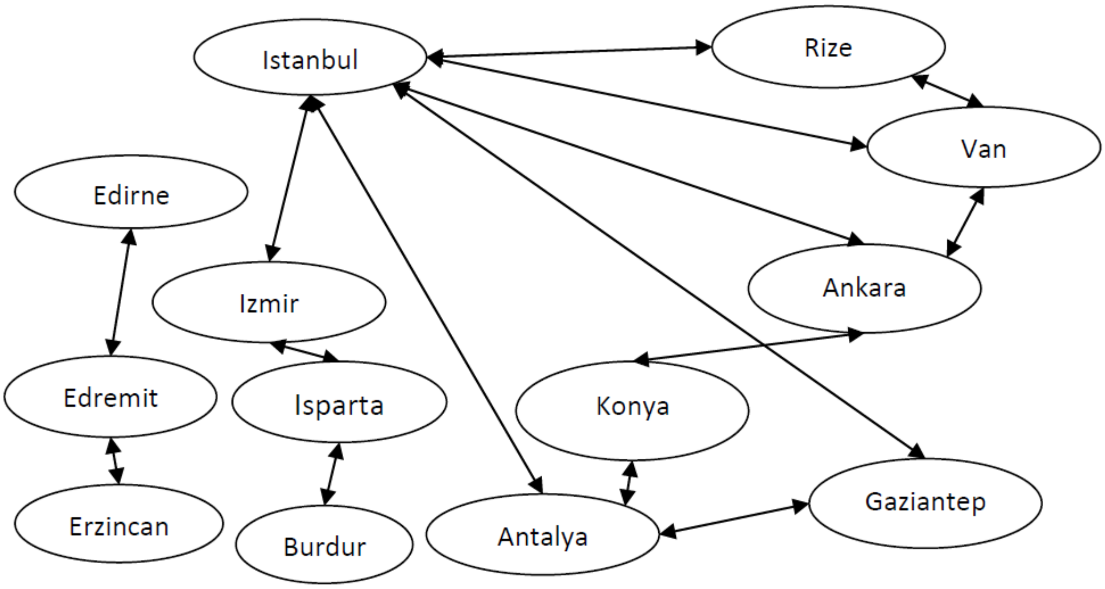
CSE 341 PROGRAMMING LANGUAGES FALL 2020
HOMEWORK 4 REPORT

Emirhan UZUN

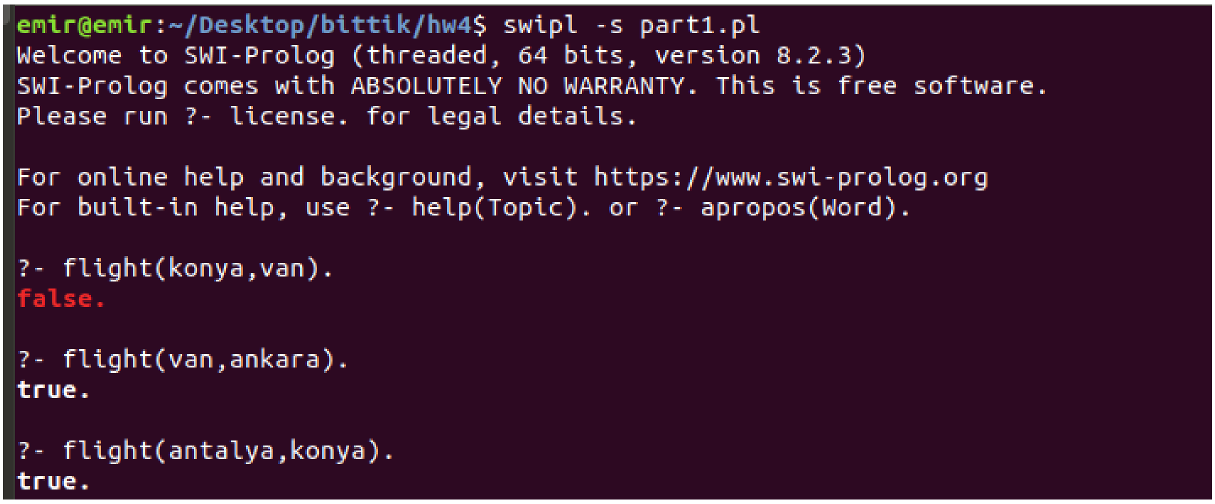
171044019



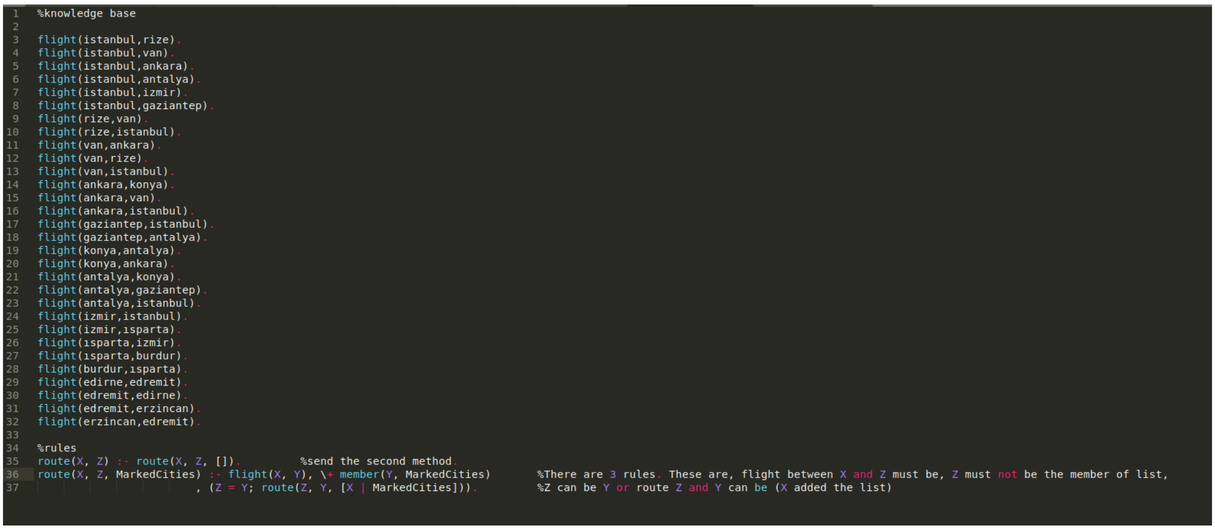
PART1

In the graph above we see the possible flights between some of the cities in Turkey. Write the predicate "route(X,Y) – a route between X and Y exists" that returns true of if there is a route between any given two cities. You can execute the program like this :

1 – swipl -s part1.pl

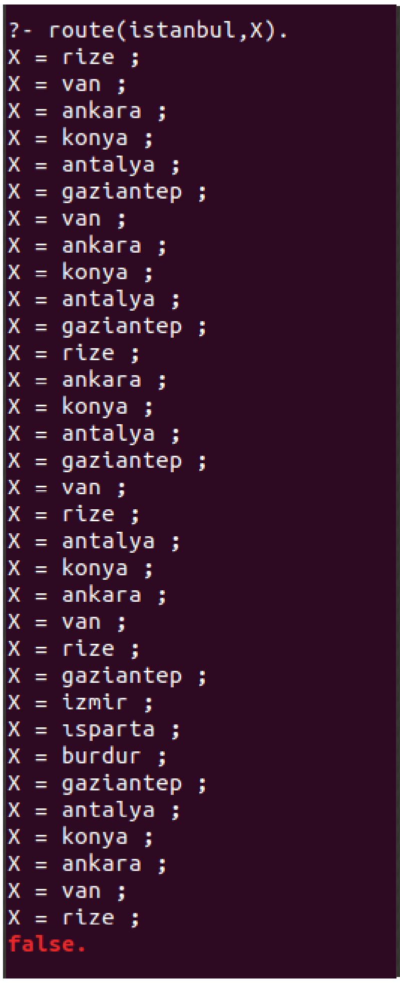


```
emir@emir:~/Desktop/bittik/hw4$ swipl -s part1.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- flight(konya,van).
false.

?- flight(van,ankara).
true.

?- flight(antalya,konya).
true.
```

