ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΑΘΗΜΑ: ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ
ΑΚΑΔ. ΕΤΟΣ: 2023-24

2η Σειρά Προγραμματιστικών Ασκήσεων (Prolog)

Οι ασκήσεις αυτές είναι **ατομικές**. Οι απαντήσεις θα πρέπει να υποβληθούν με **turnin**, το αργότερο μέχρι την **Παρασκευή 31 Μαΐου 2024, ώρα 23:59**. Οι δύο σειρές προγραμματιστικών ασκήσεων δίνουν 1.5 μονάδες bonus στον τελικό βαθμό της τελικής και της επαναληπτικής εξέτασης, υπό την προϋπόθεση ότι ο βαθμός του αντίστοιχου γραπτού είναι τουλάχιστον 5. Η βαθμολογία των προγραμματιστικών ασκήσεων δεν μεταφέρεται σε άλλη ακαδημαϊκή χρονιά και δεν λαμβάνεται υπόψη στη βαθμολογία της επί πτυχίω εξέτασης.

Πριν ξεκινήσετε να γράφετε τα προγράμματα που ζητούνται στις ασκήσεις της σειράς αυτής, **διαβάστε πολύ προσεκτικά τις αναλυτικές οδηγίες** που ακολουθούν.

Οδηγίες

- Για να εγκαταστήσετε την Prolog στον υπολογιστή σας, μπορείτε να κατεβάσετε το διερμηνέα GNU Prolog από το σύνδεσμο

  http://www.gprolog.org/

  Συνοπτικές οδηγίες για τη χρήση του GNU Prolog υπάρχουν στις σημειώσεις του μαθήματος.

- Για τη συγγραφή των προγραμμάτων επιτρέπεται να χρησιμοποιήσετε προκαθορισμένα κατηγορήματα και προκαθορισμένους τελεστές **μόνο εφόσον αναφέρονται στις σημειώσεις του μαθήματος**.

- Για τη συγγραφή των προγραμμάτων θα πρέπει να χρησιμοποιήσετε το αρχείο πρότυπο Prolog.pro, στο οποίο για κάθε κατηγόρημα που ζητείτε να ορίσετε στις παρακάτω ασκήσεις, υπάρχει ένας κανόνας ο οποίος το ορίζει έτσι ώστε να επιστρέφει πάντα την απάντηση no. Για να απαντήσετε στις ασκήσεις αντικαταστήστε καθέναν από τους παραπάνω κανόνες με ένα κατάλληλο σύνολο προτάσεων που να ορίζει το αντίστοιχο κατηγόρημα. **Δεν θα πρέπει να τροποποιήσετε το όνομα κανενός κατηγορήματος ούτε το πλήθος των ορισμάτων του**.

- Οι ασκήσεις θεωρούν ως δεδομένο τον ορισμό κάποιων κατηγορημάτων. Τα κατηγορήματα αυτά ορίζονται με ένα πλήθος προτάσεων οι οποίες περιέχονται στο αρχείο πρότυπο Prolog.pro. **Δεν θα πρέπει να σβήσετε ούτε να τροποποιήσετε τις προτάσεις αυτές**.

- Μπορείτε να ορίσετε όσα βοηθητικά κατηγορήματα θέλετε, τα οποία θα χρησιμοποιούνται για τον ορισμό των κατηγορημάτων που σας ζητείται να υλοποιήσετε. Σε καμία περίπτωση δεν θα πρέπει να προσθέσετε άλλα ορίσματα στα κατηγορήματα που σας ζητούνται.

