ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

# Λογικός Προγραμματισμός

Ασκήσεις

Παναγιώτης Σταματόπουλος

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

# Περιεχόμενα

# 1. Ασκήσεις "Λογικού Προγραμματισμού" Ακαδημαϊκού Έτους 2000-01

## 1.1 Άσκηση 1

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 τα κατηγορήματα `matr_transp/2, matr_mult/3` και `matr_det/1` έτσι ώστε το `matr_transp(M1,M2)` να επιστρέφει στο M2 τον ανάστροφο πίνακα (δηλαδή αυτόν που προκύπτει με εναλλαγή γραμμών και στηλών) του πίνακα M1, το `matr_mult(M1,M2,M3)` να επιστρέφει στο M3 το γινόμενο των πινάκων M1 και M2 και το `matr_det(M,D)` να επιστρέφει στο D την ορίζουσα του πίνακα M. Παραδείγματα:

```
?- 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]]


?- 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]]


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

D = 51
```

## 1.2 Άσκηση 2

Ορίστε σε Prolog το κατηγόρημα `whirl/2` το οποίο όταν καλείται με ορίσματα δύο θετικούς ακεραίους A και B, δηλαδή `whirl(A,B)`, να εκτυπώνει στην οθόνη μία ορθογώνια διάταξη πλάτους `B` και ύψους `A` των αριθμών από το 1 έως το `A*B` με τον ελικοειδή τρόπο που φαίνεται στη συνέχεια. Οι στήλες πρέπει να έχουν στοιχηθεί δεξιά και να καταλαμβάνουν *N+1* θέσεις, όπου *N* είναι ο αριθμός των ψηφίων στη δεκαδική αναπαράσταση του `A*B`.

```
?- whirl(12, 15).

   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  50  51  52  53  54  55  56  57  58  59  60  61  62  63  16
  49  92  93  94  95  96  97  98  99 100 101 102 103  64  17
  48  91 126 127 128 129 130 131 132 133 134 135 104  65  18
  47  90 125 152 153 154 155 156 157 158 159 136 105  66  19
  46  89 124 151 170 171 172 173 174 175 160 137 106  67  20
  45  88 123 150 169 180 179 178 177 176 161 138 107  68  21
  44  87 122 149 168 167 166 165 164 163 162 139 108  69  22
  43  86 121 148 147 146 145 144 143 142 141 140 109  70  23
```

```
  42  85 120 119 118 117 116 115 114 113 112 111 110  71  24
  41  84  83  82  81  80  79  78  77  76  75  74  73  72  25
  40  39  38  37  36  35  34  33  32  31  30  29  28  27  26
yes
```

## 1.3 Άσκηση 3

Να ορισθεί ένα 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`. Για παράδειγμα:

```
?- cycle(3,4,C).
C = [0]      %  Αφού 3/4 = 0.25000...
 ?- cycle(4,3,C). C = [3]      %  Αφού 4/3 = 1.3333...
 ?- cycle(1,7,C). C = [1,4,2,8,5,7]
             %  Αφού 1/7 = 0.142857142857...

?- cycle(129,55,C).
C = [4,5]    %  Αφού 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]
             %  Αφού 13/43 = 0.302325581395348837209302325...
```

## 1.4 Άσκηση 4

Έστω ένας κατευθυνόμενος γράφος που περιγράφεται σε Prolog με το κατηγόρημα `arrow/2`. Για παράδειγμα:

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

Το ζητούμενο είναι να ορισθεί ένα κατηγόρημα `loops/1` που να επιστρέφει μία λίστα που περιλαμβάνει όλους τους "ελάχιστους" κύκλους του γράφου. Ένας κύκλος παριστάνεται από μία λίστα που περιέχει όλους τους κόμβους του κύκλου με τη σειρά που συνδέονται μέσω του κατηγορήματος `arrow/2` στο γράφο. Επίσης, αρχικός κόμβος του κύκλου μπορεί να είναι οποιοσδήποτε κόμβος του, αλλά αυτός συμπίπτει και με τον τελικό κόμβο. Ένας κύκλος είναι "ελάχιστος" όταν δεν περιλαμβάνει άλλους κύκλους. Το `loops/1` πρέπει να επιστρέφει όλους τους