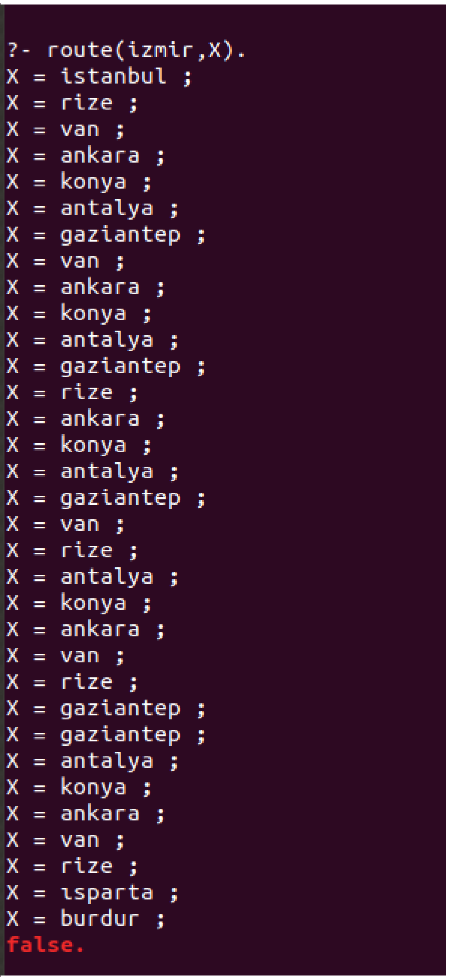


```prolog
%knowledge base

flight(istanbul,rize).
flight(istanbul,van).
flight(istanbul,ankara).
flight(istanbul,antalya).
flight(istanbul,izmir).
flight(istanbul,gaziantep).
flight(rize,van).
flight(rize,istanbul).
flight(van,ankara).
flight(van,rize).
flight(van,istanbul).
flight(ankara,konya).
flight(ankara,van).
flight(ankara,istanbul).
flight(gaziantep,istanbul).
flight(gaziantep,antalya).
flight(konya,antalya).
flight(konya,ankara).
flight(antalya,konya).
flight(antalya,gaziantep).
flight(antalya,istanbul).
flight(izmir,istanbul).
flight(izmir,ısparta).
flight(ısparta,izmir).
flight(ısparta,burdur).
flight(burdur,ısparta).
flight(edirne,edremit).
flight(edremit,edirne).
flight(edremit,erzincan).
flight(erzincan,edremit).

%rules
route(X, Z) :- route(X, Z, []).          %send the second method.
route(X, Z, MarkedCities) :- flight(X, Y), \+ member(Y, MarkedCities)       %There are 3 rules. These are, flight between X and Z must be, Z must not be the member of list,
                        , (Z = Y; route(Z, Y, [X | MarkedCities])).         %Z can be Y or route Z and Y can be (X added the list)
```

The first rule calls the other route rule. Then this rule checks 3 things. First one is checks if there is a flight between X and Y cities. Then second rule checks if Y is not member of the list which is marked already. And the last one checks if the Z city is equal the Y city that is the expected or send the route method Z and Y cities with marked the X city and add the list.



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

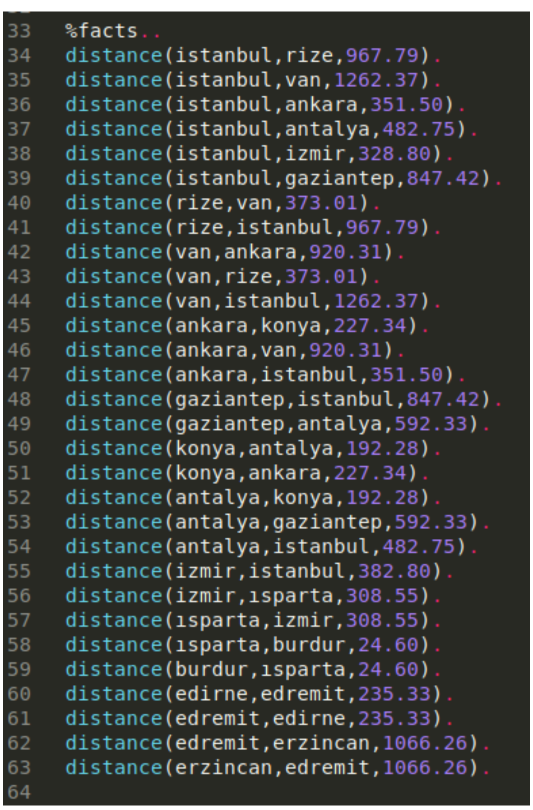


```
?- route(izmir,X).
X = istanbul ;
X = rize ;
X = van ;
X = ankara ;
X = konya ;
X = antalya ;
X = gaziantep ;
X = van ;
X = ankara ;
X = konya ;
X = antalya ;
X = gaziantep ;
X = rize ;
X = ankara ;
X = konya ;
X = antalya ;
X = gaziantep ;
X = van ;
X = rize ;
X = antalya ;
X = konya ;
X = ankara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

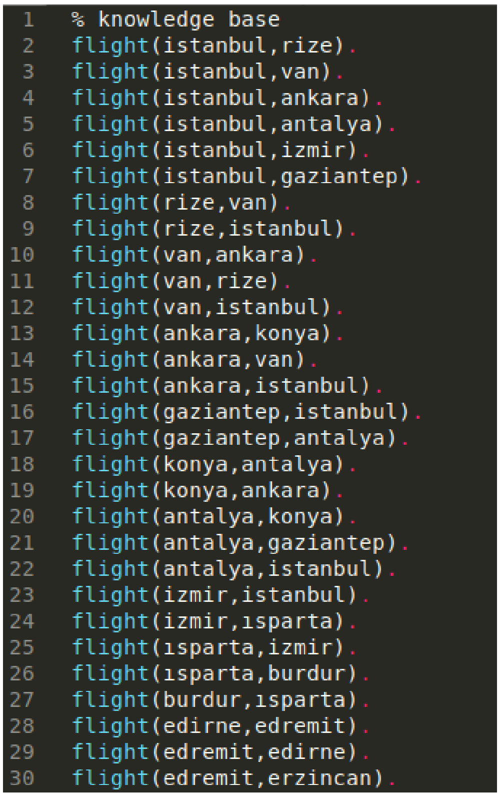
PART2

Continuing with the previous problem, we are asked to write a program that checks if a route exists between two cities and if so, provides the shortest route. You can execute the program like this :

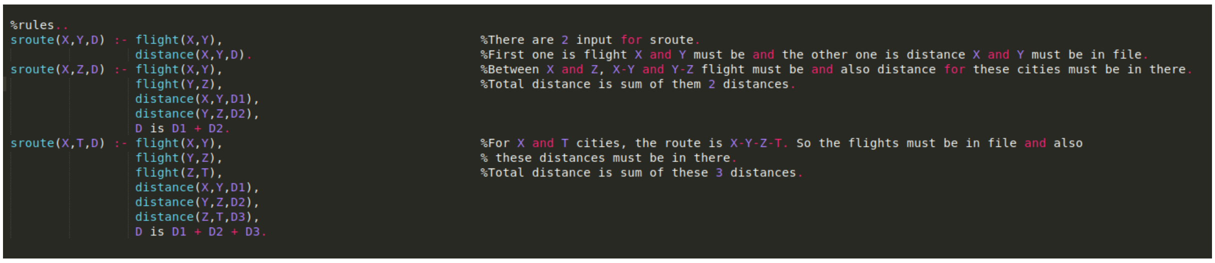
1 – swipl -s part2.pl