- **Αν χρησιμοποιήσετε προκαθορισμένα κατηγορήματα ή τελεστές που δεν αναφέρονται στις σημειώσεις του μαθήματος, η αντίστοιχη άσκηση δεν θα βαθμολογηθεί.**

- Ο έλεγχος της ορθότητας των απαντήσεων θα γίνει με ημι-αυτόματο τρόπο. Σε καμία περίπτωση δεν θα πρέπει ο βαθμολογητής να χρειάζεται να κάνει παρεμβάσεις στο αρχείο που θα υποβάλετε. Συνεπώς θα πρέπει να λάβετε υπόψη τα παρακάτω:

  1. Καθένα από τα κατηγορήματα που σας ζητείται να υλοποιήσετε θα πρέπει να έχει το συγκεκριμένο όνομα και το συγκεκριμένο πλήθος ορισμάτων που περιγράφεται στην εκφώνηση της αντίστοιχης άσκησης και που υπάρχει στο αρχείο πρότυπο Prolog.pro. **Αν σε κάποια άσκηση το όνομα ή το πλήθος των ορισμάτων δεν συμφωνεί με αυτόν που δίνεται στην εκφώνηση, η άσκηση δεν θα βαθμολογηθεί**.
  2. Το αρχείο που θα παραδώσετε δεν θα πρέπει να περιέχει συντακτικά λάθη. Αν υπάρχουν τμήματα κώδικα που περιέχουν συντακτικά λάθη, τότε θα πρέπει να τα διορθώσετε ή να τα αφαιρέσετε πριν από την παράδοση. **Αν το αρχείο που θα υποβάλετε περιέχει συντακτικά λάθη, τότε ολόκληρη η εργαστηριακή άσκηση θα μηδενιστεί**.
  3. Κατα τη διόρθωση των ασκήσεων οι βαθμολογητές δεν θα χρησιμοποιήσουν ερωτήσεις που εμπεριέχουν τα βοηθητικά κατηγορήματα τα οποία ενδεχομένως θα έχετε ορίσει. Η χρήση των βοηθητικών κατηγορημάτων θα πρέπει να γίνεται μέσα από τα κατηγορήματα που σας ζητείται να υλοποιήσετε.

- Για υποβολή με turnin γράψτε:

turnin Prolog@myy401 Prolog.pro

Άσκηση 1.
Μία διακρατική υπηρεσία για την πάταξη της τρομοκρατίας συλλέγει πληροφορίες για ένα πλήθος υπόπτων για τρομοκρατικές ενέργειες και καταγράφει γεγονότα τα οποία ενδεχομένως σχετίζονται με τρομοκρατικές δράσεις.

Τα δεδομένα αυτά καταγράφονται ως γεγονότα σε γλώσσα Prolog. Για λόγους ευκολίας, οι ημερομηνίες αναπαρίστανται χρησιμοποιώντας φυσικούς αριθμούς ξεκινώντας από κάποια αρχική ημερομηνία.

Μεταξύ άλλων, το πρόγραμμα Prolog περιέχει γεγονότα που ορίζουν τα παρακάτω κατηγορήματα:

- at(X,A,B,C): ο ύποπτος X βρισκόταν στην πόλη C από την ημέρα A ως και την ημέρα B.
- event(E,C,D): το γεγονός E συνέβη στην πόλη C την ημέρα D.
- country(C,S): η πόλη C βρίσκεται στη χώρα S.
- dif(T1,T2): ο όρος T1 είναι διαφορετικός από τον T2 (προϋποθέτει ότι κανένας από τους όρους T1 και T2 δεν είναι μεταβλητή).

Για την επεξεργασία των παραπάνω δεδομένων, σας ζητείται να γράψετε κανόνες που να ορίζουν τα παρακάτω κατηγορήματα:

- q1(C1,C2): Στις πόλεις C1 και C2 έχουν συμβεί καταγεγραμμένα γεγονότα την ίδια ημέρα (το κατηγόρημα να αληθεύει και στην περίπτωση όπου C1=C2).
- q2(X,Y,C): Οι ύποπτοι X και Y έχουν βρεθεί την ίδια μέρα στην πόλη C (το κατηγόρημα δεν πρέπει να αληθεύει στην περίπτωση όπου X=Y).
- q3(S): Στη χώρα S έχουν συμβεί τρία διαφορετικά γεγονότα σε διάστημα μίας εβδομάδας (ως εβδομάδα θεωρούμε το διάστημα που αποτελείται από οποιεσδήποτε 7 συνεχόμενες ημέρες).
- q4(X): Ο X έχει βρεθεί σε πόλεις δύο διαφορετικών χωρών σε ημέρες κατά τις οποίες συνέβησαν γεγονότα στις πόλεις αυτές.

