ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Α’ Ομάδα Ασκήσεων "Λογικού Προγραμματισμού"
Ακαδημαϊκού Έτους 2018-19

Άσκηση 1

Το φίδι της Πληροφορικής 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 της Πληροφορικής 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). Ορίστε σε Prolog ένα κατηγόρημα `disnake/3`, το οποίο όταν καλείται σαν `disnake(Pattern, Width, Height)` να εκτυπώνει τη στάση ξεκούρασης του φιδιού της Πληροφορικής, με επαναλαμβανόμενο σχήμα τους χαρακτήρες της λίστας `Pattern` και πλάτος (αντίστοιχα, ύψος) του ορθογωνίου που προαναφέρθηκε το μήκος της λίστας `Width` (αντίστοιχα, `Height`). Το πρόγραμμά σας δεν πρέπει να χρησιμοποιήσει ενσωματωμένα κατηγορήματα της Prolog, εκτός από τα απολύτως απαραίτητα για την έξοδο του αποτελέσματος, δηλαδή τα `write/1` και `nl/0`. Κάποια παραδείγματα εκτέλεσης του προγράμματος είναι τα εξής:

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

?- disnake([0,1,2,3,4,5,6,7,8,9],
           [_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,
            _,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_],
           [_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_]).
012345678901234567890123456 78
765432109876543210987654321 09
890123456789012345678901234 56
543210987654321098765432109 87
678901234567890123456789012 34
321098765432109876543210987 65
456789012345678901234567890 12
109876543210987654321098765 43
234567890123456789012345678 90
987654321098765432109876543 21
012345678901234567890123456 78
```

```
765432109876543210987654321 09
890123456789012345678901234 56
543210987654321098765432109 87
678901234567890123456789012 34
321098765432109876543210987 65
456789012345678901234567890 12
109876543210987654321098765 43
234567890123456789012345678 90
```

Παραδοτέο για την άσκηση είναι ***ένα*** **πηγαίο αρχείο Prolog** με όνομα **`disnake.pl`**.

Άσκηση 2[1]

Έστω ότι δίνονται τα παρακάτω γεγονότα Prolog[2]:

```
activity(a01, act(0,3)).
activity(a02, act(0,4)).
activity(a03, act(1,5)).
activity(a04, act(4,6)).
activity(a05, act(6,8)).
activity(a06, act(6,9)).
activity(a07, act(9,10)).
activity(a08, act(9,13)).
activity(a09, act(11,14)).
activity(a10, act(12,15)).
activity(a11, act(14,17)).
activity(a12, act(16,18)).
activity(a13, act(17,19)).
activity(a14, act(18,20)).
activity(a15, 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, υλοποιήστε σε Prolog ένα κατηγόρημα `assignment/4`, το οποίο, όταν καλείται ως `assignment(NP, ST, ASP, ASA)`, 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 στις μεταβλητές `ASP` και `ASA` δύο ισοδύναμες αναπαραστάσεις της ανάθεσης που πραγματοποιήθηκε. Συγκεκριμένα, η μεταβλητή `ASP` να είναι μία λίστα από στοιχεία

[1] Για την άσκηση αυτή πρέπει να χρησιμοποιήσετε απλή Prolog και όχι επεκτάσεις της που υποστηρίζουν προγραμματισμό με περιορισμούς.
[2] Βρίσκονται στο αρχείο: http://www.di.uoa.gr/~takis/activity.pl.