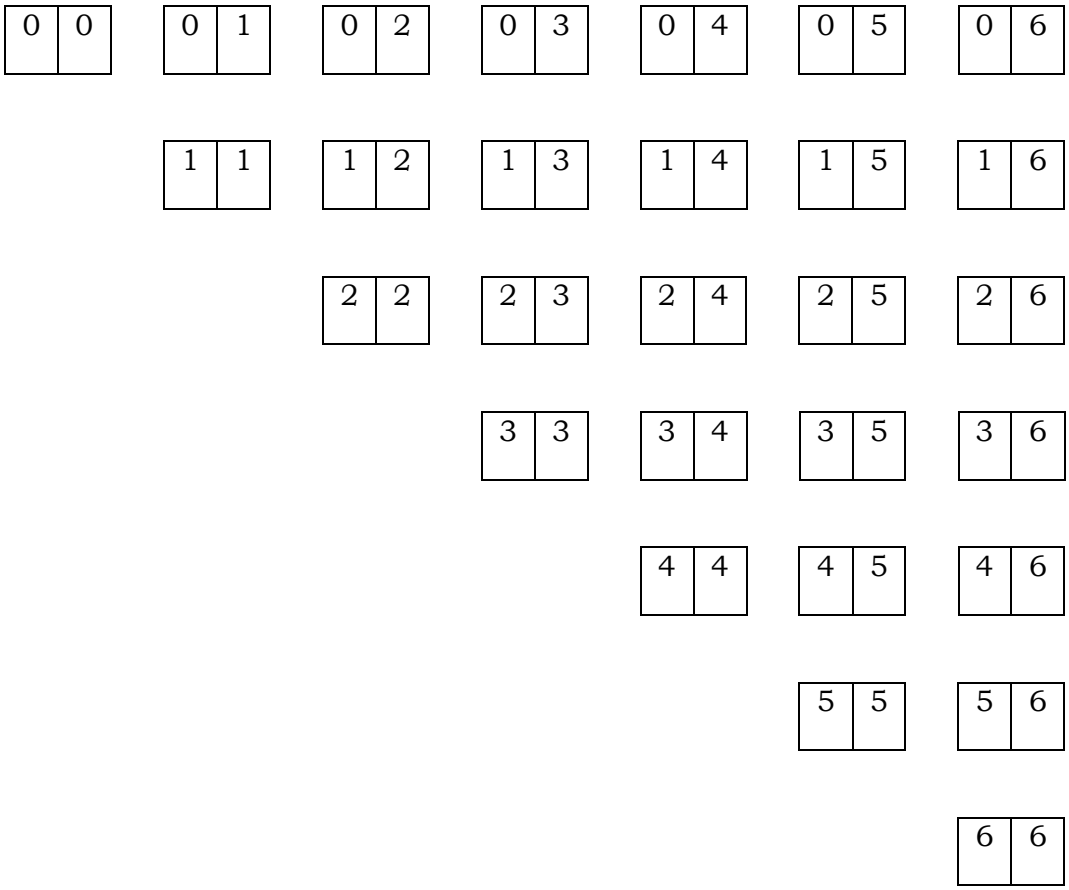
"ελάχιστους" κύκλους, αλλά μία φορά τον καθένα. Για παράδειγμα, στην περίπτωση του προηγούμενου γράφου, η σωστή λύση είναι μία από τις `[[c,c],[a,d,a]]` ή `[[c,c],[d,a,d]]` ή `[[a,d,a],[c,c]]` ή `[[d,a,d],[c,c]]`. Δοκιμάστε το πρόγραμμά σας και για πιο πολύπλοκους γράφους, όπως:

```
arrow(a,b).  arrow(b,c).  arrow(b,j).  arrow(c,f).

arrow(c,j).  arrow(d,c).  arrow(d,f).  arrow(e,d).

arrow(f,g).  arrow(g,e).  arrow(g,h).  arrow(g,i).

arrow(h,f).  arrow(i,h).  arrow(i,j).  arrow(j,a).
```

Αυτός ο γράφος περιλαμβάνει 7 "ελάχιστους" κύκλους.

