CSE341 – Programming Languages

Homework #4

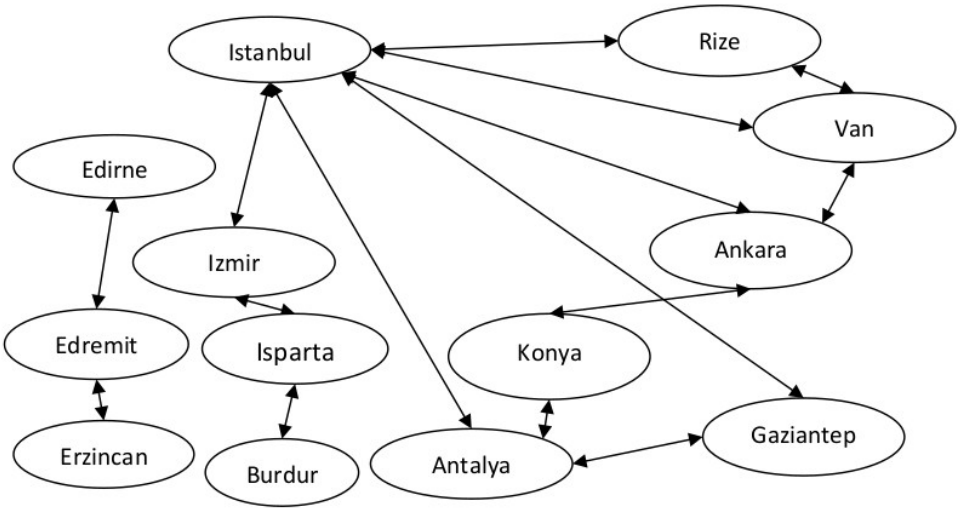
Course Teacher : Yakup Genç

Assistant : Muhammed Ali Dede

Melih Yabaş

161044072

Part1



All possiple flights created as knowledge base.

```
flight(istanbul,izmir).
flight(istanbul,antalya).
flight(istanbul,gaziantep).
flight(istanbul,ankara).
flight(istanbul,van).
flight(istanbul,rize).

flight(edirne,edremit).

flight(edremit,edirne).
flight(edremit,erzincan).

flight(erzincan,edremit).

flight(burdur,ısparta).

flight(ısparta,burdur).
flight(ısparta,izmir).

flight(izmir,ısparta).
flight(izmir,istanbul).

flight(antalya,istanbul).
flight(antalya,konya).
flight(antalya,gaziantep).

flight(gaziantep,istanbul).
flight(gaziantep,antalya).

flight(konya,antalya).
flight(konya,ankara).

flight(ankara,konya).
flight(ankara,istanbul).
flight(ankara,van).

flight(van,ankara).
flight(van,istanbul).
flight(van,rize).

flight(rize,van).
flight(rize,istanbul).
```

Direct flights described. If we test them;

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

?- flight(istanbul, burdur).
false.
```

There is a direct flight from istanbul to antalya.

There is no direct flight from istanbul to burdur.

However, there is a route between them.

Rules written to determine routes.

```
%rules..
route(X, Y) :- allRoutes(X, Y, []).

allRoutes(X, Y, Visited) :- flight(X, Z), not(member(Z, Visited))
                          , (Y = Z; allRoutes(Z, Y, [X | Visited])).
```

```
|    route(istanbul, burdur).
true .

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

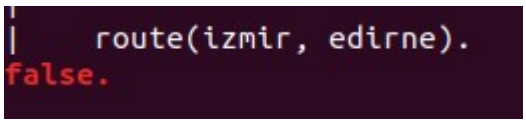
And there are many routes from istanbul to other cities.

Also, all the possible routes are active now.

If there is a no direct flight, It checks whether the transfer exists. This transfer must be Z to Y. X is added to the list because is visited.

For example;

There is no route from izmir to edirne.



