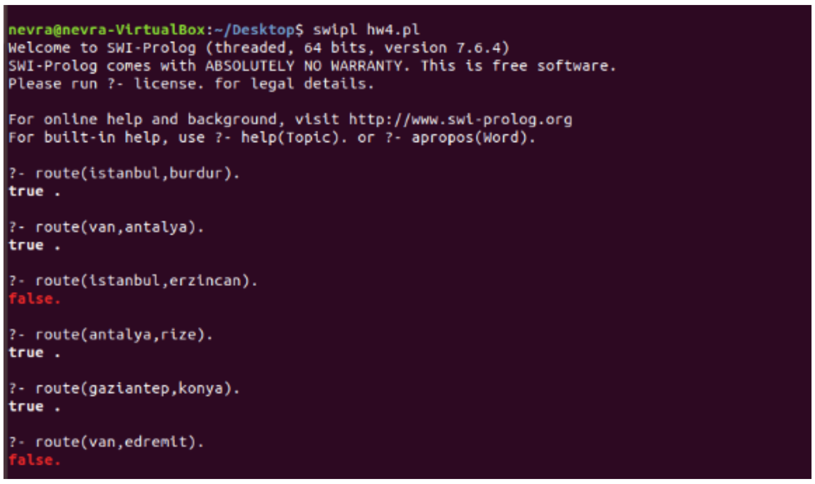
NEVRA GÜRSES
161044071

CSE341 – Programming Languages

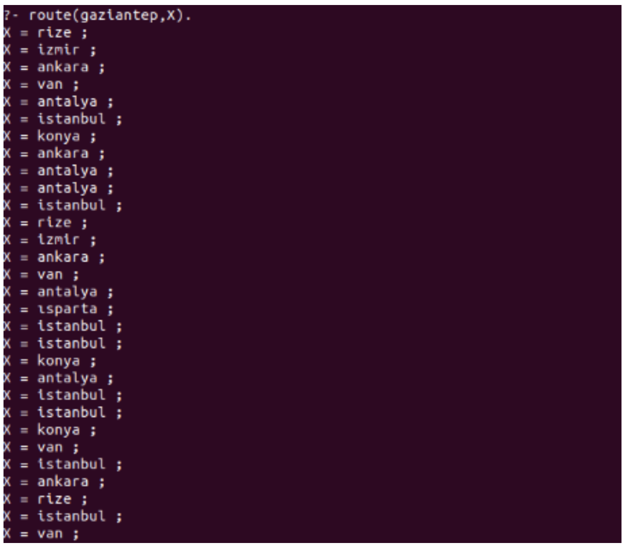
Homework #4

PART 1: Firstly,I write all facts.Then I control directed or connected route between two cities.

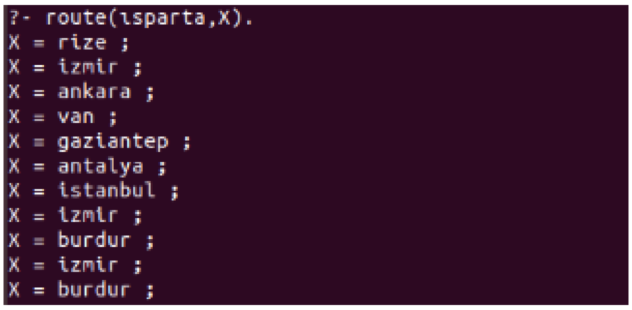
Sample outputs of "route" predicate are:



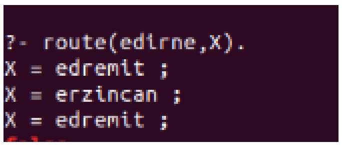
```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- route(istanbul,burdur).
true .

?- route(van,antalya).
true .

?- route(istanbul,erzincan).
false.

?- route(antalya,rize).
true .

?- route(gaziantep,konya).
true .

?- route(van,edremit).
false.
```



```
?- route(gaziantep,X).
X = rize ;
X = izmir ;
X = ankara ;
X = van ;
X = antalya ;
X = istanbul ;
X = konya ;
X = ankara ;
X = antalya ;
X = antalya ;
X = istanbul ;
X = rize ;
X = izmir ;
X = ankara ;
X = van ;
X = antalya ;
X = isparta ;
X = istanbul ;
X = istanbul ;
X = konya ;
X = antalya ;
X = istanbul ;
X = istanbul ;
X = konya ;
X = van ;
X = istanbul ;
X = ankara ;
X = rize ;
X = istanbul ;
X = van ;
```



```
?- route(isparta,X).
X = rize ;
X = izmir ;
X = ankara ;
X = van ;
X = gaziantep ;
X = antalya ;
X = istanbul ;
X = izmir ;
X = burdur ;
X = izmir ;
X = burdur ;
```

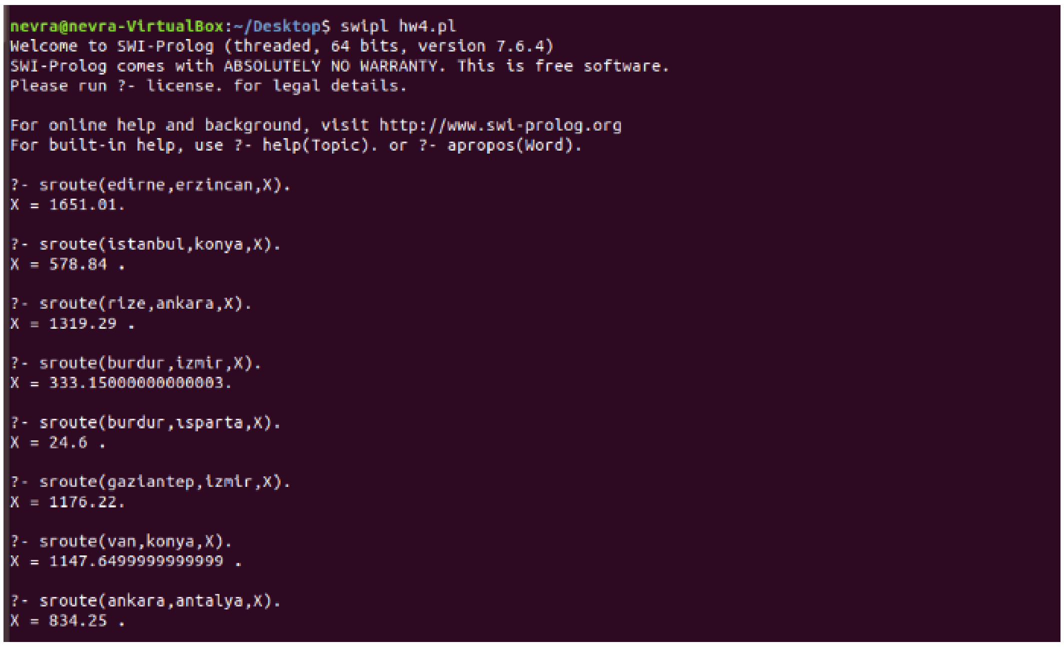


```
?- route(edirne,X).
X = edremit ;
X = erzincan ;
X = edremit ;
```

NEVRA GÜRSES
161044071

PART 2: In this part,according to “distance” facts,I calculate directed or connected distance between two cities.