## 1.5 Άσκηση 5

Έστω ότι έχουμε στη διάθεσή μας τις παρακάτω 28 πλάκες "ντόμινο":




Γράψτε ένα πρόγραμμα Prolog που να μας πληροφορεί πώς πρέπει να τις τοποθετήσουμε σ' ένα πλαίσιο 7x8, έτσι ώστε η τελική διάταξη να είναι η εξής:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | 1 | 2 | 6 | 6 | 1 | 2 | 2 |
| 3 | 4 | 1 | 5 | 3 | 0 | 3 | 6 |
| 5 | 6 | 6 | 1 | 2 | 4 | 5 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | 6 | 4 | 1 | 3 | 3 | 0 | 0 |
| 6 | 1 | 0 | 6 | 3 | 2 | 4 | 0 |
| 4 | 1 | 5 | 2 | 4 | 3 | 5 | 5 |
| 4 | 1 | 0 | 2 | 4 | 5 | 2 | 0 |

Σημειώστε ότι ένα "ντόμινο" μπορεί να τοποθετηθεί στο πλαίσιο με οποιοδήποτε από τους τέσσερις δυνατούς τρόπους. Δηλαδή το "ντόμινο"

| | |
|---|---|
| 2 | 5 |

μπορεί να τοποθετηθεί με κάποιον από τους εξής τρόπους:

| | |
|---|---|
| 2 | 5 |

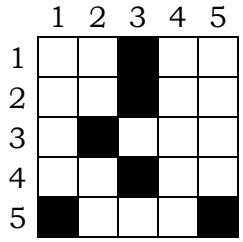
| |
|---|
| 2 |
| 5 |

| | |
|---|---|
| 5 | 2 |

| |
|---|
| 5 |
| 2 |

## 1.6 Άσκηση 6

Ένα σταυρόλεξο τετραγώνου σχήματος μπορεί να ορισθεί από τη διάστασή του και τις συντεταγμένες των μαύρων θέσεων. Για παράδειγμα, το σταυρόλεξο




μπορεί να περιγραφεί από τα γεγονότα Prolog:

```
dimension(5).

black(1,3).
black(2,3).
black(3,2).
black(4,3).
black(5,1).
black(5,5).
```

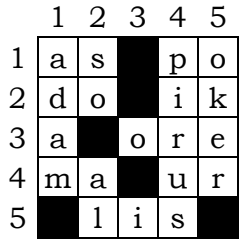
Δεδομένου και ενός γεγονότος με κατηγόρημα `words/1` μέσω του οποίου δίνονται οι λέξεις που πρέπει να τοποθετηθούν στο σταυρόλεξο, ορίστε ένα κατηγόρημα `crossword/1` που να επιστρέφει τη λίστα των διαθεσίμων λέξεων με τη σειρά που αυτές μπορούν να τοποθετηθούν στο σταυρόλεξο, πρώτα οι οριζόντιες και μετά οι κάθετες. Για παράδειγμα, αν δίνεται και το

```
words([do,ore,ma,lis,ur,as,po,so,pirus,oker,al,adam,ik]).
```

η σωστή απάντηση στο προηγούμενο σταυρόλεξο, δηλαδή αυτή που πρέπει να επιστρέψει το `crossword/1`, είναι η

```
[as,po,do,ik,ore,ma,ur,lis,adam,so,al,pirus,oker]
```

που αντιπροσωπεύει την εξής λύση:




Αν τυχόν υπάρχουν περισσότερες της μίας λύσεις, πρέπει το `crossword/1` να τις υπολογίζει όλες μέσω οπισθοδρόμησης. Φυσικά, αν δεν υπάρχει λύση, πρέπει να αποτυγχάνει. Θεωρείται γνωστό, επίσης, ότι στα σταυρόλεξα δεν λογίζονται σαν λέξεις αυτές του ενός γράμματος. Δοκιμάστε το πρόγραμμά σας και σε μεγαλύτερα σταυρόλεξα, όπως:

```
dimension(20).

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

words([ados,aere,aleser,alliee,apis,apivor,aria,arno,
       attire,attristee,ave,aviser,blocage,blonde,bol,ca,
       casse,caverne,client,colibri,cou,cultiver,dada,
       deite,demi,doter,eau,eclos,ecrase,ecrin,ecuri,
       eden,egri,elevation,emier,envol,eole,eolie,epte,
       ese,esope,et,etendard,etier,etoilee,etroitesse,
       europeenne,evier,evoe,feu,fraisier,gambader,gre,
       hesperides,id,ille,ils,in,ios,ir,isar,isole,lace,
       lave,le,les,levrette,lie,lieue,lippe,lo,mer,mulet,
       nato,nep,nie,noel,norme,notaire,ohe,ointe,ole,olt,
```

```
        orienter,oui,peler,pitre,puissante,rale,ratier,
        rose,savonner,soir,tavele,tirer,tresorier,trou,
        tulle,usa,usurper,utilise,va,valse,vilipe,voleter]).
```