```
%facts..
distance(istanbul,rize,967.79).
distance(istanbul,van,1262.37).
distance(istanbul,ankara,351.50).
distance(istanbul,antalya,482.75).
distance(istanbul,izmir,328.80).
distance(istanbul,gaziantep,847.42).
distance(rize,van,373.01).
distance(rize,istanbul,967.79).
distance(van,ankara,920.31).
distance(van,rize,373.01).
distance(van,istanbul,1262.37).
distance(ankara,konya,227.34).
distance(ankara,van,920.31).
distance(ankara,istanbul,351.50).
distance(gaziantep,istanbul,847.42).
distance(gaziantep,antalya,592.33).
distance(konya,antalya,192.28).
distance(konya,ankara,227.34).
distance(antalya,konya,192.28).
distance(antalya,gaziantep,592.33).
distance(antalya,istanbul,482.75).
distance(izmir,istanbul,382.80).
distance(izmir,ısparta,308.55).
distance(ısparta,izmir,308.55).
distance(ısparta,burdur,24.60).
distance(burdur,ısparta,24.60).
distance(edirne,edremit,235.33).
distance(edremit,edirne,235.33).
distance(edremit,erzincan,1066.26).
distance(erzincan,edremit,1066.26).
```

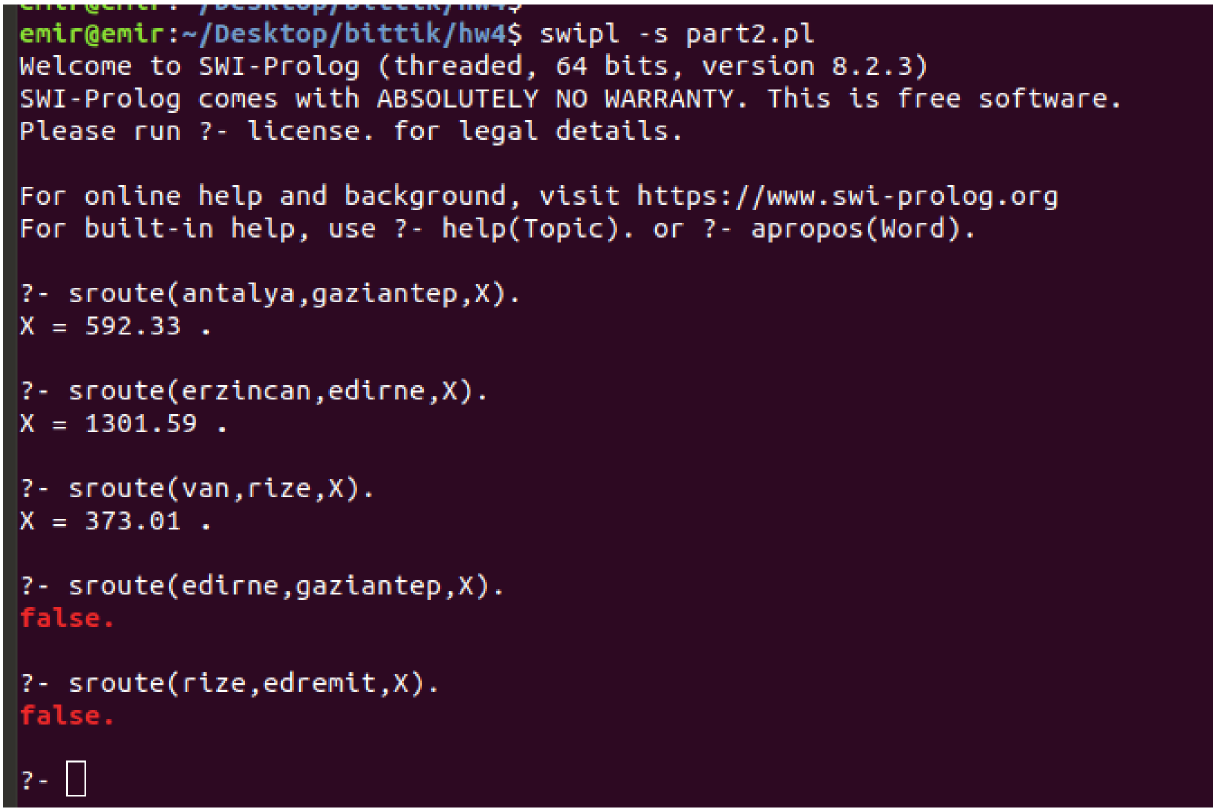


```
% knowledge base
flight(istanbul,rize).
flight(istanbul,van).
flight(istanbul,ankara).
flight(istanbul,antalya).
flight(istanbul,izmir).
flight(istanbul,gaziantep).
flight(rize,van).
flight(rize,istanbul).
flight(van,ankara).
flight(van,rize).
flight(van,istanbul).
flight(ankara,konya).
flight(ankara,van).
flight(ankara,istanbul).
flight(gaziantep,istanbul).
flight(gaziantep,antalya).
flight(konya,antalya).
flight(konya,ankara).
flight(antalya,konya).
flight(antalya,gaziantep).
flight(antalya,istanbul).
flight(izmir,istanbul).
flight(izmir,ısparta).
flight(ısparta,izmir).
flight(ısparta,burdur).
flight(burdur,ısparta).
flight(edirne,edremit).
flight(edremit,edirne).
flight(edremit,erzincan).
```



```
%rules..
sroute(X,Y,D) :- flight(X,Y),                      %There are 2 input for sroute.
                 distance(X,Y,D).                  %First one is flight X and Y must be and the other one is distance X and Y must be in file.
sroute(X,Z,D) :- flight(X,Y),                      %Between X and Z, X-Y and Y-Z flight must be and also distance for these cities must be in there.
                 flight(Y,Z),                      %Total distance is sum of them 2 distances.
                 distance(X,Y,D1),
                 distance(Y,Z,D2),
                 D is D1 + D2.
sroute(X,T,D) :- flight(X,Y),                      %For X and T cities, the route is X-Y-Z-T. So the flights must be in file and also
                 flight(Y,Z),                      % these distances must be in there.
                 flight(Z,T),                      %Total distance is sum of these 3 distances.
                 distance(X,Y,D1),
                 distance(Y,Z,D2),
                 distance(Z,T,D3),
                 D is D1 + D2 + D3.
```

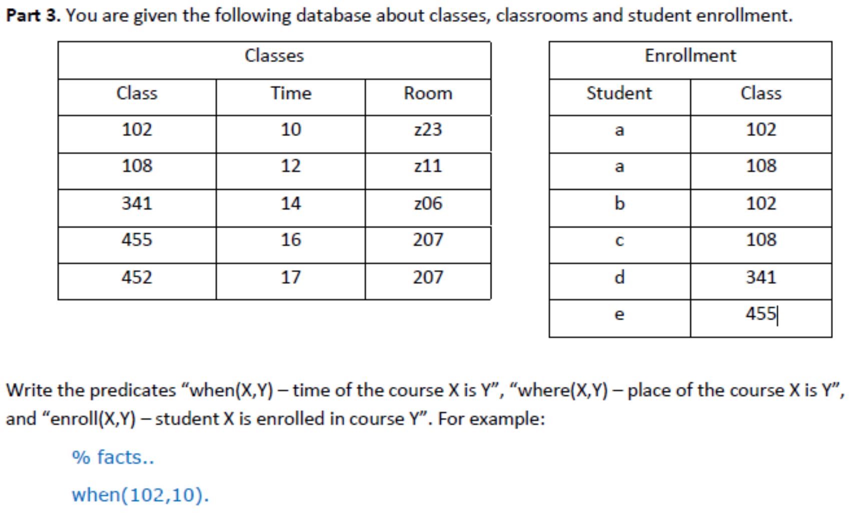
In the part 2, we are asked to find the route distance. My explanations about this part is in the file code and above picture. The result images are below:



```
emir@emir:~/Desktop/bittik/hw4$
emir@emir:~/Desktop/bittik/hw4$ swipl -s part2.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- sroute(antalya,gaziantep,X).
X = 592.33 .

?- sroute(erzincan,edirne,X).
X = 1301.59 .

?- sroute(van,rize,X).
X = 373.01 .

?- sroute(edirne,gaziantep,X).
false.

?- sroute(rize,edremit,X).
false.