της μορφής N-A-T, όπου N είναι ο αύξων αριθμός ενός ατόμου (1, 2, …, NP), A είναι η λίστα των δραστηριοτήτων που ανατέθηκαν στο άτομο αυτό και T είναι η συνολική διάρκεια των δραστηριοτήτων που του ανατέθηκαν. Η μεταβλητή ASA να είναι μία λίστα από στοιχεία της μορφής A-N, που σημαίνουν ότι η δραστηριότητα A ανατίθεται στο άτομο N. Παραδείγματα ερωτήσεων και των αντίστοιχων απαντήσεων στο πρόγραμμα:

```
?- assignment(3, 14, ASP, ASA).

ASP = [1 - [a15, a12, a09, a07, a05, a03] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a10, a06, a02] - 12]
ASA = [a01 - 2, a02 - 3, a03 - 1, a04 - 2, a05 - 1, a06 - 3,
       a07 - 1, a08 - 2, a09 - 1, a10 - 3, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a10, a07, a05, a03] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a09, a06, a02] - 12]
ASA = [a01 - 2, a02 - 3, a03 - 1, a04 - 2, a05 - 1, a06 - 3,
       a07 - 1, a08 - 2, a09 - 3, a10 - 1, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a10, a06, a03] – 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a09, a07, a05, a02] - 12]
ASA = [a01 - 2, a02 - 3, a03 - 1, a04 - 2, a05 - 3, a06 - 1,
       a07 - 3, a08 - 2, a09 - 3, a10 - 1, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a09, a06, a03] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a10, a07, a05, a02] - 12]
ASA = [a01 - 2, a02 - 3, a03 - 1, a04 - 2, a05 - 3, a06 - 1,
       a07 - 3, a08 - 2, a09 - 1, a10 - 3, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a10, a06, a02] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a09, a07, a05, a03] - 12]
ASA = [a01 - 2, a02 - 1, a03 - 3, a04 - 2, a05 - 3, a06 - 1,
       a07 - 3, a08 - 2, a09 - 3, a10 - 1, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a09, a06, a02] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a10, a07, a05, a03] - 12]
ASA = [a01 - 2, a02 - 1, a03 - 3, a04 - 2, a05 - 3, a06 - 1,
       a07 - 3, a08 - 2, a09 - 1, a10 - 3, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a09, a07, a05, a02] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a10, a06, a03] - 12]
```

```
ASA = [a01 - 2, a02 - 1, a03 - 3, a04 - 2, a05 - 1, a06 - 3,
       a07 - 1, a08 - 2, a09 - 1, a10 - 3, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

ASP = [1 - [a15, a12, a10, a07, a05, a02] - 13,
       2 - [a14, a11, a08, a04, a01] - 14,
       3 - [a13, a09, a06, a03] - 12]
ASA = [a01 - 2, a02 - 1, a03 - 3, a04 - 2, a05 - 1, a06 - 3,
       a07 - 1, a08 - 2, a09 - 3, a10 - 1, a11 - 2, a12 - 1,
       a13 - 3, a14 - 2, a15 - 1]  --> ;

no

?- assignment(2, 40, ASP, ASA).
no

?- assignment(3, 13, ASP, ASA).
no
```

Το πρόγραμμα που θα γράψετε πρέπει να επιστρέφει όλες τις διαφορετικές λύσεις του προβλήματος, θεωρώντας ότι τα άτομα που συμμετέχουν στην ανάθεση είναι ισοδύναμα μεταξύ τους. Αυτό σημαίνει ότι για δεδομένη ανάθεση σε Ν άτομα, είναι ορθές και όλες οι Ν! αναθέσεις που προκύπτουν αν μεταθέσουμε τα άτομα με όλους τους δυνατούς τρόπους, αλλά, επί της ουσίας, είναι όλες ίδιες μεταξύ τους. Το πρόγραμμά σας πρέπει να επιστρέφει μόνο μία από αυτές. Για παράδειγμα, αν είχαμε

```
activity(a1, act(0,3)).
activity(a2, act(4,6)).
activity(a3, act(1,2)).
```

θα πρέπει η συμπεριφορά του προγράμματος να είναι κάπως έτσι:

```
?- assignment(2, 10, ASP, ASA).
ASP = [1 - [a3] - 1, 2 - [a2, a1] - 5]
ASA = [a1 - 2, a2 - 2, a3 - 1]  --> ;

ASP = [1 - [a3, a2] - 3, 2 - [a1] - 3]
ASA = [a1 - 2, a2 - 1, a3 - 1]  --> ;

no
```

Δηλαδή, υπάρχουν ακριβώς δύο διακριτές λύσεις για το πρόβλημα αυτό και όχι τέσσερις που θα προέκυπταν αν ανταλλάσαμε τις αναθέσεις των δύο ατόμων.

Παραδοτέο για την άσκηση είναι **ένα πηγαίο αρχείο Prolog** με όνομα **`assignment.pl`**, μέσα στο οποίο **δεν** θα πρέπει να περιέχονται γεγονότα `activity/2`.