## 1.7 Άσκηση 7

Να ορισθεί ένα κατηγόρημα `simplify/2` το οποίο να δέχεται στο πρώτο όρισμά του μία αλγεβρική έκφραση που μπορεί να περιλαμβάνει γινόμενα, αθροίσματα και διαφορές μεταξύ πολυωνύμων μίας μεταβλητής (έστω x) και να μετασχηματίζει αυτήν την έκφραση στο απλούστερο δυνατό πολυώνυμο που να το επιστρέφει στο δεύτερο όρισμα. Για παράδειγμα:

```
?- simplify((2*x^2-3*x+2)*(3*x-4)-(4*x^2+18*x-7),P).
P = 6*x^3-21*x^2-1
```

## 1.8 Άσκηση 8

Έστω ότι δίνεται η παρακάτω Prolog βάση δεδομένων:

```
borders(belgium,       france).
borders(belgium,       germany).
borders(belgium,       luxembourg).
borders(belgium,       netherlands).
borders(denmark,       germany).
borders(france,        belgium).
borders(france,        germany).
borders(france,        italy).
borders(france,        luxembourg).
borders(france,        spain).
borders(germany,       belgium).
borders(germany,       denmark).
borders(germany,       france).
borders(germany,       luxembourg).
borders(germany,       netherlands).
borders(great_britain, ireland).
borders(ireland,       great_britain).
borders(italy,         france).
borders(luxembourg,    belgium).
borders(luxembourg,    france).
borders(luxembourg,    germany).
borders(netherlands,   belgium).
borders(netherlands,   germany).
borders(portugal,      spain).
borders(spain,         france).
borders(spain,         portugal).

flows(achelous, greece).
flows(aliakmon, greece).
flows(danube,   germany).
flows(ebro,     spain).
flows(elbe,     germany).
flows(garonne,  france).
flows(garonne,  spain).
flows(loire,    france).
flows(maas,     belgium).
flows(maas,     france).
flows(maas,     netherlands).
flows(po,       italy).
flows(rhine,    france).
flows(rhine,    germany).
```

```
flows(rhine,    netherlands).
flows(rhone,    france).
flows(seine,    france).
flows(tagus,    portugal).
flows(tagus,    spain).
flows(thames,   great_britain).
flows(tiber,    italy).

belongs(brussels,   belgium).
belongs(copenhagen, denmark).
belongs(lyon,       france).
belongs(marseille,  france).
belongs(paris,      france).
belongs(berlin,     germany).
belongs(hamburg,    germany).
belongs(munich,     germany).
belongs(birmingham, great_britain).
belongs(glasgow,    great_britain).
belongs(london,     great_britain).
belongs(athens,     greece).
belongs(salonika,   greece).
belongs(dublin,     ireland).
belongs(milan,      italy).
belongs(naples,     italy).
belongs(rome,       italy).
belongs(luxembourg, luxembourg).
belongs(amsterdam,  netherlands).
belongs(hague,      netherlands).
belongs(rotterdam,  netherlands).
belongs(lisbon,     portugal).
belongs(madrid,     spain).
belongs(barcelona,  spain).
```

Τα κατηγορήματα `borders/2`, `flows/2` και `belongs/2` έχουν την προφανή σημασία. Ορίστε ένα κατηγόρημα `answer/2` που να είναι σε θέση να βρίσκει τις κατάλληλες απαντήσεις σε ερωτήσεις που είναι διατυπωμένες σε απλή φυσική Ελληνική γλώσσα[1]. Παραδείγματα:

```
?- answer('poies xwres synoreuoun me thn Ispania;',L).
L = [france,portugal]

?- answer('me poies xwres synoreuei h Gallia;',L).
L = [belgium,germany,italy,luxembourg,spain]

?- answer('poia potamia diarreoun thn Ellada;',L).
L = [achelous,aliakmon]

?- answer('poies xwres diarreei o Rhnos;',L).
L = [france,germany,netherlands]

?- answer('poies poleis briskontai sthn Ollandia;',L).
L = [amsterdam,hague,rotterdam]

?- answer('se poia xwra brisketai to Milano;',L).
```