?- 
```

PART3

1 – swipl -s part3.pl



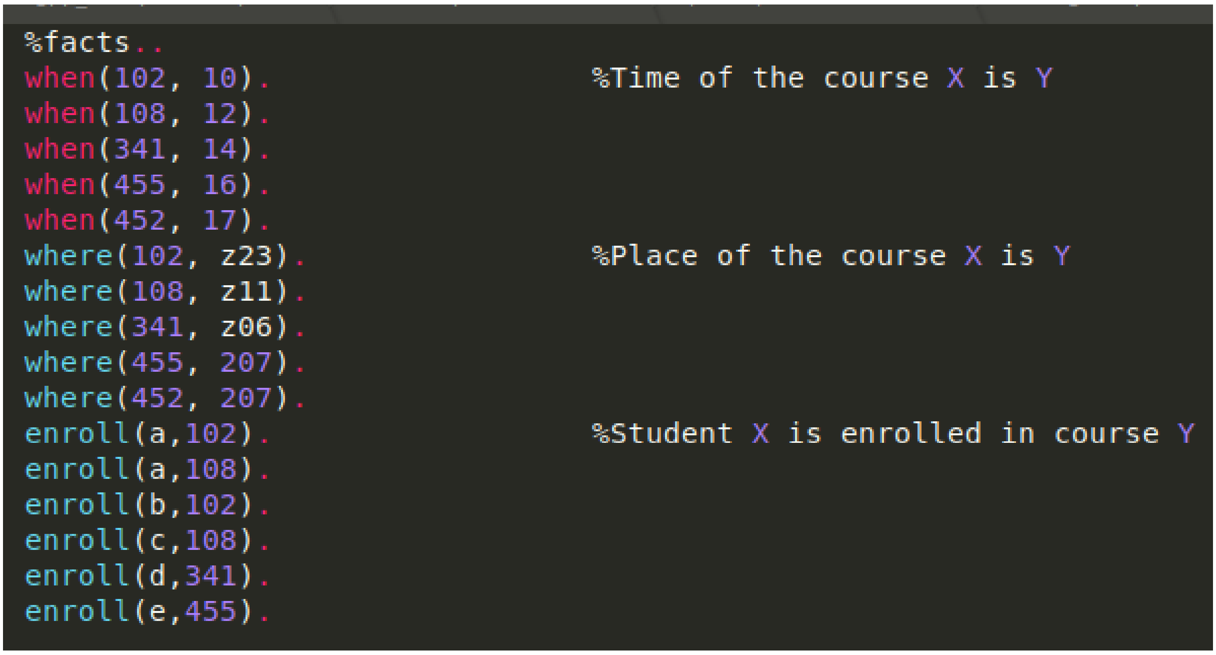
Part 3. You are given the following database about classes, classrooms and student enrollment.

Classes		
Class	Time	Room
102	10	z23
108	12	z11
341	14	z06
455	16	207
452	17	207

Enrollment	
Student	Class
a	102
a	108
b	102
c	108
d	341
e	455

Write the predicates “when(X,Y) – time of the course X is Y”, “where(X,Y) – place of the course X is Y”, and “enroll(X,Y) – student X is enrolled in course Y”. For example:

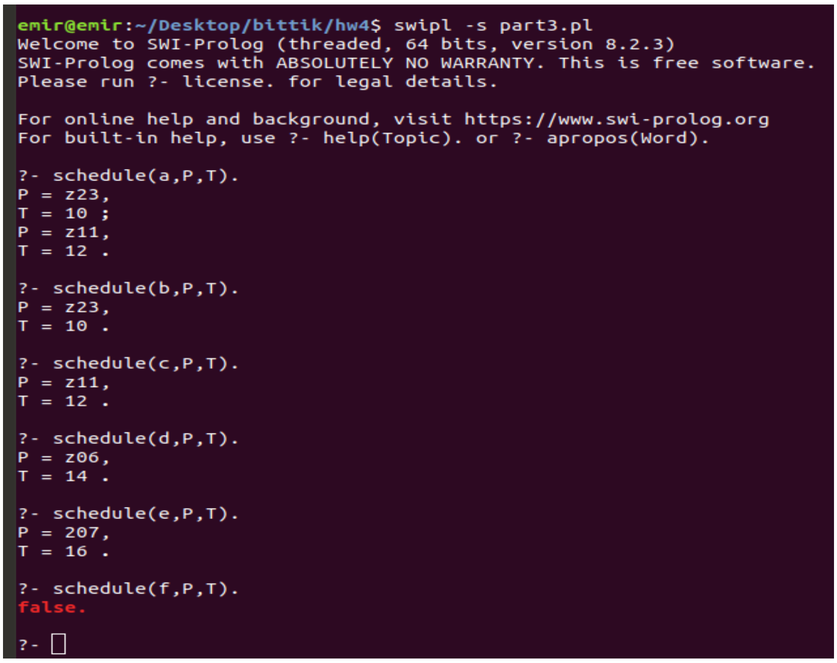
```
% facts..
when(102,10).
```



```
%facts..
when(102, 10).                %Time of the course X is Y
when(108, 12).
when(341, 14).
when(455, 16).
when(452, 17).
where(102, z23).              %Place of the course X is Y
where(108, z11).
where(341, z06).
where(455, 207).
where(452, 207).
enroll(a,102).                %Student X is enrolled in course Y
enroll(a,108).
enroll(b,102).
enroll(c,108).
enroll(d,341).
enroll(e,455).
```

The explanations about the rules are in the source code file. This facts are, when(X,Y) means that classes time of the course X is Y. where(X,Y) means that place of the course X is Y , and “enroll(X,Y) – student X is enrolled in course Y . The examples are below :

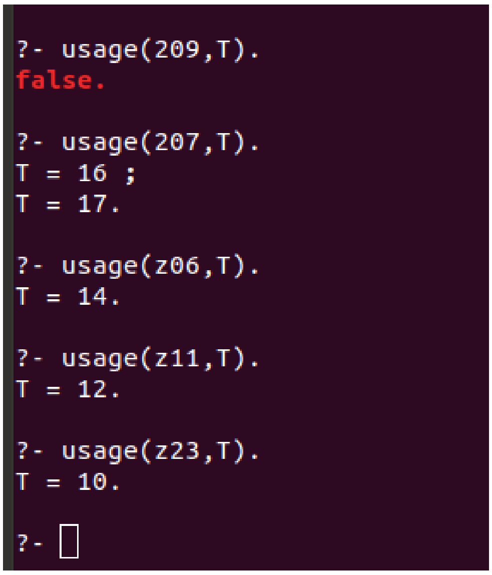
PART3.1



```
emir@emir:~/Desktop/bittik/hw4$ swipl -s part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://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 .

?- schedule(f,P,T).
false.

?- 
```

This means for instance, student a(b,c,d,e,f) in P class at T time.

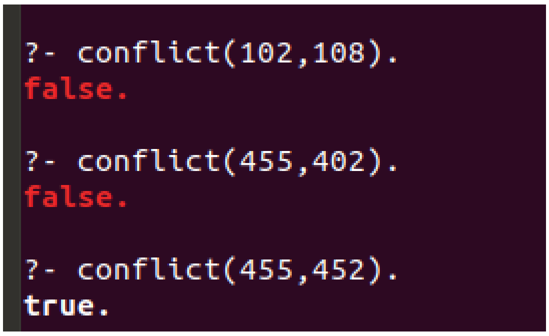
PART3.2



```
?- usage(209,T).
false.

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

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

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

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

?- 
```

This means that, the class 209(207,z06,z11,z23) is used at T time.

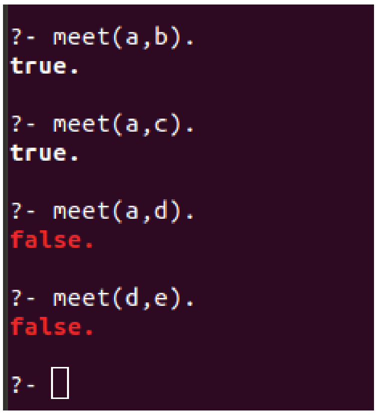
PART3.3



```
?- conflict(102,108).
false.

?- conflict(455,402).
false.

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

“conflict(X,Y)” that gives true if X and Y conflicts due to classroom or time.

