Λογικός Προγραμματισμός με Περιορισμούς

Τμ. Εφαρμοσμένης Πληροφορικής

Εργασία 1 2016-2017

1. Sales!

Εκπτώσεις! Οι τιμές έπεσαν στο 1/3 της αρχικής τους! Έχετε παραγγείλει ταμπελάκια με τις παλαιές και νέες τιμές, αλλά δυστυχώς ο τυπογράφος σας έκανε 2 λάθη.

- Σας επέστρεψε τις ταμπέλες **με αύξουσα σειρά**, ενώ τις θέλατε ανά ζεύγη (παλαιά-νέα τιμή).
- Μπέρδεψε την παραγγελία σας με κάποια άλλη και σας έστειλε **μερικές παραπάνω**.

α) Να γράψετε ένα κατηγόρημα **split_the_labels(Labels,Correct,Waste)**, όπου **Labels** είναι η διατεταγμένη λίστα με τα ταμπελάκια που σας έστειλε ο τυπογράφος, **Correct** η λίστα με τα σωστά ταμπελάκια και σωστή ταξινόμηση, και **Waste** η λίστα με τα ταμπελάκια που περισσεύουν. Σωστή ταξινόμηση σημαίνει πρώτα η τιμή έκπτωσης και έπειτα η αρχική τιμή. Για παράδειγμα:

```
?- split_the_labels([10, 15, 20, 60], Correct, Waste).
Correct = [20, 60]
Waste = [10, 15]
Yes (0.00s cpu, solution 1, maybe more)

?- split_the_labels([5, 10, 15, 30, 20, 40, 60], Correct, Waste).
Correct = [5, 15, 10, 30, 20, 60]
Waste = [40]
Yes (0.00s cpu, solution 1, maybe more)

?- split_the_labels([5, 10, 15, 30, 20, 40, 60], Correct, Waste).
Correct = [5, 15, 10, 30, 20, 60]
Waste = [40]
Yes (0.00s cpu, solution 1, maybe more)
```

Το κατηγόρημα σας να επιστρέφει όλες τις δυνατές λύσεις. Για παράδειγμα:

```
?- split_the_labels([5, 15, 20, 45], Correct, Waste).
Correct = [5, 15]
Waste = [20, 45]
Yes (... more)

Correct = [15, 45]
Waste = [5, 20]
Yes (... more)

Correct = []
Waste = [5, 15, 20, 45]
Yes
```

β) Αν έχετε **Ν προϊόντα** στο κατάστημά σας, να γράψετε ένα κατηγόρημα **lowest_labels(Labels,N,L)**, το οποίο επιστρέφει τη λίστα με τις ταμπέλες L, που αντιστοιχούν σε αυτά τα Ν προϊόντα, με τις μικρότερες τιμές. Πρέπει να επιστρέφεται μόνο μια λύση. Για παράδειγμα:

```
?- lowest_labels([5, 10, 15, 30, 20, 40, 60], 1, L).
L = [5, 15]
Yes

?- lowest_labels([5, 10, 15, 30, 20, 40, 60], 2, L).
L = [5, 15, 10, 30]
Yes
```

2. Patterns Overlap

Η Alice, εργαζόμενη στην google, τακτοποιεί τα βιβλία της σε δύο κουτιά. Κάθε κουτί έχει μια ταμπέλα η οποία, και αν το τίτλος του βιβλίου ταιριάζει σε αυτή την ταμπέλα, τότε το βιβλίο μπαίνει στο αντίστοιχο κουτί. Οι ταμπέλες στα κουτιά αποτελούνται από τους χαρακτήρες του Αγγλικού αλφάβητου (πεζά μόνο) και το χαρακτήρα ***,** ο οποίος ταιριάζει με κανένα, έως και 4 χαρακτήρες. Για παράδειγμα τα βιβλία gonegirl και gonetomorrow μπορούν να μπουν στο κουτί με την ταμπέλα gone**, αλλά τα βιβλία thegonegirl, και gonewiththewind δεν μπορούν.

Τίτλος	**Ταμπέλα (pattern)**	**Εξήγηση**
gonegirl	gone**	πρώτο * χαρ girl δεύτερο * κανένας χαρακτήρας (ένα πιθανό ταίριασμα)
gonetomorrow	gone**	πρώτο * χαρ tomo δεύτερο * χαρ rrow
thegonegirl	gone**	Δεν ταιριάζει.

Το πρόβλημα της Alice είναι θέλει να ξέρει αν υπάρχει τίτλος, ο οποίος μπορεί να ταιριάξει και σε δύο διαφορετικά ταμπελάκια (patterns). Για παράδειγμα, τα ταμπελάκια:

shakes*e και s*speare ταιριάζουν και τα δύο με τον τίτλο shakespeare.

