του κόσμου αυτού με ένα γεγονός της μορφής:




```
lines([line(l1,[b,f,h,i]),
       line(l2,[a,d,g,i]),
       line(l3,[a,c,f]),
       line(l4,[e,g,h]),
       line(l5,[b,c,d,e])]).
```

   Ορίστε σε Prolog ένα κατηγόρημα `is_triangle/1` το οποίο να επιστρέφει στο όρισμά του μέσω οπισθοδρόμησης όλα τα τρίγωνα που υπάρχουν στο σχήμα μας. Για παράδειγμα:

```
?- is_triangle(T).
T = triangle(b,f,c)  -> ;
T = triangle(b,h,e)  -> ;
T = triangle(b,i,d)  -> ;
T = triangle(f,i,a)  -> ;
.....................
```

   Θα εκτιμηθεί περισσότερο ένας ορισμός αν δεν επιστρέφει περισσότερες από μία φορά ένα τρίγωνο (με αναδιάταξη των κορυφών του, π.χ. και σαν `BFC` και σαν `FCB`).

3. (α) Ορίστε σε Prolog ένα κατηγόρημα `repeatN/3` έτσι ώστε το `repeatN(L1,N,L2)` να επιτυγχάνει όταν η λίστα `L2` είναι η παράθεση `N` φορές της λίστας `L1`. Γράψτε έτσι το κατηγόρημα αυτό ώστε για δεδομένη λίστα `L2` να επιστρέφει μη-ντετερμινιστικά όλες τις δυνατές λύσεις. Δηλαδή:

```
?- repeatN(L,N,[a,b,c,a,b,c,a,b,c,a,b,c]).
L = [a,b,c]
N = 4        -> ;
L = [a,b,c,a,b,c]
N = 2        -> ;
L = [a,b,c,a,b,c,a,b,c,a,b,c]
N = 1
yes.
```

Δουλεύει σωστά ο ορισμός σας αν δοθούν τα `L1` και `N` και ζητείται το `L2`; Αν όχι, χωρίς να δώσετε την απαιτούμενη υλοποίηση, περιγράψτε πως θα αντιμετωπίζατε το πρόβλημα ώστε να καλύπτετε και τους δύο τρόπους χρήσης του `repeatN/3`.

(β) Δυαδικό λεξικό είναι ένα δυαδικό δέντρο του οποίου κάθε κόμβος είναι "μεγαλύτερος" από όλους τους κόμβους του αριστερού υποδέντρου του και "μικρότερος" από όλους τους κόμβους του δεξιού υποδέντρου του. Οι όροι "μεγαλύτερος" και "μικρότερος" αναφέρονται σε μία δεδομένη σχέση διάταξης. Αφού προτείνετε μία κατάλληλη αναπαράσταση δυαδικών δέντρων (άρα και δυαδικών λεξικών), ορίστε σε Prolog ένα κατηγόρημα `dictionary/1` το οποίο να επιτυγχάνει όταν το όρισμα που του δίνεται είναι δυαδικό λεξικό.

4. (α) Έστω το λογικό πρόγραμμα P:

$$\begin{aligned} myplus(X, 0, X) &\longleftarrow \\ myplus(X, s(Y), s(Z)) &\longleftarrow myplus(X, Y, Z) \end{aligned}$$

Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το πρόγραμμα αυτό; Είναι δυνατόν ένα μοντέλο του P να περιέχει το άτομο $myplus(0, 0, s(0))$; Αν T_P είναι η απαιτούμενη απεικόνιση για τη σημασιολογική μελέτη του P μέσω σταθερού σημείου, ποια είναι η εικόνα $T_P(\emptyset)$ της κενής ερμηνείας $\emptyset$; Ο στόχος $\longleftarrow myplus(s(0), s(s(0)), s(s(s(0))))$ ανήκει στο σύνολο επιτυχίας $SS(P)$ του P; Αν ναι, δώστε την SLD-απόρριψη του στόχου αυτού.

