Input:
Assume that the input is a set of facts specified by the following predicates:

- childOf(X, Y): X is a child of Y
- female(X): X is female
- male(X): X is male

Task:
Write an ASP program that derives atoms over the following predicates:

- cousin(X, Y): whenever X is a cousin of Y

You can (but do not have to) use helper predicates, but make sure to keep the #show directives in the answer box!

Zum Beispiel:

Test	Resultat

`childOf(bart,homer). childOf(lisa,homer). childOf(maggie,homer).`
`childOf(homer,abe). childOf(herbert,abe).`
`childOf(marge,jacqueline). childOf(patty,jacqueline). childOf(selma,jacqueline).`
`childOf(bart,marge). childOf(lisa,marge). childOf(maggie,marge).`
`female(lisa). female(maggie). female(marge). female(jacqueline). female(patty). female(selma).`
`male(bart). male(homer). male(abe). male(herbert).`	

Antwort: (Abzugssystem: 10, 20, ... %)

Antwort zurücksetzen

```
1 sibling(A,B) :- childOf(A,C), childOf(B,C), A != B.
2 cousin(X,Y) :- childOf(X,D), childOf(Y,E), sibling(D,E).
3 #show cousin/2.
```

Input:
Assume that the input is a set of facts specified by the following predicates:

- adjacent(X, Y): country X is adjacent to country Y
- eu(X): country X is a member of the European Union
- candidate(X): country X is a candidate member
- euro(X): country X uses the Euro
- schengen(X): country X belongs to the Schengen area

Note: The name of a state is given by a string constant.

Task:
Write an ASP program that computes all Schengen countries that share a border with a Non-Schengen country. For each such county C, derive an atom outerBorder(C).

Zum Beispiel:

Test	Resultat
`#show outerBorder/1.`	`outerBorder("Bulgaria") outerBorder("Estonia") outerBorder("Finland") outerBorder("France") outerBorder("Greece") outerBorder("Hungary") outerBorder("Italy") outerBorder("Latvia") outerBorder`

Antwort: (Abzugssystem: 10, 20, ... %)

Antwort zurücksetzen

```
1 outerBorder(X) :- schengen(X), adjacent(X,Y), not schengen(Y).
2 #show outerBorder/1.
```

Kalkül | Regeln | Quantoren | Konjunktion | Symmetrie | Vollständigkeit | Semantik | Prädikat | Funktionssymbole

Fügen sie die fehlenden Begriffe an den richtigen Stellen ein!

Die Prädikatenlogik ist mächtiger, da man mit einem [Prädikat] im Regelfall viele gleichartige Zustände zusammenfassen und dieses dann abstrakt durch [Regeln] definieren kann. Die Syntak der Prädikatenlogik sieht neben Variablen und Konstanten auch [Funktionssymbole] vor, die wiederum auf Terme angewendet werden können. Ein wichtiges Element der Prädikatenlogik sind [Quantoren], mit denen Regeln festgelegt werden können.

Die [Semantik] der Prädikatenlogik definiert die Formeln rekursiv, indem Objekte der realen Welt den Konstanten, den Variablen und den Funktionssymbolen zugeordnet werden. Eine Wissensbasis setzt sich aus Regeln und Fakten zusammen, die per [Konjunktion] miteinander verknüpft sind und per [Kalkül] abgefragt werden können. Es gelten diverse Axiome wie die Reflexivität oder die [Symmetrie]. Die [Vollständigkeit] und Korrektheit einer PL-basierten Wissensbasis ist zu gewährleisten.