Να υλοποιήσετε ένα κατηγόρημα **patterns(Pat1,Pat2,Title)** το οποίο, δοθέντος δύο ταμπελών (patterns), εκφρασμένων σαν λίστες χαρακτήρων, πετυχαίνει όταν υπάρχει πιθανός τίτλος Title (ακολουθία χαρακτήρων σε λίστα), ο οποίος *ταιριάζει και στις δύο ταμπέλες*. Αν δεν υπάρχει, τότε το κατηγόρημα αποτυγχάνει. Για παράδειγμα:

```
?- patterns([s, h, a, k, e, s, *, e], [s, *, s, p, e, a, r, e], Title).
Title = [s, h, a, k, e, s, p, e, a, r, e]
Yes

?- patterns([s, h, a, k, e, s, *, e], [*, p, e, a, r, e], Title).
No

?- patterns([s, h, *, e], [*, p], Title).
No
```

Για την διευκόλυνσή σας, σας δίνονται έτοιμα ζεύγη patterns σε ένα αρχείο (δες τέλος εργασίας) όπως φαίνεται παρακάτω:

```
pat(0,[t,e,s,t],[t,e,s,t]).
pat(1,[t,*,s,t],[t,e,*,t]).
pat(2,[*,*,*,*],[i,t]).
pat(3,[s,h,a,k,e,s,*,e],[s,*,s,p,e,a,r,e]).
...
```

3. **London Metro**

Το μετρό του Λονδίνου αποτελείται από ένα σύνολο γραμμών και στάσεων με πολλές συνδέσεις μεταξύ τους. Η εύρεση της διαδρομής από ένα σταθμό σε κάποιο άλλο είναι συνήθως μια αρκετά δύσκολη εργασία. Η παρακάτω εικόνα παρουσιάζει την κεντρική περιοχή του μετρό.

Εικόνα 1: Μετρό του Κεντρικού Λονδίνου

Όπως είναι προφανές από το παραπάνω, από κάθε σταθμό διέρχονται περισσότερες από μια γραμμές. Η πληροφορία σχετικά με τις στάσεις και τις γραμμές του Μετρό σας δίνεται σε ένα αρχείο exec1.ecl στο compus. Η πληροφορία είναι με την ακόλουθη μορφή:

```
station(<Όνομα Σταθμού>,<X>,<Y>,[<Λίστα Γραμμών>]).
```

όπου <X> και <Y> οι συντεταγμένες του σταθμού (μη ακριβείς) πάνω σε κάποιο χάρτη. Για παράδειγμα από τον σταθμό camden_town, (22.0, 18.8) διέρχονται 2 γραμμές (northen_city και northen_west):

```
station(camden_town,22.0,18.8,[northern_city,northern_west]).
```

(α) Να υλοποιήσετε ένα κατηγόρημα **`connected(St1,St2,Line)`** το οποίο πετυχαίνει όταν οι σταθμοί **`St1`** και **`St2`**, είναι διαφορετικοί και συνδέονται απευθείας με την γραμμή **`Line`** (βρίσκονται στην ίδια γραμμή). Το κατηγόρημα θα μπορεί να επιστρέψει όλους τους πιθανούς συνδυασμούς σταθμών και γραμμών που τους συνδέουν. Για παράδειγμα:

```
?- connected(cannon_street, monument, Line).
Line = circle
Yes
Line = district
Yes
No

?- connected(camden_town, monument, Line).
No

?- connected(St1, St2, Line).
St1 = acton_town
```

```
St2 = aldgate_east
Line = district
Yes
St1 = acton_town
St2 = barons_court
Line = piccadilly
Yes
St1 = acton_town
St2 = barons_court
Line = district
Yes
St1 = acton_town
St2 = blackfriars
Line = district
Yes
(υπάρχουν ακόμη πολλές λύσεις...)
```

(β) Να ορίσετε ένα κατηγόρημα **number_of_stations(N)** το οποίο επιστρέφει το συνολικό αριθμό των στάσεων που περιέχονται στο αρχείο. Για παράδειγμα:

```
?- number_of_stations(N).
N = 102
Yes (0.00s cpu)
```

(γ) Να ορίσετε ένα κατηγόρημα το οποίο επιστρέφει το συνολικό αριθμό των γραμμών, που αναφέρονται στα γεγονότα **station/4**. Παράδειγμα εκτέλεσης:

```
?- number_of_lines(N).
N = 15
Yes (0.00s cpu)
```