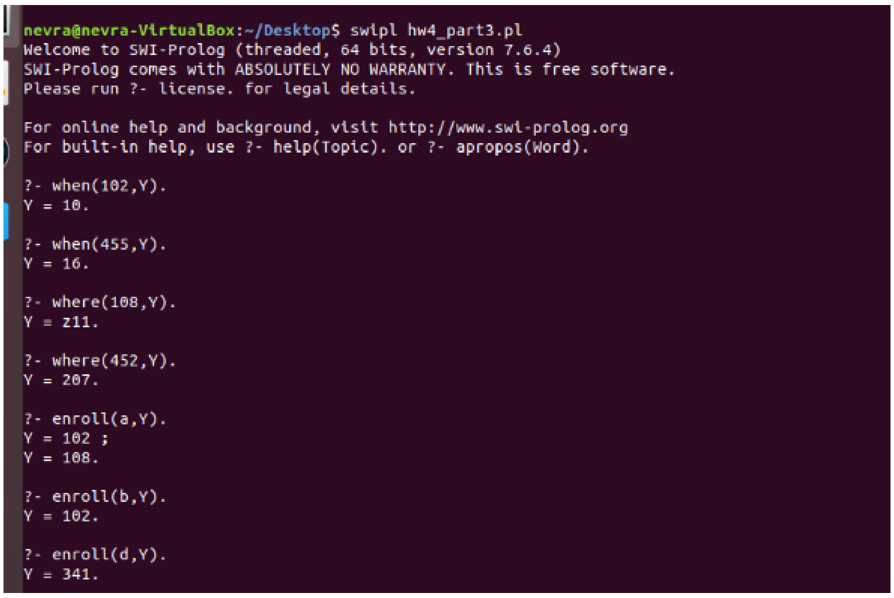
Sample output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- sroute(edirne,erzincan,X).
X = 1651.01.

?- sroute(istanbul,konya,X).
X = 578.84 .

?- sroute(rize,ankara,X).
X = 1319.29 .

?- sroute(burdur,izmir,X).
X = 333.15000000000003.

?- sroute(burdur,ısparta,X).
X = 24.6 .

?- sroute(gaziantep,izmir,X).
X = 1176.22.

?- sroute(van,konya,X).
X = 1147.6499999999999 .

?- sroute(ankara,antalya,X).
X = 834.25 .
```

PART 3: In this part,firstly I write all facts.This facts are “classes” facts and “enrollment” facts.After I write “when” predicate.I write this predicate using classes facts.Then I write “where” predicate.I write also this predicate using classes facts.Then,I write enroll predicate.I write this fact using enrollment facts.

Sampe outputs of this predicates are:

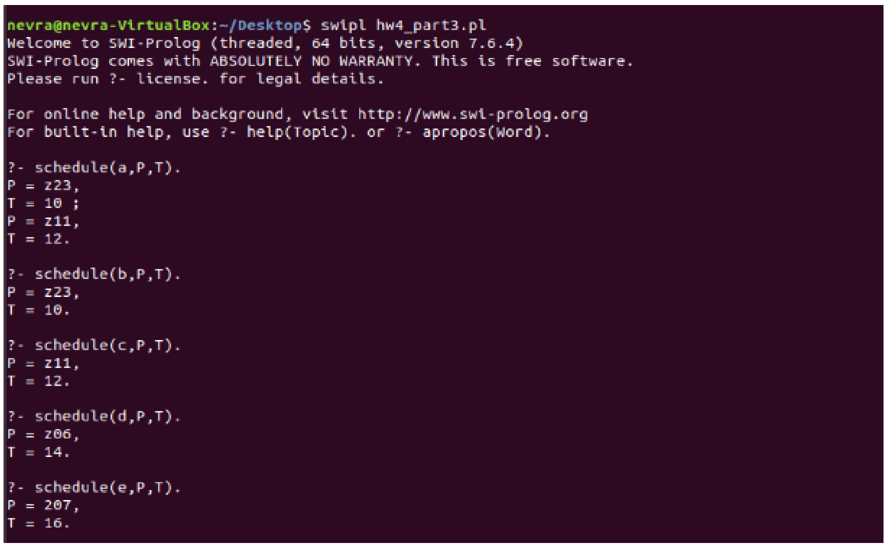


```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- when(102,Y).
Y = 10.