Για έλεγχο χρησιμοποιήστε τις παρακάτω ερωτήσεις. Οι απαντήσεις προϋποθέτουν ότι τα κατηγορήματα at, event, country και dif έχουν οριστεί από τις προτάσεις που περιέχονται στο αρχείο Prolog.pro. Για λόγους απόκρυψης πληροφορίας δεν αναγράφονται τα πραγματικά ονόματα των υπόπτων, αλλά χρησιμοποιούνται ψευδώνυμα. Σημειώνεται ότι τα κατηγορήματα που θα ορίσετε θα πρέπει να δουλεύουν σωστά για οποιοδήποτε εναλλακτικό ορισμό των κατηγορημάτων at, event και country.

```
| ?- q1('Milan','Amsterdam').
yes
| ?- q1('Brussels','Frankfurt').
yes
| ?- q1('Rome','Bristol').
no

| ?- q2(hawk,tiger,'Berlin').
yes
| ?- q2(shark,tiger,'Rome').
yes
| ?- q2(hawk,tiger,'Hamburg').
no

| ?- q3('Belgium').
yes
| ?- q3('France').
yes
| ?- q3('Greece').
no

| ?- q4(tiger).
yes
| ?- q4(wolf).
yes
| ?- q4(snake).
no
```

Άσκηση 2.

Η συγκοινωνία ανάμεσα στις πόλεις μίας χώρας γίνεται με αεροπλάνα, πλοία και τρένα. Οι διαθέσιμες συνδέσεις περιγράφονται από τα παρακάτω κατηγορήματα:

- plane(A,B,N): η πόλη A συνδέεται με την πόλη B με αεροπλάνο και το κόστος του εισιτηρίου για τη μετακίνηση αυτή είναι N.
- boat(A,B,N): η πόλη A συνδέεται με την πόλη B με πλοίο και το κόστος του εισιτηρίου για τη μετακίνηση αυτή είναι N.
- train(A,B,N): η πόλη A συνδέεται με την πόλη B με τρένο και το κόστος του εισιτηρίου για τη μετακίνηση αυτή είναι N.

Έστω ότι τα κατηγορήματα plane, boat και train έχουν οριστεί από ένα πλήθος γεγονότων και κανόνων. Γράψτε κανόνες που να ορίζουν τα παρακάτω κατηγορήματα:

- p(A,B,N): μπορούμε να μεταβούμε από την πόλη A στην πόλη B μέσω οποιουδήποτε πλήθους από ενδιάμεσες πόλεις, έτσι ώστε το συνολικό κόστος των εισιτηρίων να είναι το πολύ N. Για τη μετάβαση από τη μία πόλη προς την επόμενη μπορούμε να χρησιμοποιήσουμε οποιοδήποτε μέσο συνδέει αυτές τις πόλεις.

- q(A,B): Υπάρχει αεροπορική σύνδεση ανάμεσα στις πόλεις A και B και επίσης υπάρχει τρόπος να μεταβούμε από την A και B χρησιμοποιώντας πλοία και τρένα, έτσι ώστε το συνολικό κόστος της μετακίνησης με πλοία και τρένα να είναι μικρότερο από το 80% της τιμής του αεροπορικού εισιτηρίου.
- r(A,B,N): μπορούμε να μεταβούμε από την πόλη A στην πόλη B μέσω οποιουδήποτε πλήθους από ενδιάμεσες πόλεις, έτσι ώστε το συνολικό κόστος των εισιτηρίων να είναι ακριβώς N, να χρησιμοποιήσουμε και τα τρία μέσα τουλάχιστον μία φορά το καθένα και να μην υπάρχουν διαδοχικές μετακινήσεις με τρένο (δηλαδή αν φτάσουμε σε μία ενδιάμεση πόλη με τρένο, θα πρέπει να φύγουμε από αυτή με πλοίο ή αεροπλάνο).