PART3.4



```
?- meet(a,b).
true.

?- meet(a,c).
true.

?- meet(a,d).
false.

?- meet(d,e).
false.

?- 
```

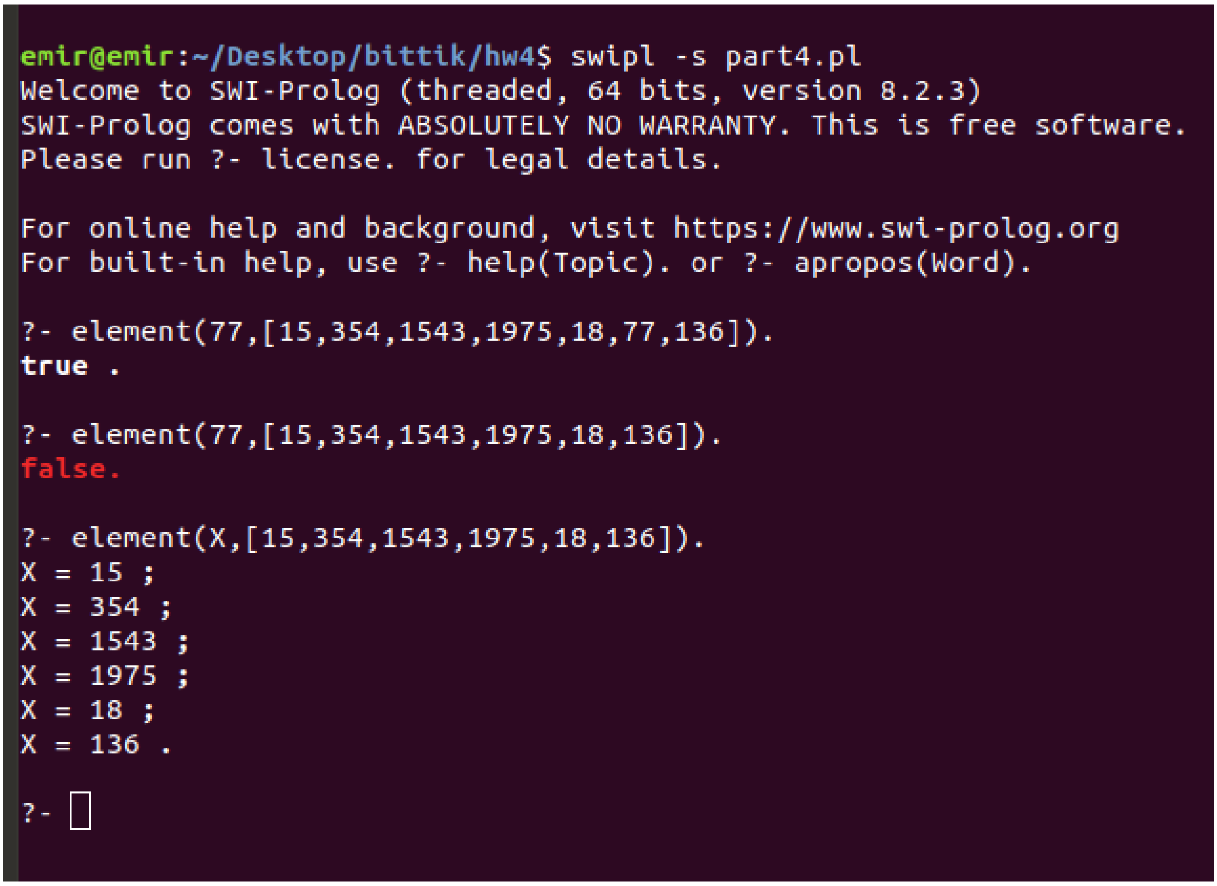
“meet(X,Y)” that gives true if student X and student Y are present in the same classroom at the same time

PART4

1 – swipl -s part4.pl

PART4.1

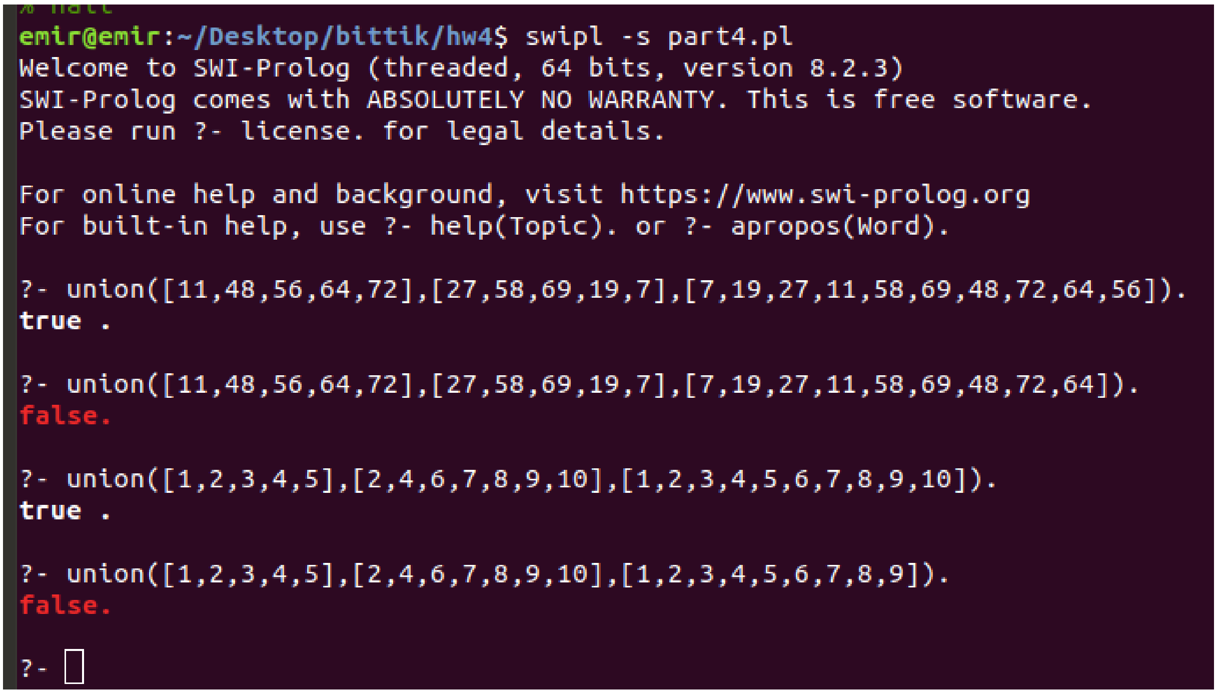
Define a Prolog predicate “element(E,S)” that returns true if E is in S.



```
emir@emir:~/Desktop/bittik/hw4$ swipl -s part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- element(77,[15,354,1543,1975,18,77,136]).
true .

?- element(77,[15,354,1543,1975,18,136]).
false.

?- element(X,[15,354,1543,1975,18,136]).
X = 15 ;
X = 354 ;
X = 1543 ;
X = 1975 ;
X = 18 ;
X = 136 .

?- 
```