?- when(455,Y).
Y = 16.

?- where(108,Y).
Y = z11.

?- where(452,Y).
Y = 207.

?- enroll(a,Y).
Y = 102 ;
Y = 108.

?- enroll(b,Y).
Y = 102.

?- enroll(d,Y).
Y = 341.
```

NEVRA GÜRSES
161044071

3.1-) I write Schedule predicate using “enroll”, “where”, “when”, predicate.If student x is taking course y according to enroll predicate,then we find class and time of course y with using “when” and “where” predicates.

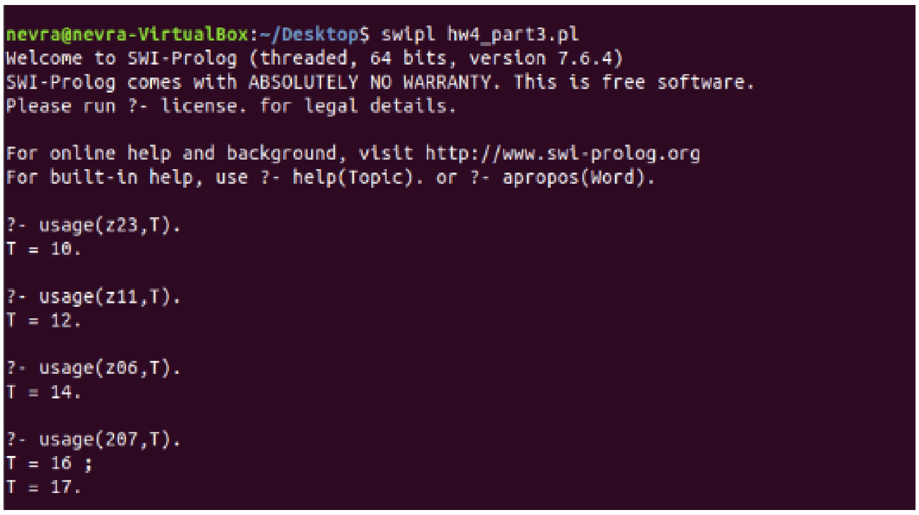
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- schedule(a,P,T).
P = z23,
T = 10 ;
P = z11,
T = 12.

?- schedule(b,P,T).
P = z23,
T = 10.

?- schedule(c,P,T).
P = z11,
T = 12.

?- schedule(d,P,T).
P = z06,
T = 14.

?- schedule(e,P,T).
P = 207,
T = 16.
```

3.2-) I write usage predicate using **“**classes” facts.

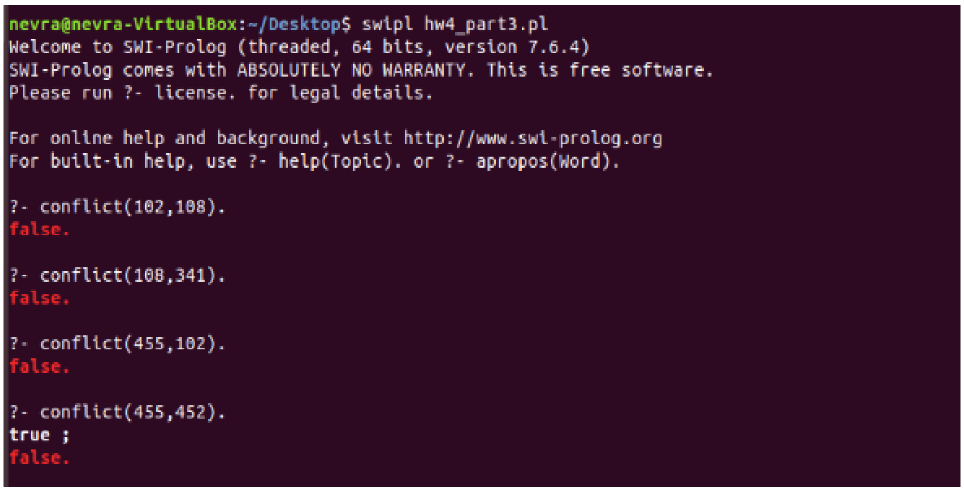
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- usage(z23,T).
T = 10.