(β) Τι γνωρίζετε για την υλοποίηση συστημάτων λογικού προγραμματισμού μέσω μεταφραστών/μεταγλωττιστών (compilers);

(γ) Έχει κάποιο ιδιαίτερο πλεονέκτημα η τεχνολογία του λογικού προγραμματισμού σχετικά με την υποστήριξη της οπίσθιας συλλογιστικής σε έμπειρα συστήματα; Θεωρείτε επαρκές αυτό το πλεονέκτημα για ό,τι λειτουργίες χρειάζεται να έχει ένα έμπειρο σύστημα;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β' Περιόδου 1998

1. Σε κάποιο πανεπιστημιακό τμήμα, ισχύουν, μεταξύ άλλων, τα εξής: "*Ο κύριος Τζαφέρης διδάσκει κάθε Τρίτη 11-13 και Πέμπτη 10-12 το υποχρεωτικό μάθημα του 1ου εξαμήνου 'Εισαγωγή στον Προγραμματισμό' στην αίθουσα Α. Οι κύριοι Μολυμπάκης και Θεοφάνους διδάσκουν το υποχρεωτικό μάθημα του 3ου εξαμήνου 'Ηλεκτρονική και Εφαρμογές στην Πληροφορική' κάθε Δευτέρα 13-15 και Πέμπτη 13-16 στην αίθουσα Γ. Η 'Αριθμητική Γραμμική Άλγεβρα' είναι βασικό μάθημα 1ης κατεύθυνσης του 3ου εξαμήνου και διδάσκεται από τον κύριο Τζαφέρη κάθε Τρίτη 9-11 και Τετάρτη 13-15 στην αίθουσα Γ. Ο κύριος Σαγκριώτης διδάσκει το υποχρεωτικό μάθημα του 5ου εξαμήνου 'Εισαγωγή στα Συστήματα Ψηφιακών Επικοινωνιών' κάθε Δευτέρα 9-11 στην αίθουσα Γ και κάθε Πέμπτη 13-15 στην αίθουσα Δ. Το μάθημα 'Μεταγλωττιστές' είναι βασικό 1ης κατεύθυνσης και επιλογή 2ης κατεύθυνσης στο 5ο εξάμηνο και διδάσκεται από τον κύριο Κοτρώνη κάθε Δευτέρα 16-18 και Πέμπτη 9-11 στην αίθουσα Δ. Ο κύριος Σφηκόπουλος κάθε Παρασκευή 12-15 διδάσκει στην αίθουσα Β1 την επιλογή 3ης κατεύθυνσης 'Τηλεπικοινωνιακά Ψηφιακά Δίκτυα' για το 7ο εξάμηνο.*"

   (α) Δώστε έναν τρόπο αναπαράστασης του παραπάνω κόσμου σε Prolog.

   (β) Ορίστε κατάλληλα κατηγορήματα με τη βοήθεια των οποίων να μπορείτε να απαντάτε σε ερωτήσεις, όπως:

   - Πόσες ώρες την εβδομάδα διδάσκεται κάποιο μάθημα;
   - Ποια είναι τα μαθήματα συγκεκριμένου τύπου ενός εξαμήνου;
   - Ποιες ημέρες έχει μάθημα κάποιος διδάσκων;

   (γ) Διατυπώστε συγκεκριμένα στιγμιότυπα των προηγούμενων ερωτήσεων σε Prolog.

   (δ) Προτείνετε μία εξ ίσου (αν όχι περισσότερο) ενδιαφέρουσα ερώτηση με τις παραπάνω και ορίστε κατηγόρημα που να είναι σε θέση να βρει τη ζητούμενη απάντηση.