(δ) Να ορίσετε ένα κατηγόρημα **find_route(InitialStation, FinalStation, Route)**, το οποίο επιστρέφει την διαδρομή από τον αρχικό σταθμό **InitiaStation**, στον τελικό σταθμό **FinalStation**. Το κατηγόρημα θα επιστρέφει μια διαδρομή, στην οποία ο επιβάτης χρησιμοποιεί **μια γραμμή το πολύ μια φορά και ΔΕΝ επισκέπτεται το ίδιο σταθμό δύο φορές**. Η μορφή της διαδρομής που επιστρέφεται θα πρέπει να έχει ανάμεσα σε δύο διαδοχικές στάσεις και την γραμμή που θα χρησιμοποιηθεί για να μεταβεί ο επιβάτης από την μια στάση στην επόμενη, σαν οδηγία (**get_line(<Γραμμή>)**). Για παράδειγμα

```
?- find_route(cannon_street, monument, Path).
Path = [cannon_street, get_line(circle), monument]
Yes (0.00s cpu, solution 1, maybe more)

Path = [cannon_street, get_line(district), monument]
Yes (0.03s cpu, solution 2, maybe more)

Path = [cannon_street, get_line(district), acton_town,
get_line(piccadilly), finsbury_park, get_line(victoria), euston,
get_line(northern_city), bank,
get_line(subway_between_bank_and_monument), monument]
Yes (0.04s cpu, solution 3, maybe more)

(υπάρχει πλήθος άλλων λύσεων)
```

```
?- find_route(camden_town, monument, Path).
Path     =     [camden_town,     get_line(northern_city),     bank,
get_line(subway_between_bank_and_monument), monument]
Yes (0.00s cpu, solution 1, maybe more)

Path     =     [camden_town,     get_line(northern_city),     bank,
get_line(central),  bond_street,  get_line(jubilee),  baker_street,
get_line(circle), monument]
Yes (0.00s cpu, solution 2, maybe more)

Path     =     [camden_town,     get_line(northern_city),     bank,
get_line(central),  bond_street,  get_line(jubilee),  baker_street,
get_line(metropolitan), aldgate_east, get_line(district), monument]
Yes (0.01s cpu, solution 3, maybe more)

(υπάρχει πλήθος άλλων λύσεων)
```

Προσοχή: Σε μερικές περιπτώσεις η λύση μπορεί να απαιτήσει χρόνο για να βρεθεί.

ΠΡΟΑΙΡΕΤΙΚΉ ΆΣΚΗΣΗ

Η διεύθυνση του μετρό, περηφανεύεται ότι αλλάζοντας *το πολύ τρεις γραμμές*, μπορείτε να φτάσετε από οποιαδήποτε σταθμό σε οποιαδήποτε άλλο σταθμό. Γράψτε ένα κατηγόρημα το οποίο βρίσκει ζεύγη σταθμών, όπου η μετάβαση από τον ένα στον άλλο απαιτεί ο επιβάτης να αλλάξει περισσότερες από τρεις γραμμές. Για παράδειγμα:

```
?- prove_manager_wrong(List).
List   =   [(aldwych,   angel),   (aldwych,   borough),   (aldwych,
london_bridge), (aldwych, old_street), (angel, kensington_olympia),
(borough, kensington_olympia), (kensington_olympia, london_bridge),
(kensington_olympia, old_street)]
```

Tips: (a) Μια λίστα με N μη δεσμευμένες μεταβλητές δημιουργείται με το length(List,N). (b) Αν υπάρχει σύνδεση από τον σταθμό Α στον Β, τότε υπάρχει και από το Β στο Α. Δεν χρειάζεται να βρείτε διαδρομή και για τα δύο ζεύγη.

ΠΑΡΑΔΟΣΗ

Στο compus, θα βρείτε το αρχείο exec1.ecl, το οποίο περιέχει κατηγορήματα που αναφέρονται στις ασκήσεις. Σε αυτό το αρχείο θα συμπληρώσετε τον κώδικά σας.

Θα παραδώσετε εντός της ημερομηνίας που αναφέρεται στο compus τα ακόλουθα:

- Ένα αρχείο με το όνομα **exec1.ecl** το οποίο θα περιέχει τις λύσεις (κατηγορήματα) **ΌΛΩΝ των ασκήσεων. Στην αρχή του αρχείου, σαν σχόλιο βάλτε τον Αριθμό μητρώου σας και το όνομά σας.**
- Ένα αρχείο **report.pdf** (***σε μορφή pdf***) το οποίο θα περιέχει:
  - **Στην πρώτη σελίδα το Όνομά σας, τον Αριθμό μητρώου σας και το email σας.**

- Για κάθε μια από τις ασκήσεις:
  - τον **κώδικα** (ασχέτως αν βρίσκεται και στο αρχείο exec1.ecl) και σχολιασμό σχετικά με αυτόν.
  - Παραδείγματα εκτέλεσης (2-3 ανάλογα για κάθε κατηγόρημα), όπου είναι εφαρμόσιμο.
  - Bugs και προβλήματα που έχει ο κώδικάς σας.

ΠΡΟΣΟΧΉ: ΝΑ ΑΚΟΛΟΥΘΗΣΕΤΕ ΑΥΣΤΗΡΑ ΤΑ ΟΝΟΜΑΤΑ ΚΑΙ ΤΗ ΣΕΙΡΑ ΤΩΝ ΟΡΙΣΜΑΤΩΝ ΠΟΥ ΔΙΝΕΤΑΙ ΠΑΡΑΠΑΝΩ (ΑΥΤΟΜΑΤΟΣ ΕΛΕΓΧΟΣ ΚΩΔΙΚΑ)

Καλή επιτυχία (*και have fun with Prolog!*)