?- usage(z11,T).
T = 12.

?- usage(z06,T).
T = 14.

?- usage(207,T).
T = 16 ;
T = 17.
```

3.3-) I write conflict predicate using “when” and “where” predicates.I contol conflict of courses in this predicate.If time or place of course X and Y is overlap,then they are conflict.

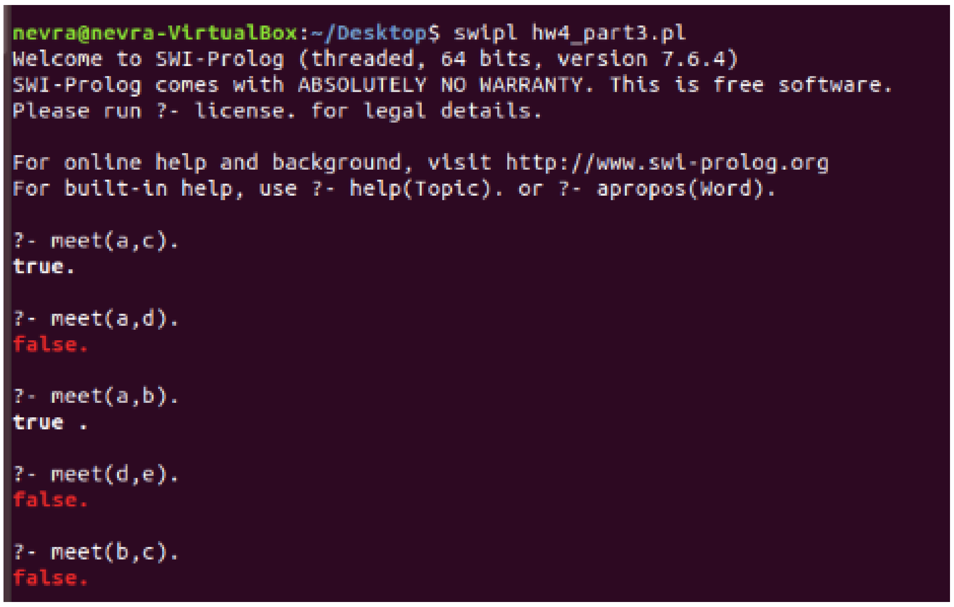
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- conflict(102,108).
false.

?- conflict(108,341).
false.

?- conflict(455,102).
false.

?- conflict(455,452).
true ;
false.
```

3.4-) I write meet predicate using enrollment facts.If student x and y are taking same courses,then they meet.

Output of this predicate is:



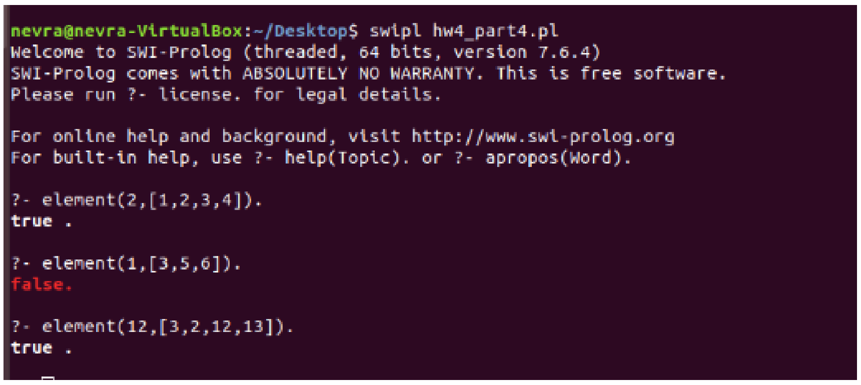
```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- meet(a,c).
true.

?- meet(a,d).
false.

?- meet(a,b).
true .

?- meet(d,e).
false.

?- meet(b,c).
false.
```

NEVRA GÜRSES
161044071

PART 4:

4.1-) I wrote “element” predicate using member predicate.