2. Έστω ότι δίνονται τα παρακάτω γεγονότα Prolog:

```
activity(a01,act(0,3)). activity(a06,act(6,9)).   activity(a11,act(14,17)).
activity(a02,act(0,4)). activity(a07,act(9,10)).  activity(a12,act(16,18)).
activity(a03,act(1,5)). activity(a08,act(9,13)).  activity(a13,act(17,19)).
activity(a04,act(4,6)). activity(a09,act(11,14)). activity(a14,act(18,20)).
activity(a05,act(6,8)). activity(a10,act(12,15)). activity(a15,act(19,20)).

persons([p1,p2,p3]).
```

   Τα γεγονότα `activity/2` κωδικοποιούν δραστηριότητες που θα πρέπει να στελεχωθούν από άτομα. Κάθε γεγονός $activity(A, act(S, E))$ σημαίνει ότι η δραστηριότητα A αρχίζει τη χρονική στιγμή S και τελειώνει τη χρονική στιγμή E. Το γεγονός `persons/1` έχει σαν όρισμα μία λίστα από άτομα που είναι διαθέσιμα να στελεχώσουν δραστηριότητες. Υποθέστε ότι κάθε δραστηριότητα πρέπει να στελεχωθεί από ένα ακριβώς άτομο και ότι κάθε άτομο μπορεί να αναλάβει όσες δραστηριότητες απαιτούνται. Υπάρχει όμως ο περιορισμός ότι όχι μόνο δεν μπορεί ένα άτομο να αναλάβει δύο χρονικά επικαλυπτόμενες δραστηριότητες, αλλά θα πρέπει μεταξύ δύο οιωνδήποτε διαδοχικών δραστηριοτήτων κάθε ατόμου να μεσολαβεί τουλάχιστον μία μονάδα χρόνου. Με τα δεδομένα αυτά, γράψτε ένα πρόγραμμα Prolog, για παράδειγμα μέσω της υλοποίησης ενός κατηγορήματος `assignment/1`, το οποίο να αναθέτει τις δοθείσες δραστηριότητες στα δοθέντα άτομα με εφικτό τρόπο. Ένα παράδειγμα ερώτησης και της αντίστοιχης απάντησης στο πρόγραμμα, θα μπορούσε να ήταν το εξής:

```
?- assignment(A).
A = [p1 - a01, p2 - a02, p3 - a03, p1 - a04, p2 - a05, p3 - a06,
     p1 - a07, p2 - a08, p1 - a09, p3 - a10, p2 - a11, p1 - a12,
     p3 - a13, p2 - a14, p1 - a15]     -> ;
A = ....................................
```

3. (α) Έχοντας στη διάθεσή μας ένα πρόγραμμα Prolog που ορίζει ένα κατηγόρημα, έστω `p/1`, σαν το

   ```
   p(a).          p(b).          p(c).
   ```

   η υποβολή της ερώτησης `?- p(X)` μας επιστρέφει, όπως γνωρίζουμε, μέσω οπισθοδρόμησης τα `X` εκείνα για τα οποία ισχύει η ιδιότητα `p` (εδώ `a`, `b` και `c`). Υπάρχει τρόπος να ρωτήσουμε "*ποια είναι τα* `X` *εκείνα για τα οποία* **δεν** *ισχύει η ιδιότητα* `p`;" Τι σημαίνει η ερώτηση `?- not(p(X))` και τι επιστρέφει; Τι επιστρέφουν οι ερωτήσεις `?- not(p(a))`, `?- not(p(d))`, αλλά και οι `?- not(not(p(b)))`, `?- not(not(p(e)))` και `?- not(not(p(X)))`; Μπορούμε να ισχυριστούμε ότι δύο αρνήσεις στην Prolog ισοδυναμούν με μία κατάφαση;

   (β) Ορίστε σε Prolog ένα κατηγόρημα `combs/3` το οποίο όταν καλείται σαν `combs(M,L1,L2)` να επιστρέφει στο `L2` όλους τους συνδυασμούς, μέσω οπισθοδρόμησης, των στοιχείων της λίστας `L1` ανά `M`. Για παράδειγμα:

   ```
   ?- combs(2,[a,b,c,d],L).
   L = [a, b]     -> ;
   L = [a, c]     -> ;
   L = [a, d]     -> ;
   L = [b, c]     -> ;
   L = [b, d]     -> ;
   L = [c, d]
   yes.
   ```

4. (α) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το λογικό πρόγραμμα του διπλανού σχήματος; Γιατί οι ερμηνείες
   $I_1 = \{$`p(d), q(a), q(b), r(b), r(c), s(d)`$\}$
   $I_2 = \{$`p(b), p(d), q(a), q(b), r(b), r(d), s(d)`$\}$
   $I_3 = \{$`q(a), q(b), r(c), r(d), s(d)`$\}$
   $I_4 = \{$`p(b), p(d), q(a), q(b), q(c), r(b), r(c), r(d), s(d)`$\}$
   $I_5 = \{$`p(b), q(a), q(b), r(b), r(c), r(d), s(a), s(d)`$\}$
   δεν είναι μοντέλα Herbrand για το πρόγραμμα; Ποιο είναι το ελάχιστο μοντέλο Herbrand; Με ποιο τρόπο το κατασκευάσατε;

   ```
   p(X) :- q(X),
           r(X).
   r(X) :- s(X).
   p(X) :- s(X).
   q(a).
   q(b).
   r(b).
   r(c).
   s(d).
   ```

   (β) Σε ποια κατηγορία προβλημάτων εφαρμόζονται οι μέθοδοι "*γέννα-και-δοκίμασε*" και "*περιόρισε-και-γέννα*"; Πού διαφέρουν οι δύο μέθοδοι, πότε θα επέλεγε κάποιος τη μία και πότε την άλλη για την αντιμετώπιση ενός προβλήματος; Υπάρχει κάποιο πρόβλημα στα θέματα που προηγήθηκαν για το οποίο χρησιμοποιήσατε (ή θα χρησιμοποιούσατε) κάποια από αυτές τις μεθόδους και ποιο;

   (γ) Τι είναι κέλυφος εμπείρου συστήματος και από τι αποτελείται; Τι είναι η παροχή εξηγήσεων από ένα κέλυφος εμπείρου συστήματος και πόσο εύκολη ή δύσκολη είναι η υποστήριξή της;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Εξετάσεις Α' Περιόδου 1997

1. (α) Μερικές φορές, στο σώμα ενός κανόνα (ή σε μία ερώτηση) Prolog, θέλοντας κάποιος να ελέγξει αν ένα στοιχείο περιέχεται σε μία λίστα και, στην περίπτωση αυτή, να το διαγράψει απ'αυτήν, γράφει

   ```
   ..... :- ....., member(X,L1), delete(X,L1,L2), .....
   ```

   όπου τα `member/2` και `delete/3` είναι τα γνωστά κατηγορήματα. Μπορείτε να σχολιάσετε αυτό το τμήμα προγράμματος; Είναι "σωστός", "λάθος", "καλός", "κακός" προγραμματισμός σε Prolog και γιατί;

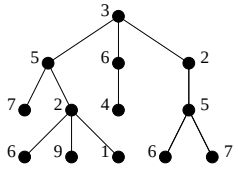
   (β) Υλοποιήστε σε Prolog τη συμπλήρωση του "μαγικού" τετραγώνου 4×4 του διπλανού σχήματος, τοποθετώντας στα κενά τετραγωνίδια τους αριθμούς από το 1 έως το 16 που δεν έχουν ήδη τοποθετηθεί, έτσι ώστε σε κάθε γραμμή, στήλη και κύρια διαγώνιο, το άθροισμα των αριθμών να είναι 34. Πόσο αποδοτικό πιστεύετε ότι είναι το πρόγραμμα που γράψατε; Μπορείτε να προτείνετε πιθανές τροποποιήσεις του που θα βελτίωναν την απόδοσή του;