Για έλεγχο χρησιμοποιήστε τις παρακάτω ερωτήσεις. Οι απαντήσεις προϋποθέτουν ότι τα κατηγορήματα plane, boat και train έχουν οριστεί από τα γεγονότα που περιέχονται στο αρχείο Prolog.pro. Σημειώνεται ωστόσο ότι τα κατηγορήματα που θα ορίσετε θα πρέπει να δουλεύουν σωστά για οποιοδήποτε εναλλακτικό ορισμό των κατηγορημάτων plane, boat και train.

```
| ?- p(astralCity,brightCity, 84).
yes
| ?- p(astralCity,brightCity, 83).
no
| ?- p(zeroTown,luckyTown,190).
yes
| ?- p(whitePort,timePort,120).
yes
| ?- p(whitePort,timePort,110).
no


| ?- q(eternalCity,dreamCity).
yes
| ?- q(astralCity,crazyCity).
yes
| ?- q(brightCity,crazyCity).
no
| ?- q(crazyCity,dreamCity).
no

| ?- r(xenonTown,crazyCity,150).
yes
| ?- r(yellowTown,greenTown,150).
no
| ?- r(vainPort,rainyPort,176).
yes
| ?- r(oldTown,quietPort,176).
no
```

Άσκηση 3.
Μπορούμε να αναπαραστήσουμε κλάσματα στην Prolog χρησιμοποιώντας όρους της μορφής A/B, όπου A και B είναι θετικοί ακέραιοι αριθμοί. Γράψτε ένα πρόγραμμα σε Prolog για την υλοποίηση της πρόσθεσης δύο κλασμάτων. Συγκεκριμένα, ορίστε ένα κατηγόρημα fracSum(X,Y,Z) το οποίο θα αληθεύει αν ισχύουν τα παρακάτω:

- X,Y,Z είναι κλάσματα τέτοια ώστε X + Y = Z.
- Το Z είναι ένα απλοποιημένο κλάσμα, δηλαδή ο αριθμητής και ο παρονομαστής του δεν έχουν κανέναν κοινό διαιρέτη μεγαλύτερο του 1.

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- fracSum(1/4,1/12,Z).
Z = 1/3
| ?- fracSum(5/6,7/6,Z).
Z = 2/1
| ?- fracSum(3/5,3/20,Z).
Z = 3/4
```

Άσκηση 4.
Ο διπολικός διαιρέτης ενός θετικού ακέραιου αριθμού n ορίζεται με τον παρακάτω τρόπο:

- Αν $n = 1$ ή ο n είναι πρώτος αριθμός, τότε ο διπολικός διαιρέτης του n είναι ο εαυτός του.
- Αν ο n είναι σύνθετος αριθμός, τότε ο διπολικός διαιρέτης του n είναι το γινόμενο του μεγαλύτερου πρώτου αριθμού που διαιρεί το n επί τον μικρότερο πρώτο αριθμό που διαιρεί το n.

Γράψτε ένα πρόγραμμα σε Prolog για τον υπολογισμό του διπολικού διαρέτη ενός δεδομένου θετικού ακέραιου αριθμού. Συγκεκριμένα, ορίστε ένα κατηγόρημα bipolarDivisor(N,D), το οποίο θα αληθεύει αν η τιμή της N είναι ένας θετικός ακέραιος αριθμός n και η τιμή του D είναι ο διπολοκός διαιρέτης του n. Δεν μας ενδιαφέρει η συμπεριφορά του κατηγορήματος, όταν το πρώτο όρισμα δεν είναι φυσικός αριθμός.

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- bipolarDivisor(525,D).
D = 21

| ?- bipolarDivisor(1001,D).
D = 91

| ?- bipolarDivisor(5005,D).
D = 65
```

