Λογικός Προγραμματισμός με Περιορισμούς

Τμ. Εφαρμοσμένης Πληροφορικής

Εργασία 2 2019-2020

(υπάρχουν συνολικά 3 ασκήσεις)

1. No 4s available (40/100)

Πρέπει να εκδώσετε μια επιταγή για ένα βραβείο, αλλά ανακαλύψατε ότι το μηχάνημα δεν μπορεί να τυπώσει το ψηφίο 4. Η λύση που πρότεινε η διοίκηση, είναι να εκδοθούν δύο επιταγές, συνολικής αξίας ίσης με το αρχικό ποσό, έτσι ώστε τα ποσά στις δύο επιταγές να μην περιέχουν τον αριθμό 4. Για παράδειγμα, αν το βραβείο είναι 44 ευρώ οι δύο επιταγές που θα εκδοθούν είναι αξίας 5 και 39 ευρώ αντίστοιχα.

Να γράψετε ένα CLP κατηγόρημα, **split_check(Amount,Check1,Check2)**, το οποίο πετυχαίνει αν τα ποσά Check1 και Check2 έχουν άθροισμα Amount και δεν περιέχουν το ψηφίο 4. παράδειγμα εκτέλεσης:

```
?- split_check(44, Check1, Check2).
Check1 = 5
Check2 = 39

?- split_check(40404, Check1, Check2).
Check1 = 501
Check2 = 39903
```

Σημ: Στο compus, *θα βρείτε* το αρχείο exec2.ecl το οποίο έχει το κατηγόρημα **number_of_digits(Number, Digits),** το οποίο πετυχαίνει όταν Digits είναι ο αριθμός των ψηφίων του ακεραίου Number, πχ.

```
?- number_of_digits(1453, Digits).
Digits = 4
Yes (0.00s cpu)
```

Ίσως σας φανεί χρήσιμο.

Σημ2: Ύψωση σε δύναμη (3^5) μέσα σε περιορισμούς ίσως σας δημιουργήσει προβλήματα. Αν ο όρος είναι σταθερός, υπολογίστε την τιμή του με is/2 και χρησιμοποιήστε την μέσα στους περιορισμούς.

(google code jam problem)

2. BackUp my Servers (40/100)

Σε ένα περιβάλλον υπολογιστικής νέφους (cloud computing) πρέπει να γίνουν αντίγραφα ασφαλείας (backup) σε μια απομακρυσμένη υπηρεσία. Οι διαδικασίες που πρέπει να γίνουν αφορούν εξυπηρετητές βάσεων δεδομένων (db) και εξυπηρετητές δικτύου (web) και δίνονται από τα ακόλουθα γεγονότα:

```
%% Backup(Type,releaseDate,Duration,Bandwidth)
backup(db,srv_d1, 0, 5, 10).
backup(db,srv_d2, 2, 8, 18).
backup(db,srv_d3, 0, 4, 11).
backup(web,srv_w1, 0, 7, 8).
backup(web,srv_w2, 3, 11, 10).
```

Το πρώτο όρισμα του κάθε γεγονότος είναι ο τύπος του server που πρέπει να γίνει backup, το δεύτερο το όνομα του server, το τρίτο το *πότε* μπορεί να ξεκινήσει (release date) το backup, το τέταρτο η εκτιμώμενη διάρκεια (duration) και το πέμπτο οι απαιτήσεις σε εύρος δικτύου (bandwidth). Υποθέστε ότι:

- οι εργασίες αντιγράφων ασφαλείας για εξυπηρετητές (servers) *του ιδίου τύπου δεν* μπορεί να γίνει ταυτόχρονα, για παράδειγμα, *δεν μπορεί να γίνει ταυτόχρονα backup* στους δύο web servers,
- το *συνολικό εύρος δικτύου* (bandwidth) είναι 25, το οποίο σημαίνει ότι δεν είναι δυνατό να γίνουν ταυτόχρονα δύο οποιεσδήποτε εργασίες με άθροισμα σε απαιτήσεις πάνω από 25. Για παράδειγμα οι εργασίες για τον εξυπηρετητή srv_d2 (18) και srv_w2 (10) δεν μπορούν να γίνουν ταυτόχρονα.