7	9		
6			

2. Ένας οργανισμός έχει ιεραρχική δομή αποτελούμενος από διοικητικές μονάδες διαφόρων επιπέδων, με την έννοια ότι κάθε μονάδα έχει υπό την εποπτεία της έναν αριθμό μονάδων του αμέσως χαμηλότερου επιπέδου (εκτός βέβαια από αυτές του τελευταίου επιπέδου). Για παράδειγμα, μία δημόσια υπηρεσία μπορεί να αποτελείται από διευθύνσεις, αυτές, με τη σειρά τους, από υποδιευθύνσεις, αυτές από τμήματα κ.ο.κ. Στον οργανισμό εργάζονται υπάλληλοι, ορισμένοι από τους οποίους είναι προϊστάμενοι των διοικητικών μονάδων, για παράδειγμα οι διευθυντές, οι υποδιευθυντές, οι τμηματάρχες κ.λ.π. Φυσικά, υπάρχουν και απλοί υπάλληλοι οι οποίοι εργάζονται για κάποια συγκεκριμένη διοικητική μονάδα (οποιουδήποτε επιπέδου) ο καθένας. Κάθε υπάλληλος είναι γνωστός με το ονοματεπώνυμό του, έχει προσληφθεί μία συγκεκριμένη ημερομηνία και παίρνει συγκεκριμένο μισθό. Προτείνετε μία κωδικοποίηση του κόσμου αυτού σε Prolog, δίνοντας τις προδιαγραφές των απαιτούμενων, κατά τη γνώμη σας, κατηγορημάτων. Στη συνέχεια, ορίστε κατάλληλους κανόνες Prolog για: **α)** τον έλεγχο αν ένας υπάλληλος προΐσταται κάποιου άλλου υπαλλήλου (δηλαδή, ο δεύτερος εργάζεται σε μονάδα που είτε είναι η ίδια είτε είναι τελικά υπομονάδα αυτής που προΐσταται ο πρώτος), **β)** την εύρεση του συνολικού μισθού των υπαλλήλων δεδομένης μονάδας, και **γ)** τον έλεγχο αν σε κάποια μονάδα υπάρχει υπάλληλος που η χρονολογία πρόσληψής του είναι προγενέστερη αυτής του προϊσταμένου της μονάδας.

3. (α) Ορίστε σε Prolog το κατηγόρημα `rem_dupls/2` έτσι ώστε όταν υποβάλλεται η ερώτηση `rem_dupls(L1,L2)` με δεδομένη λίστα `L1`, να επιστρέφει στο `L2` τη λίστα που προκύπτει αν από την `L1` διαγραφούν όλες οι πολλαπλές εμφανίσεις στοιχείων της. Για παράδειγμα, η ερώτηση `?-rem_dupls([a,b,c,b,a,c,d,b],L)` να επιστρέφει στο `L` τη λίστα `[a,c,d,b]` (όχι απαραίτητα με αυτήν τη σειρά στα στοιχεία της).

   (β) Προτείνετε έναν τρόπο αναπαράστασης δέντρων (όχι απαραίτητα δυαδικών), όπως αυτό του διπλανού σχήματος, σαν σύνθετους όρους Prolog. Επίσης, ορίστε ένα κατηγόρημα Prolog το οποίο να αληθεύει όταν ένα στοιχείο αποτελεί κόμβο ενός δέντρου. Το κατηγόρημα αυτό να μπορεί να χρησιμοποιηθεί είτε για να ελεγχθεί αν δεδομένο στοιχείο ανήκει στο δέντρο είτε για να γεννηθούν μέσω οπισθοδρόμησης όλα τα στοιχεία του δέντρου.