Άσκηση 5.
Μπορούμε να αναπαραστήσουμε φυσικούς αριθμούς στην Prolog χρησιμοποιώντας το ατομικό όρο 0 και ένα συναρτησιακό σύμβολο με ένα όρισμα, π.χ. το s. Ο φυσικός αριθμός n αναπαρίσταται με τον όρο $\underbrace{s(s(...s(}_{n}0)...))$ τον οποίο στη συνέχεια θα συμβολίζουμε για συντομία με $s^n(0)$ (η μοναδιαία αναπαράσταση περιγράφεται με περισσότερες λεπτομέρειες στις σημειώσεις, σελ 38). Η πράξη της διαίρεσης σε μοναδιαία αναπαράσταση περιγράφεται από την παρακάτω συνάρτηση (όπου το undefined είναι ατομικός όρος):

$$udiv(s^n(0), s^m(0)) = \begin{cases} s^{\lfloor \frac{n}{m} \rfloor}(0) & \text{αν } m > 0 \\ undefined & \text{αν } m = 0 \end{cases}$$

Γράψτε ένα πρόγραμμα σε Prolog για υλοποίηση της διαίρεσης φυσικών αριθμών σε μοναδιαία αναπαράσταση. Συγκεκριμένα ορίστε ένα κατηγόρημα divide(X,Y,D), το οποίο θα αληθεύει αν η τιμή της D ισούται με την τιμή της συνάρτησης *udiv* με ορίσματα τις τιμές των X,Y. **Το πρόγραμμα δεν επιτρέπεται να μετατρέπει τους όρους που αναπαριστούν ακέραιους σε ακέραιους της Prolog και να χρησιμοποιεί τους ενσωματωμένους αριθμητικούς τελεστές και το προκαθορισμένο κατηγόρημα is για αποτίμηση αριθμητικών παραστάσεων.**

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- divide(s(s(s(s(s(s(s(s(0)))))))),s(s(s(0))),D).
D = s(s(0))
| ?- divide(s(s(s((0)))),s(s(s(s(0)))),D).
D = 0
| ?- divide(s(s(s((0)))),0,D).
D = undefined
| ?- divide(0,s(s(s((0)))),D).
D = 0
```

Άσκηση 6.
Σε μία ψηφοφορία για την ανάδειξη του καλύτερου ποδοσφαιριστή της χρονιάς, ψηφίζουν n δημοσιογράφοι και νικητής είναι όποιος συγκεντρώσει περισσότερες από $n/2$ ψήφους. Αν κανένας υποψήφιος δεν συγκεντρώσει περισσότερες από $n/2$ ψήφους τότε δεν υπάρχει νικητής.

Γράψτε ένα πρόγραμμα σε Prolog το οποίο με δεδομένη μία λίστα με τις ψήψους των δημοσιογράφων θα βρίσκει το νικητή της ψηφοφορίας, εφόσον υπάρχει. Συγκεκριμένα, ορίστε ένα κατηγόρημα majority(L,X), το οποίο θα αληθεύει αν το X εμφανίζεται σε περισσότερες θέσεις από όσο είναι το μισό του μήκους της λίστας L.

Άσκηση 7.

Με δεδομένη μία λίστα, θέλουμε να μετρήσουμε το πλήθος των εμφανίσεων κάθε στοιχείου σε αυτή και να σχηματίσουμε μία νέα λίστα, η οποία θα περιέχει ένα όρο της μορφής `N*X`, για κάθε στοιχείο `X` που εμφανίζεται `N` φορές στην αρχική λίστα.

Γράψτε ένα πρόγραμμα σε Prolog που θα υλοποιεί την παραπάνω κατασκευή. Συγκεκριμένα, ορίστε ένα κατηγόρημα `freq(L,S)`, το οποίο θα αληθεύει αν η `S` είναι λίστα όρων της μορφής `N*X`, στην οποία ο όρος `N*X` περιέχεται ακριβώς μία φορά αν και μόνο αν το στοιχειο `X` περιέχεται ακριβώς `N` φορές στη λίστα `L`.

Για έλεγχο χρησιμοποιήστε τις παρακάτω ερωτήσεις. Η σειρά των όρων στην επιστρεφόμενη λίστα `S` δεν παίζει ρόλο.