PART4.2

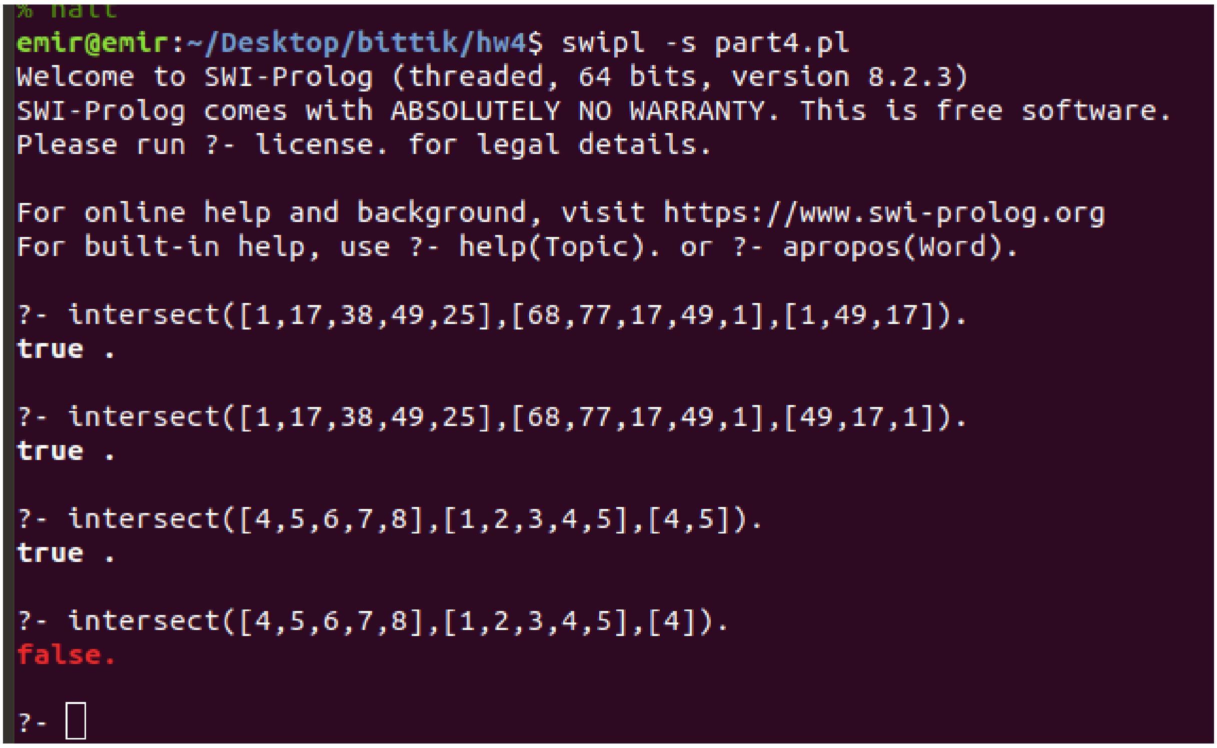
“union(S1,S2,S3)” that returns true if S3 is the union of S1 and S2.



```
% halt
emir@emir:~/Desktop/bittik/hw4$ swipl -s part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- union([11,48,56,64,72],[27,58,69,19,7],[7,19,27,11,58,69,48,72,64,56]).
true .

?- union([11,48,56,64,72],[27,58,69,19,7],[7,19,27,11,58,69,48,72,64]).
false.

?- union([1,2,3,4,5],[2,4,6,7,8,9,10],[1,2,3,4,5,6,7,8,9,10]).
true .

?- union([1,2,3,4,5],[2,4,6,7,8,9,10],[1,2,3,4,5,6,7,8,9]).
false.

?- 
```

PART4.3

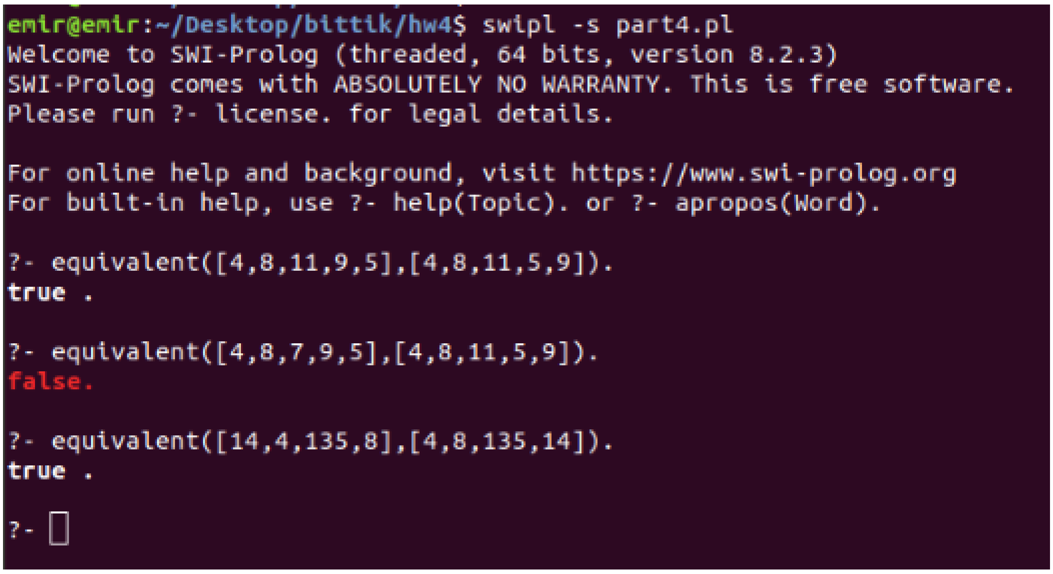
“intersect(S1,S2,S3)” that returns true if S3 is the intersection of of S1 and S2.



```
% halt
emir@emir:~/Desktop/bittik/hw4$ swipl -s part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- intersect([1,17,38,49,25],[68,77,17,49,1],[1,49,17]).
true .

?- intersect([1,17,38,49,25],[68,77,17,49,1],[49,17,1]).
true .

?- intersect([4,5,6,7,8],[1,2,3,4,5],[4,5]).
true .

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

?- 
```

PART4.4

“equivalent(S1,S2)” that returns true if S1 and S2 are equivalent sets.

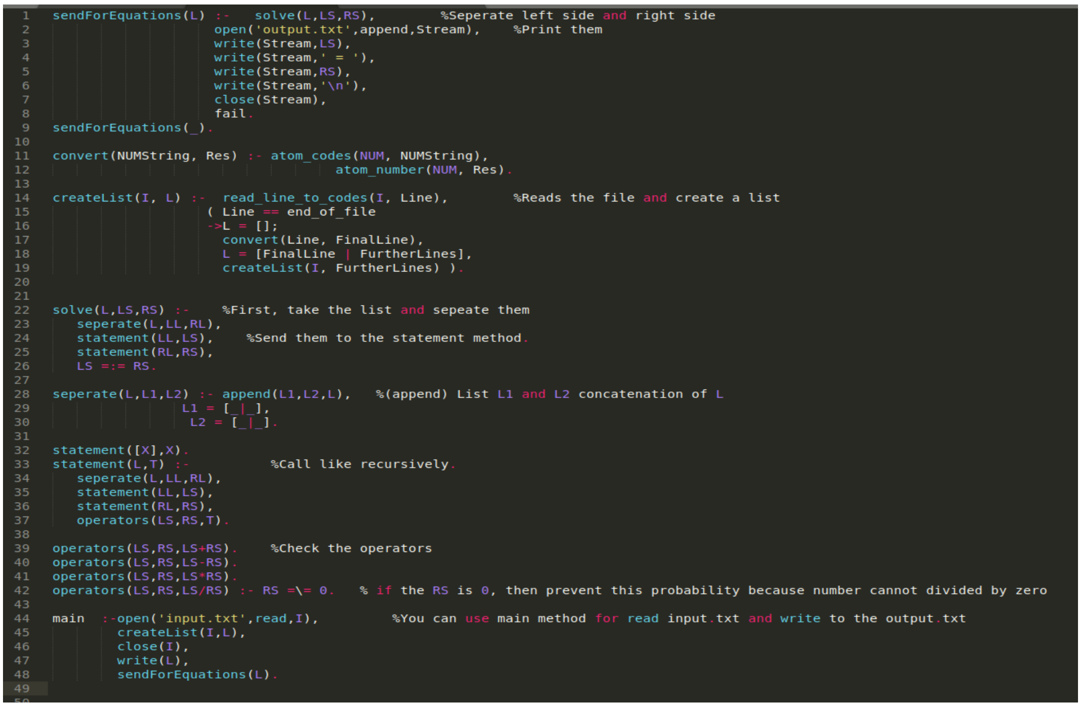


```
emir@emir:~/Desktop/bittik/hw4$ swipl -s part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- equivalent([4,8,11,9,5],[4,8,11,5,9]).
true .

?- equivalent([4,8,7,9,5],[4,8,11,5,9]).
false.

?- equivalent([14,4,135,8],[4,8,135,14]).
true .

?-
```