4. (α) Ποιο είναι το σύμπαν Herbrand και ποια η βάση Herbrand για το λογικό πρόγραμμα του διπλανού σχήματος; Γιατί οι ερμηνείες $I_1 = \{\texttt{p(a,a)}, \texttt{p(a,b)}, \texttt{q(a,b)}\}$, $I_2 = \{\texttt{p(a,a)}, \texttt{p(a,b)}, \texttt{p(b,b)}\}$, $I_3 = \{\texttt{p(a,b)}, \texttt{p(b,b)}, \texttt{q(a,b)}\}$ και $I_4 = \{\texttt{p(a,a)}, \texttt{p(a,b)}, \texttt{p(b,b)}, \texttt{q(a,a)}, \texttt{q(a,b)}, \texttt{q(b,a)}\}$ δεν είναι μοντέλα Herbrand για το πρόγραμμα; Μπορείτε να κατασκευάσετε το ελάχιστο μοντέλο Herbrand;

```
p(X,Y) :- q(X,Y).
p(X,X) :- p(X,Y).
p(Y,Y) :- p(X,Y).
q(a,b).
```

   (β) Τι είναι η εμπρόσθια συλλογιστική στα έμπειρα συστήματα; Σε ποιες περιπτώσεις είναι χρήσιμη, ή ίσως και επιβάλλεται, η εφαρμογή της; Υπάρχουν προβλήματα που, κατά τη γνώμη σας, παρουσιάζει; Πιστεύετε ότι διευκολύνεται η υλοποίησή της σε περιβάλλον λογικού προγραμματισμού και γιατί;

   (γ) Ποιες περιπτώσεις καθαρής παραλληλίας σε συστήματα λογικού προγραμματισμού γνωρίζετε; Πώς νοείται, σε κάθε περίπτωση, η εκμετάλλευση της παραλληλίας; Πότε πιστεύετε ότι υπάρχει ουσιαστικό όφελος;

ΛΟΓΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ
Εξετάσεις Β' Περιόδου 1997

1. (α) Ως γνωστόν, η ακολουθία Fibonacci ορίζεται από τον αναδρομικό τύπο $a_{n+2} = a_{n+1} + a_n$ $\forall n \geq 1$ με $a_1 = 1$, $a_2 = 1$. Οι παρακάτω ορισμοί των κατηγορημάτων `fiba/2` και `fibb/2` είναι δύο σωστά προγράμματα για τον υπολογισμό του `N`-οστού όρου `F` της ακολουθίας Fibonacci.

```
fiba(1,1).
fiba(2,1).
fiba(N,F) :-
   N > 2,
   N1 is N-1,
   fiba(N1,F1),
   N2 is N-2,
   fiba(N2,F2),
   F is F1+F2.
```

```
fibb(N,F) :-
   ff(2,N,1,1,F).
ff(M,N,F1,F2,F2) :-
   M >= N.
ff(M,N,F1,F2,F) :-
   M < N,
   NextM is M+1,
   NextF2 is F1+F2,
   ff(NextM,N,F2,NextF2,F).
```

   Σχολιάστε πού πλεονεκτεί και πού μειονεκτεί το ένα έναντι του άλλου.

   (β) Σε μία σκακιέρα (πλαίσιο 8×8) ένα άλογο μπορεί να κινηθεί 2 τετράγωνα κατακόρυφα ή οριζόντια και, στη συνέχεια, 1 τετράγωνο σε κάθετη διεύθυνση, δηλαδή οριζόντια ή κατακόρυφα, αντίστοιχα. Γράψτε ένα πρόγραμμα Prolog που να ελέγχει αν δεδομένη λίστα από τετράγωνα της σκακιέρας αποτελεί μία νόμιμη ακολουθία θέσεων στην κίνηση ενός αλόγου.