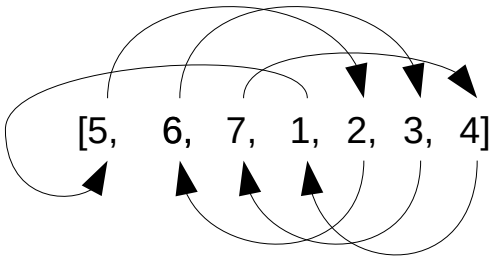
Να υλοποιήσετε ένα CLP πρόγραμμα **schedule_backups(Db, Web, M)**, το οποίο στο όρισμα **Db** περιέχει τους χρόνους έναρξης των εργασιών backup για τους εξυπηρετητές βάσεων δεδομένων, **Web** περιέχει τους χρόνους έναρξης των αντίστοιχων εργασιών για τους εξυπηρετητές δικτύου, και **M** ο συνολικός χρόνος όλων των εργασιών backup, *ο οποίος θα πρέπει να είναι ο ελάχιστος δυνατός*. Παράδειγμα εκτέλεσης:

```
?- schedule_backups(Db, Web, M).
Db = db([0, 7, 15])
Web = web([0, 15])
M = 26
Yes (0.02s cpu)
```

Σημ. Ο κώδικας σας πρέπει να είναι γενικός, δηλαδή να λειτουργεί με οποιαδήποτε αριθμό γεγονότων της παραπάνω μορφής.

3. Jumps in a List (20/100)

Στόχος σας είναι να δημιουργήσετε μια λίστα N ακεραίων από 1..N, τέτοια ώστε ξεκινώντας από τον ακέραιο που βρίσκεται στην πρώτη θέση της λίστας, και μεταβαίνοντας κάθε φορά στην θέση που αντιστοιχεί στην τιμή του, να διατρέχουμε όλες τις θέσεις επιστρέφοντας στην αρχική. Για παράδειγμα, η λίστα [5,6,7,1,2,3,4], είναι μια τέτοια λίστα, όπως φαίνεται στο σχήμα που ακολουθεί.



Αν κάθε άλμα πρέπει να είναι μεγαλύτερο από δύο (> 2), δηλαδή για παράδειγμα από την θέση 1 μπορώ να πάω στις θέσεις 4, 5, 6 ή 7, να γράψετε ένα CLP πρόγραμμα ώστε να υπολογίζει τη λίστα των παραπάνω ακεραίων. Το πρόγραμμα πρέπει να έχει ένα κατηγόρημα **arrange_list(Len,List)**, οπου **Len** είναι το μήκος της λίστας (Len > 7) και **List** η τελική λίστα των ακεραίων. Για παράδειγμα:

```
?- arrange_list(7, List).
List = [4, 5, 6, 7, 1, 2, 3]
Yes

?- arrange_list(8, List).
List = [5, 6, 7, 8, 2, 3, 4, 1]
Yes
```

Σημ: Υπάρχουν αρκετές λύσεις σε κάθε περίπτωση.

ΠΑΡΑΔΟΣΗ

Στο compus θα βρείτε το αρχείο exec2.ecl με όλο τον κώδικα που χρειάζεστε για τις παραπάνω ασκήσεις.

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

- Ένα αρχείο με το όνομα **exec2.ecl** το οποίο θα περιέχει **όλες τις λύσεις των ασκήσεων.**
- 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 τρεις ασκήσεις:
    - τον **κώδικα** (ασχέτως αν βρίσκεται και στο αντίστοιχο αρχείο) και σχολιασμό σχετικά με αυτόν.

- Παραδείγματα εκτέλεσης (για κάθε κατηγόρημα, όπου έχει νόημα)
- 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!*)