PART5

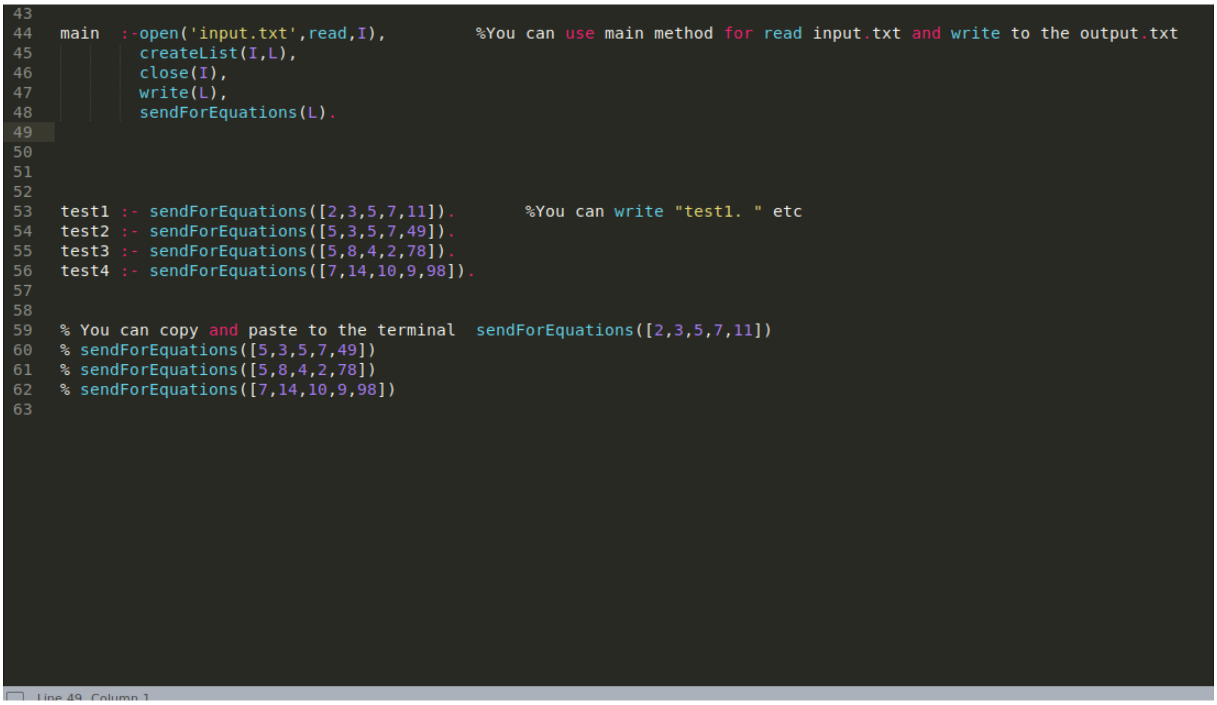
1 – swipl -s part5.pl

Given a list of integes, find a correct way of inserting arithmetic (operators) such that the result is a correct equation. Example: With the list of numbers [5,3,5,7,49] we can form the equations (5-3+5*7) = 11. Please pay attention to the **arithmetic operator precedence**



```
sendForEquations(L) :-   solve(L,LS,RS),        %Seperate left side and right side
                        open('output.txt',append,Stream),    %Print them
                        write(Stream,LS),
                        write(Stream,' = '),
                        write(Stream,RS),
                        write(Stream,'\n'),
                        close(Stream),
                        fail.
sendForEquations(_).

convert(NUMString, Res) :- atom_codes(NUM, NUMString),
                                   atom_number(NUM, Res).

createList(I, L) :-  read_line_to_codes(I, Line),        %Reads the file and create a list
                       ( Line == end_of_file
                       ->L = [];
                         convert(Line, FinalLine),
                         L = [FinalLine | FurtherLines],
                         createList(I, FurtherLines) ).


solve(L,LS,RS) :-    %First, take the list and sepeate them
   seperate(L,LL,RL),
   statement(LL,LS),    %Send them to the statement method.
   statement(RL,RS),
   LS =:= RS.

seperate(L,L1,L2) :- append(L1,L2,L),   %(append) List L1 and L2 concatenation of L
                    L1 = [_|_],
                     L2 = [_|_].

statement([X],X).
statement(L,T) :-          %Call like recursively.
   seperate(L,LL,RL),
   statement(LL,LS),
   statement(RL,RS),
   operators(LS,RS,T).

operators(LS,RS,LS+RS).    %Check the operators
operators(LS,RS,LS-RS).
operators(LS,RS,LS*RS).
operators(LS,RS,LS/RS) :- RS =\= 0.   % if the RS is 0, then prevent this probability because number cannot divided by zero

main  :-open('input.txt',read,I),         %You can use main method for read input.txt and write to the output.txt
        createList(I,L),
        close(I),
        write(L),
        sendForEquations(L).
```

This is my source code file.



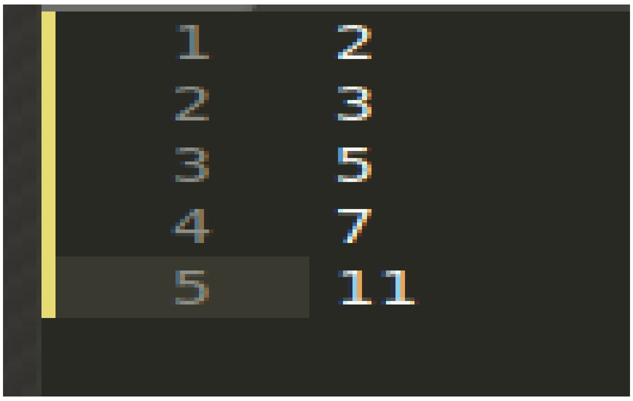
```
main  :-open('input.txt',read,I),         %You can use main method for read input.txt and write to the output.txt
        createList(I,L),
        close(I),
        write(L),
        sendForEquations(L).




test1 :- sendForEquations([2,3,5,7,11]).       %You can write "test1. " etc
test2 :- sendForEquations([5,3,5,7,49]).
test3 :- sendForEquations([5,8,4,2,78]).
test4 :- sendForEquations([7,14,10,9,98]).


% You can copy and paste to the terminal  sendForEquations([2,3,5,7,11])
% sendForEquations([5,3,5,7,49])
% sendForEquations([5,8,4,2,78])
% sendForEquations([7,14,10,9,98])
```

This shows how you can execute the program with different ways . I did this part like 3 different ways. If you can execute the program, after you compile like “swipl -s part5.l” you can write these :

1 – main. (It read the input file and write to the output file)

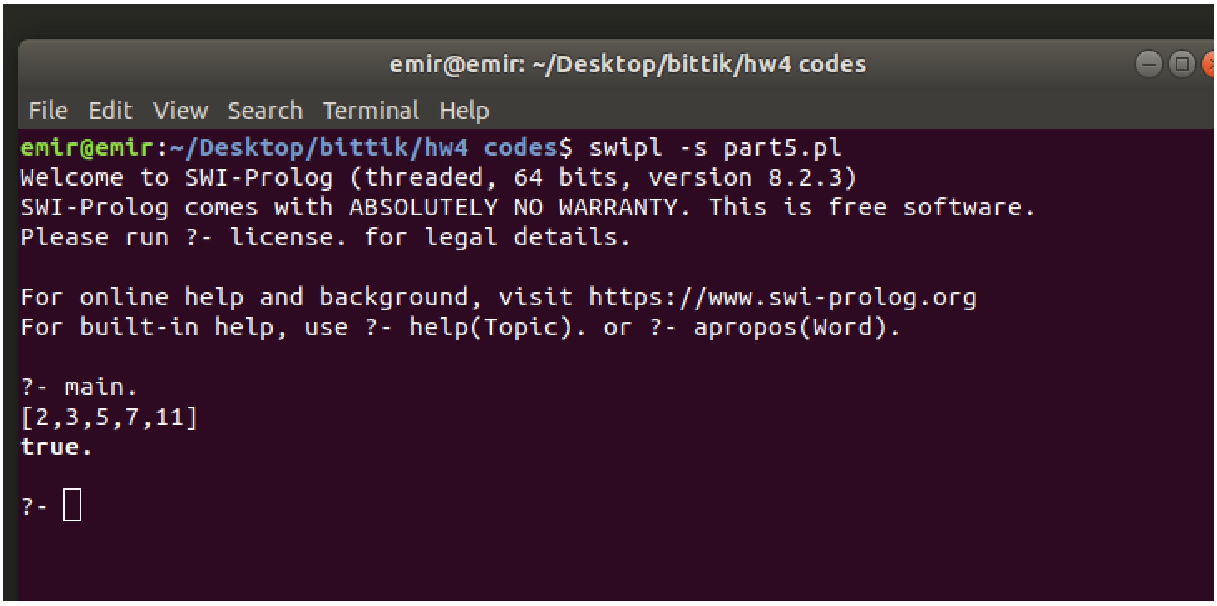
2 – sendForEquations(List) (It takes the list and write to the output file)

3 – testX. (It execute the test lists and write to the output file)