```
| ?- freq([a,b,c],S).
S = [1*a,1*b,1*c]

| ?- freq([a,a,0,a,0,0,0,b,a,0],S).
S = [5*0,4*a,1*b]

| ?- freq([5,2,4,1,7,8,3,4,2,3,5,1,2,3,4],S).
S = [2*1,3*2,3*3,3*4,2*5,1*7,1*8]
```

Άσκηση 8.
Με δεδομένη μία λίστα L= $[x_1, x_2, \ldots, x_n]$ μήκους n και μία λίστα ακεραίων I= $[i_1, i_2, \ldots, i_k]$, τέτοια ώστε $1 \leq i_1 < i_2 < \ldots < i_k \leq n]$ θέλουμε να κατασκευάσουμε τη λίστα S= $[x_{i_1}, x_{i_2}, \ldots, x_{i_k}]$ (δηλαδή τη λίστα που αποτλείται από τα στοιχεία της L, οι θέσεις των οποίων υποδεικνύονται από τα στοιχεία της I).

Γράψτε ένα πρόγραμμα σε Prolog το οποίο θα πραγματοποιεί την παραπάνω κατασκευή. Συγκεκριμένα, ορίστε ένα κατηγόρημα pick(L,I,S), το οποίο θα αληθεύει αν L είναι μία λίστα, I είναι μία λίστα θετικών ακεραίων σε γνησίως αύξουσα τάξη, που οι τιμές τους δεν υπερβαίνουν το μήκος της L και S είναι η λίστα που αποτελείται από τα στοιχεία της L τα οποία βρίσκονται στις θέσεις που υποδεικνύουν τα στοιχεία της I.

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- pick([a,b,c,d,e,f,g,h],[2,4,7],S).
S = [b,d,g]

| ?- pick([a,b,c,d,e,f,g,h],[4,1],S).
no

| ?- pick([a,b,c,d,e,f,g,h],[6,7,8,9],S).
no
```

Άσκηση 9.

Με δεδομένη μία λίστα θετικών ακεραίων αριθμών L= $[x_1, x_2, \ldots, x_n]$ μήκους n θέλουμε να μετρήσουμε το πλήθος των ζευγών (x_i, x_j), με $1 \leq i < j \leq n$, για τα οποία ισχύει $x_i > 2x_j$ ή $x_j > 2x_i$. Γράψτε ένα πρόγραμμα σε Prolog το οποίο θα πραγματοποιεί τον παραπάνω υπολογισμό.

Συγκεκριμένα, ορίστε ένα κατηγόρημα count(L,C), το οποίο θα αληθεύει αν L είναι μία λίστα αποτελούμενη από θετικούς ακέραιους $[x_1, x_2, \ldots, x_n]$ και η τιμή της C είναι $|\{(x_i, x_j) \mid 1 \leq i < j \leq n$ και $(x_i > 2x_j$ ή $x_j > 2x_i)\}|$.

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- count([7,2,84,40,18],C).
C = 10

| ?- count([12,15,8,20,16],C).
C = 1

| ?- count([1,2,3,4,5,6,7,8,9,10],C).
C = 20
```

Άσκηση 10.
Σε ένα παιχνίδι που μοιάζει με το ντόμινο, δίνεται ένα πλήθος από λίστες και το ζητούμενο είναι να τοποθετηθούν οι λίστες αυτές η μία μετά την άλλη έτσι ώστε το τελευταίο στοιχείο κάθε λίστας να είναι το πρώτο στοιχείο της επόμενης.

Ορίστε ένα κατηγόρημα domino(L) σε Prolog , το οποίο θα αληθεύει αν η L είναι μία μη κενή λίστα με στοιχεία μη κενές λίστες, οι οποίες μπορούν να τοποθετηθούν στη σειρά με τον περιορισμό που περιγράφεται παραπάνω.

Παρακάτω δίνονται ορισμένες ερωτήσεις με τις αναμενόμενες απαντήσεις, τις οποίες μπορείτε να χρησιμοποιήσετε για έλεγχο του προγράμματος:

```
| ?- domino([[1,4],[2,6],[3,7],[4,2],[5,3],[6,9],[7,1],[8,5]]).
yes

| ?- domino([[1,2],[2,3],[2,8],[3,4],[3,5],[4,1],[5,1],[8,4],[8,5]]).
no
```