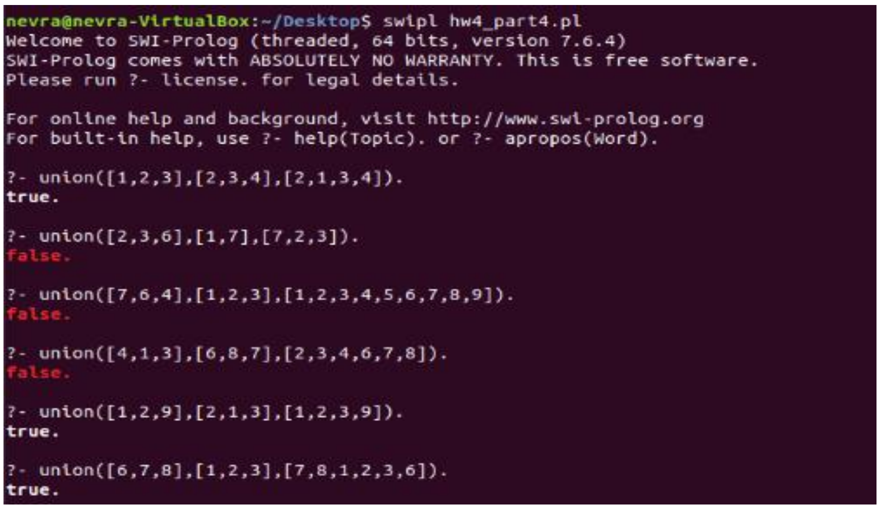
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- element(2,[1,2,3,4]).
true .

?- element(1,[3,5,6]).
false.

?- element(12,[3,2,12,13]).
true .
```

4-2) I write “union” predicate with using 2 helper predicates. “Common” predicate is to control whether all elements of S1 or S2 in S3. “CommonFlag” predicate is to control whether all elements in S3 is also in S1 or S2 .This prevent to write result of predicate true if all elements S1 and S2 in S3 but there are other elements in S3 that are not in S1 or S2.

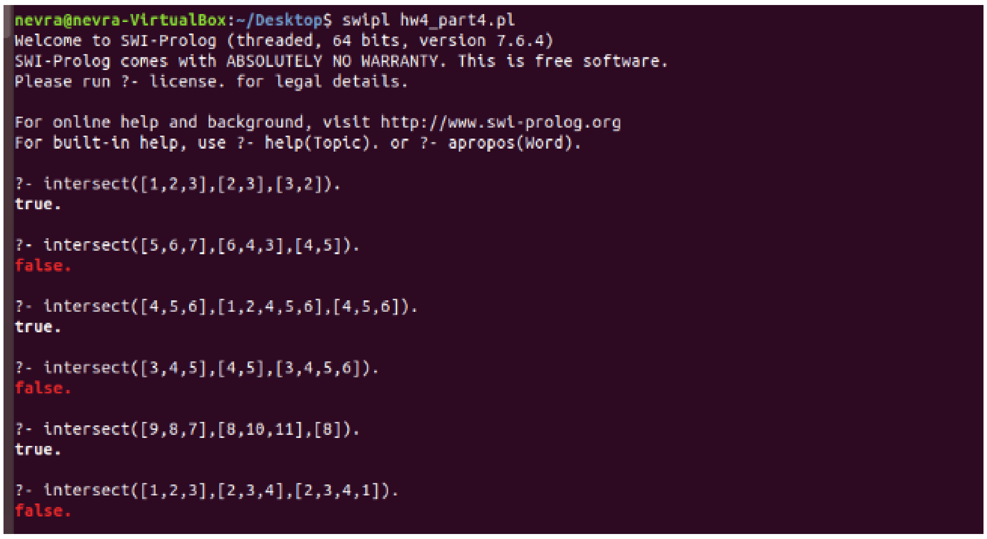
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- union([1,2,3],[2,3,4],[2,1,3,4]).
true.

?- union([2,3,6],[1,7],[7,2,3]).
false.

?- union([7,6,4],[1,2,3],[1,2,3,4,5,6,7,8,9]).
false.

?- union([4,1,3],[6,8,7],[2,3,4,6,7,8]).
false.

?- union([1,2,9],[2,1,3],[1,2,3,9]).
true.

?- union([6,7,8],[1,2,3],[7,8,1,2,3,6]).
true.
```