[1] Θα βοηθούσε ιδιαίτερα αν χρησιμοποιούσατε τη δυνατότητα που προσφέρουν διάφορα συστήματα Prolog (LPA Prolog, ECL$^{i}$PS$^{e}$, κλπ.) για την περιγραφή γλωσσών (φυσικών και άλλων) μέσω μίας γραμματικής οριστικών προτάσεων (definite clause grammar). Ανατρέξτε στο on-line help ή ζητήστε μου τα εγχειρίδια για περισσότερες λεπτομέρειες.

```
L = [italy]

?- answer('me poies xwres tis opoies diarreei o Tagos
          synoreuei h Gallia;',L).
L = [spain]

?- answer('poies xwres oi opoies synoreuoun me to Louxembourgo
          synoreuoun me thn Italia;',L).
L = [france]

?- answer('poia xwra sthn opoia brisketai h Glaskwbh
          diarreei o Tameshs;',L).
L = [great_britain]

?- answer('poio potami to opoio diarreei to Belgio
          diarreei thn Ollandia;',L).
L = [maas]
```

## 1.9 Άσκηση 9

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 στην 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 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] -> ;

C1 = ......................
.........................
```

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. Μπορείτε να γενικεύσετε το πρόγραμμά σας έτσι ώστε να δουλεύει για παραμετρικό αριθμό καθηγητών και τάξεων, καθώς και για παραμετρικές αναθέσεις μαθημάτων; Δηλαδή, μπορείτε να γράψετε το κατηγόρημα `timetable` έτσι ώστε να παίρνει στα ορίσματά του όποιες πληροφορίες χρειάζεται για καθηγητές, τάξεις και για το πλήθος των ωρών που έχει κάθε καθηγητής μάθημα σε κάθε τάξη; Μπορείτε να γενικεύσετε το πρόγραμμα περισσότερο ώστε να καλύπτει όλες τις ημέρες διδασκαλίας στην εβδομάδα;

## 1.10 Άσκηση 10

Μία εκδοχή του λεγόμενου job-shop προβλήματος είναι η εξής: Έχουμε να εκτελέσουμε μία σειρά από έργα `j1`, `j2`, `j3` ..., ανεξάρτητα το ένα από το άλλο, καθένα από τα οποία αποτελείται από έναν αριθμό από εργασίες `t11`, `t12` ..., `t21`, `t22`, ..., `t31`, `t32`, ... οι οποίες πρέπει να εκτελεσθούν με τη δεδομένη σειρά για κάθε έργο. Κάθε εργασία μπορεί να εκτελεσθεί σε συγκεκριμένο τύπο μηχανής και χρειάζεται δεδομένο χρόνο για να έλθει σε πέρας. Επίσης είναι γνωστό το πλήθος των μηχανών που έχουμε διαθέσιμες από κάθε τύπο. Αυτό που θέλουμε να βρούμε είναι πότε και σε ποιες μηχανές πρέπει να εκτελέσουμε τις εργασίες έτσι ώστε όλα τα έργα να έχουν τελειώσει πριν από δεδομένη προθεσμία. Ένα συγκεκριμένο στιγμιότυπο αυτού του προβλήματος δίνεται με τα παρακάτω γεγονότα Prolog:

```
job(j1,[t11,t12]).
```

```
job(j2,[t21,t22,t23]).
job(j3,[t31]).
job(j4,[t41,t42]).

task(t11,m1,2).
task(t12,m2,6).
task(t21,m2,5).
task(t22,m1,3).
task(t23,m2,3).
task(t31,m2,4).
task(t41,m1,5).
task(t42,m2,2).

machine(m1,1).
machine(m2,2).