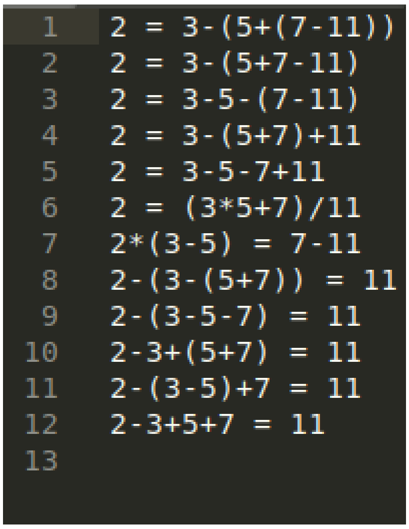
```
2
3
5
7
11
```

This is my input.txt file. You have to write the list elements one by one for each line.


emir@emir: ~/Desktop/bittik/hw4 codes
File Edit View Search Terminal Help

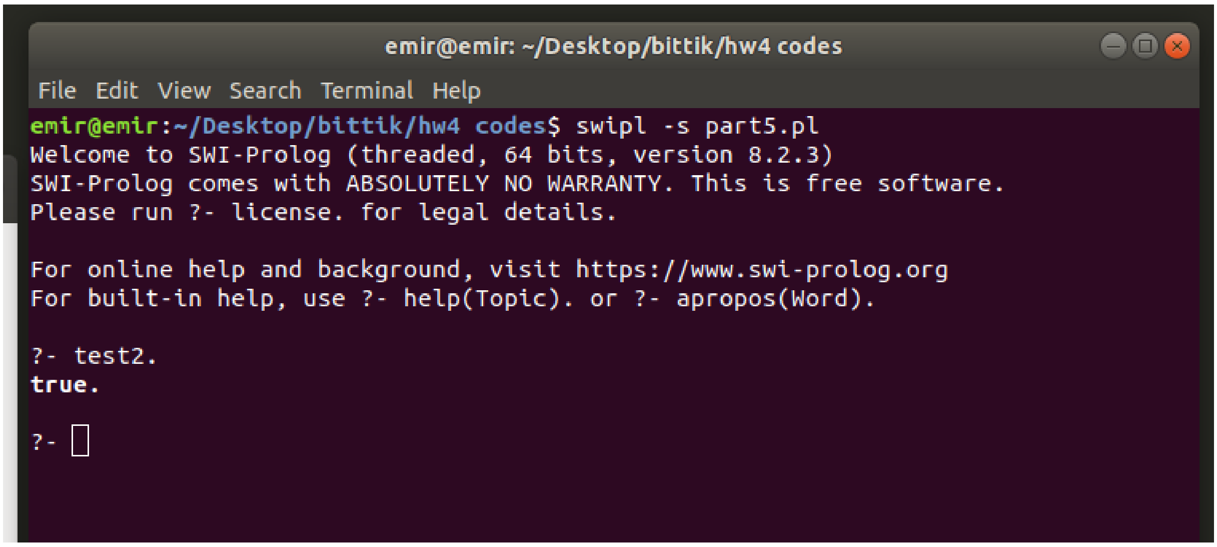
```
emir@emir:~/Desktop/bittik/hw4 codes$ swipl -s part5.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- main.
[2,3,5,7,11]
true.

?- 
```

The first way, you can enter main. And the output is in output.txt like that :



```
2 = 3-(5+(7-11))
2 = 3-(5+7-11)
2 = 3-5-(7-11)
2 = 3-(5+7)+11
2 = 3-5-7+11
2 = (3*5+7)/11
2*(3-5) = 7-11
2-(3-(5+7)) = 11
2-(3-5-7) = 11
2-3+(5+7) = 11
2-(3-5)+7 = 11
2-3+5+7 = 11
```

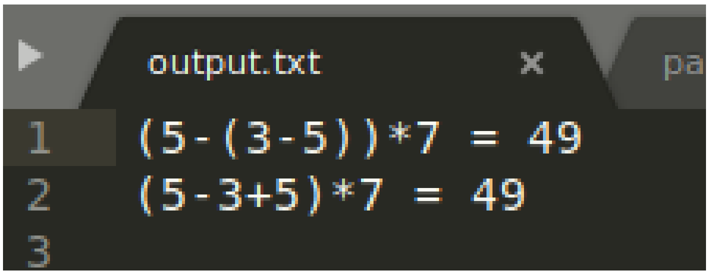
The other way is;



emir@emir: ~/Desktop/bittik/hw4 codes
File Edit View Search Terminal Help

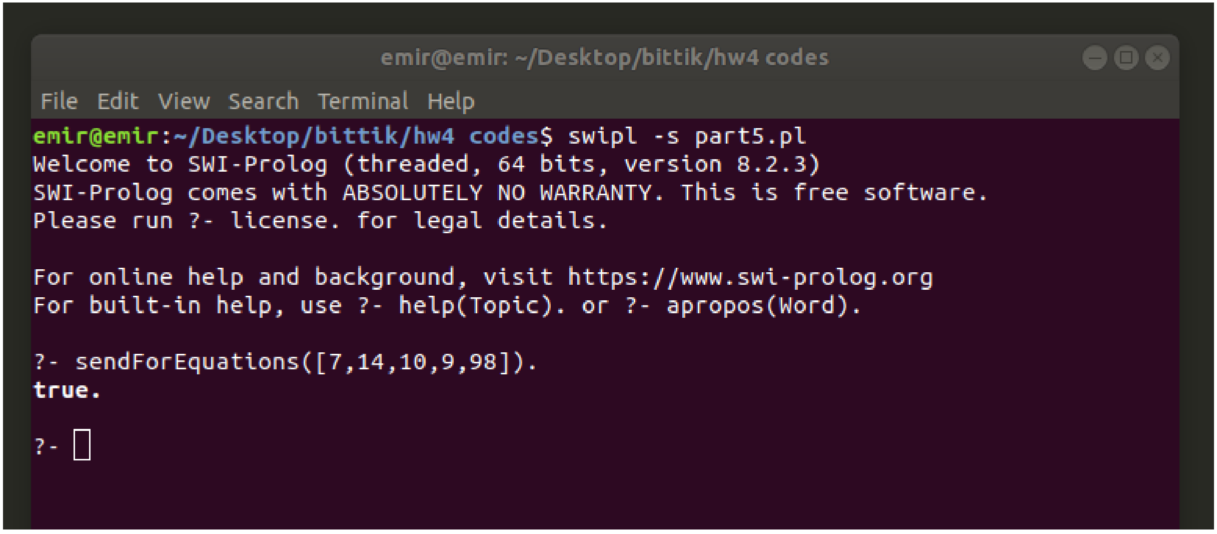
```
emir@emir:~/Desktop/bittik/hw4 codes$ swipl -s part5.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- test2.
true.

?-
```

You can enter the test2. The test lists are in the source code file. You can see or you can enter new test list for yourself. The test2 list is [5,3,5,7,49]. So the output is in txt file like that :



```
(5-(3-5))*7 = 49
(5-3+5)*7 = 49
```

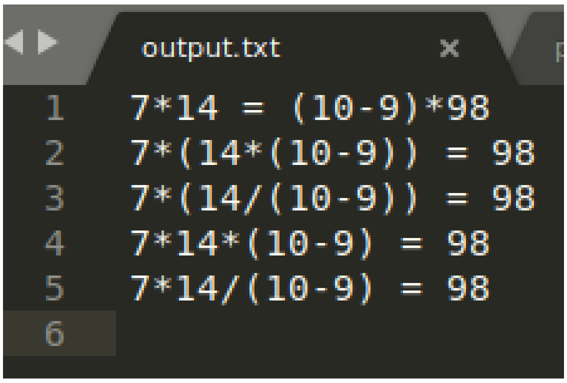
The last way is, you can write the your own list with function name.



emir@emir: ~/Desktop/bittik/hw4 codes
File Edit View Search Terminal Help

```
emir@emir:~/Desktop/bittik/hw4 codes$ swipl -s part5.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- sendForEquations([7,14,10,9,98]).
true.

?-
```

The list is above that. And the output is :



```
7*14 = (10-9)*98
7*(14*(10-9)) = 98
7*(14/(10-9)) = 98
7*14*(10-9) = 98
7*14/(10-9) = 98
```