4.3-) I write “intersect” predicate with using 2 helper predicates. “Overlap” predicate is to control whether all common elements of S1 and S2 is also in S3. “IntersectFlag” predicate is to control whether all elements in S3 is   common elements of S1 and S2 .This prevent to write result of predicate true if all common elements of S1 and S2 in S3 but there are other elements in S3 that are not in S1 and S2.

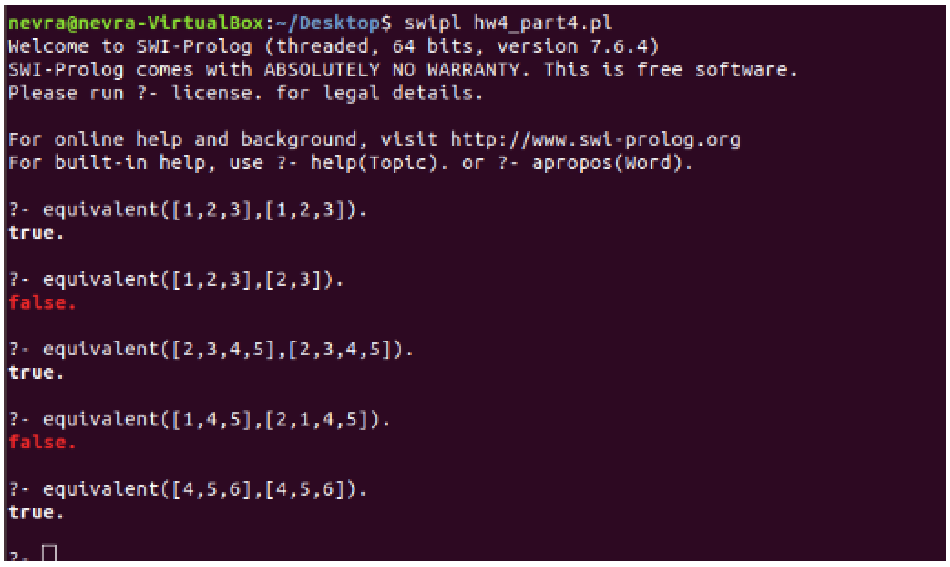
Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- intersect([1,2,3],[2,3],[3,2]).
true.

?- intersect([5,6,7],[6,4,3],[4,5]).
false.

?- intersect([4,5,6],[1,2,4,5,6],[4,5,6]).
true.

?- intersect([3,4,5],[4,5],[3,4,5,6]).
false.

?- intersect([9,8,7],[8,10,11],[8]).
true.

?- intersect([1,2,3],[2,3,4],[2,3,4,1]).
false.
```

4.4-) Equivalent predicate control whether two set equal or not.

Output of this predicate is:



```
nevra@nevra-VirtualBox:~/Desktop$ swipl hw4_part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- equivalent([1,2,3],[1,2,3]).
true.

?- equivalent([1,2,3],[2,3]).
false.

?- equivalent([2,3,4,5],[2,3,4,5]).
true.

?- equivalent([1,4,5],[2,1,4,5]).
false.

?- equivalent([4,5,6],[4,5,6]).
true.

?-
```