deadline(14).
```

Τα γεγονότα `job/3` περιγράφουν τα έργα που πρέπει να εκτελέσουμε και τις εργασίες από τις οποίες αποτελούνται, κάθε γεγονός `task/3` δίνει, για μία εργασία, σε τι τύπου μηχανή πρέπει να εκτελεσθεί και πόσες μονάδες χρόνου απαιτούνται για την εκτέλεσή της, τα γεγονότα `machine/2` μας πληροφορούν πόσες μηχανές έχουμε διαθέσιμες από κάθε τύπο και τέλος το κατηγόρημα `deadline/1` δίνει την προθεσμία (σε μονάδες χρόνου) μέσα στην οποία πρέπει να έχουν εκτελεσθεί όλα τα έργα. Ορίστε σε Prolog το κατηγόρημα `job_shop/1`, το οποίο να επιστρέφει το πρόγραμμα εκτέλεσης των εργασιών στις μηχανές. Παράδειγμα:

```
?- job_shop(S).
S = [execs(m1,[t(t11,0,2),t(t41,2,7),t(t22,7,10)]),
     execs(m2,[t(t12,2,8),t(t42,8,10),t(t23,10,13)]),
     execs(m2,[t(t21,0,5),t(t31,5,9)])]                  -> ;

S = ....................................................
....................................................
```

Η παραπάνω απάντηση σημαίνει ότι στη μηχανή τύπου `m1` θα εκτελεσθούν από τη χρονική στιγμή 0 έως τη 2 η εργασία `t11`, από τη χρονική στιγμή 2 έως την 7 η εργασία `t41` και από τη χρονική στιγμή 7 έως τη 10 η εργασία `t22`. Μετά ακολουθούν τα προγράμματα εργασίας για τις δύο μηχανές τύπου `m2`.

Μία δεύτερη εκδοχή του job-shop προβλήματος είναι αυτή στην οποία για κάθε εργασία έχουμε και ένα δεδομένο πλήθος ατόμων που πρέπει να εργασθούν στη μηχανή που θα τη φέρει σε πέρας. Δηλαδή, τώρα έχουμε το κατηγόρημα `task/4` (αντί για `task/3`), όπου το τελευταίο επιπλέον όρισμα δίνει και το πλήθος αυτό των ατόμων. Τα γεγονότα τώρα είναι:

```
task(t11,m1,2,3).
task(t12,m2,6,2).
task(t21,m2,5,2).
task(t22,m1,3,3).
task(t23,m2,3,2).
task(t31,m2,4,2).
task(t41,m1,5,4).
task(t42,m2,2,1).
```

Επίσης έχουμε δεδομένο και το προσωπικό που υπάρχει διαθέσιμο ανά πάσα χρονική στιγμή, το οποίο ορίζεται με το γεγονός `staff/1`, δηλαδή:

```
staff(6).
```

Υποτίθεται ότι κάθε εργαζόμενος μπορεί να δουλέψει σε οποιαδήποτε μηχανή για οποιοδήποτε εργασία.

Ορίστε σε Prolog το κατηγόρημα `job_shop_with_manpower/1`, το οποίο να λύνει και αυτήν την εκδοχή του job-shop προβλήματος. Παράδειγμα εκτέλεσης:

```
?- job_shop_with_manpower(S).
S = [execs(m1,[t(t11,0,2),t(t41,2,7),t(t22,7,10)]),
     execs(m2,[t(t42,7,9),t(t31,10,14)]),
     execs(m2,[t(t21,0,5),t(t12,5,11),t(t23,11,14)])]   -> ;

S = .....................................................
.....................................................
```

# Σημειώματα

## Σημείωμα Ιστορικού Εκδόσεων Έργου

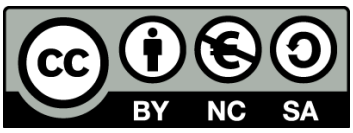
Το παρόν έργο αποτελεί την έκδοση 1.0.

## Σημείωμα Αναφοράς



## Σημείωμα Αδειοδότησης








## Διατήρηση Σημειωμάτων

- Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:
- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων

- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

## Σημείωμα Χρήσης Έργων Τρίτων

Το Έργο αυτό κάνει χρήση των ακόλουθων έργων:

Εικόνες/Σχήματα/Διαγράμματα/Φωτογραφίες

# Χρηματοδότηση


- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.