2. Στις 3 τάξεις ενός λυκείου διδάσκουν 4 καθηγητές, ο (A)ndy, ο (B)ill, ο (C)arl και ο (D)ave. Κάθε Δευτέρα ο A κάνει 2 ώρες μάθημα στην πρώτη τάξη, 1 ώρα στη δευτέρα και 1 ώρα στη τρίτη, ο B κάνει 1, 3 και 2 ώρες, ο C 1, 1 και 1 ώρα και ο D 2, 1 και 2 ώρες αντίστοιχα. Εύκολα μπορούμε να δούμε ότι κάθε τάξη έχει 6 ώρες μάθημα τη Δευτέρα. Γράψτε ένα απλό, αλλά όχι κατ'ανάγκη αποδοτικό, πρόγραμμα Prolog, το οποίο να κατασκευάζει ένα ωρολόγιο πρόγραμμα του λυκείου για την ημέρα αυτή. Για παράδειγμα, ορίστε το κατηγόρημα `timetable/3` με τη συμπεριφορά

```
?- timetable(C1,C2,C3).
C1 = [a,a,b,c,d,d]
C2 = [b,b,a,d,b,c]
C3 = [c,d,d,b,a,b]
```

   όπου η τιμή της μεταβλητής `C1` σημαίνει ότι η πρώτη τάξη έχει την 1η και 2η ώρα μάθημα με τον A, την 3η με τον B, την 4η με τον C και την 5η και 6η με τον D. Αντίστοιχα ισχύουν και για τις άλλες δύο τάξεις. Φυσικά, μπορείτε να παρατηρήσετε ότι κανένας καθηγητής δεν κάνει την ίδια ώρα μάθημα σε περισσότερες από μία τάξη.

3. (α) Ορίστε σε Prolog το κατηγόρημα `rem_ind/3` έτσι ώστε όταν υποβάλλεται η ερώτηση `rem_ind(Is,Xs,Rs)`, να επιστρέφει στη λίστα `Rs` τα στοιχεία που απομένουν όταν από τη λίστα `Xs` διαγραφούν εκείνα που βρίσκονται στις θέσεις που καθορίζονται από τους δείκτες που περιέχονται στη λίστα `Is`. Για παράδειγμα, η ερώτηση

```
?- rem_ind([1,3,4,6],[a,b,c,d,e,f,g],L).
```

   να επιστρέφει στο `L` τη λίστα `[b,e,g]`.

(β) Ποια είναι η λειτουργία του ενσωματωμένου κατηγορήματος ! στην Prolog; Τι ευκολίες προσφέρει και τι κινδύνους παρουσιάζει; Χρησιμοποιώντας τα κατηγορήματα

$is\_bird(x)$:	το x είναι πουλί
$is\_eagle(x)$:	το x είναι αετός
$is\_ostrich(x)$:	το x είναι στρουθοκάμηλος
$flies(x)$:	το x πετάει

περιγράψτε σε Prolog τη γνώση: "*Οι αετοί είναι πουλιά. Οι στρουθοκάμηλοι είναι πουλιά. Ο Bob είναι αετός. Ο Fred είναι στρουθοκάμηλος. Τα πουλιά, εκτός από τις στρουθοκαμήλους, πετούν.*"

4. (α) Ποιες μεθόδους μελέτης της σημασίας των λογικών προγραμμάτων γνωρίζετε; Ποια είναι η γενική φιλοσοφία της κάθε μίας;

(β) Γιατί, κατά τη γνώμη σας, είναι απαραίτητο τα έμπειρα συστήματα να έχουν τη δυνατότητα χειρισμού αβέβαιης πληροφορίας; Προτείνετε ένα απλό και εύλογο μοντέλο γι'αυτόν το σκοπό και δώστε την υλοποίησή του σε Prolog.

(γ) Πού είναι χρήσιμη η μεθοδολογία του λογικού προγραμματισμού με περιορισμούς; Θα μπορούσε να αντιμετωπισθεί το πρόβλημα του 2ου θέματος με αυτόν τον τρόπο; Δώστε ένα συνοπτικό πλάνο αντιμετώπισής του.