Part2

Knowledge base expanded by adding distances for the direct flights. Distances gotten from “https://www.distancecalculator.net”

```
%facts..
distance(istanbul,izmir,329).
distance(istanbul,antalya,483).
distance(istanbul,gaziantep,847).
distance(istanbul,ankara,352).
distance(istanbul,van,1262).
distance(istanbul,rize,968).

distance(edirne,edremit,244).

distance(edremit,edirne,244).
distance(edremit,erzincan,1027).

distance(erzincan,edremit,1027).

distance(burdur,ısparta,25).

distance(ısparta,burdur,25).
distance(ısparta,izmir,309).

distance(izmir,ısparta,309).
distance(izmir,istanbul,329).

distance(antalya,istanbul,483).
distance(antalya,konya,192).
distance(antalya,gaziantep,592).

distance(gaziantep,istanbul,847).
distance(gaziantep,antalya,592).

distance(konya,antalya,192).
distance(konya,ankara,227).

distance(ankara,konya,227).
distance(ankara,istanbul,352).
distance(ankara,van,920).
```

```
?- sroute(rize, konya, X).
X = 1520 .

?- sroute(van, antalya, X).
X = 1339 .
```

Closest path from rize to konya is 1520.
Closest path from van to antalya is 1339.

```
|
|    sroute(istanbul, edirne, X).
false.

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

There is no path from istanbul to edirne.
There is no path from rize to edremit.

Part3

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

```
%facts..
when(102, 10).
when(108, 12).
when(341, 14).
when(455, 16).
when(452, 17).

where(102, z23).
where(108, z11).
where(341, z06).
where(455, 207).
where(452, 207).

enroll(a,102).
enroll(a,108).
enroll(b,102).
enroll(c,108).
enroll(d,341).
enroll(e,455).
```

Fact are written according to the table.

3.1

“schedule(S,P,T)” associates a student to a place and time of class.

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

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

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

?- schedule(t,P,T).
false.
```

3.2

"usage(P,T)" gives the usage times of a classroom. See the example query and its result.

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

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

?- usage(11,T).
false.
```

3.3

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

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

?- conflict(102,452).
false.
```

3.4

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

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

?- meet(a,e).
false.
```

Part4

```
element(E, [E|_]).
element(E, [_|S]):- element(E, S).

union(S1,S2,S3) :- unionX(S1,S2,X), equivalent(X,S3).
unionX([], S2, S2).
unionX([E|S1], S2, S3) :- element(E, S2), !, unionX(S1, S2, S3).
unionX([E|S1], S2, [E|S3]) :- unionX(S1, S2, S3).

equivalent(S1, S2) :- equivalentX(S1,S2), equivalentX(S2,S1).
equivalentX([],_).
equivalentX([E|S1],S2):- element(E,S2), equivalentX(S1,S2).

intersect(S1,S2,S3) :- intersectX(S1,S2,X), equivalent(X,S3).
intersectX([], _, []).
intersectX([E|S1], S2, [E|S3]):- element(E, S2), !, intersectX(S1, S2, S3).
intersectX([_|S1], S2, S3):- intersectX(S1, S2, S3).
```

All the predicates writed operating on sets.

4.1

element(E,S) returns true if E is in S.
element(E, [E|_]) if element exist in head of list, return true.
element[E, [_|S]) :- element(E,S) Recursive call with tail of list.

```
?- element(2, [3,7,11,2,1]).
true .

?- element(8, [3,2,1]).
false.
```

```
?- element(X, [87,64,2,3,4]).
X = 87 ;
X = 64 ;
X = 2 ;
X = 3 ;
X = 4 ;
false.
```

4.2

union(S1,S2,S3) returns true if S3 is the union of S1 and S2. S1, S2 and S3 are list.

union(S1,S2,S3) :- union2(S1,S2,X), equivalent(X,S3). Assume that the predict union set be in random order.

union2(S1,S2,X) X will be union of S1,S2.

equivalent(X,S3) whether union result X is equivalent entered union S3. If they are equivalent then return true, else return false.

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

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

4.3

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

```
|    intersect([1,2,3,4],[3,4,5],[3,4]).
true .

?-
|    intersect([1,2,9,3,4],[4,5],[3,4]).
false.
```

4.4

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

```
|    equivalent([5,6,1,7],[1,5,6,7]).
true .

?-
|    equivalent([5,9,1,8],[10,5,6,7]).
false.
```

Part5

This program finds correct ways of inserting arithmetic operators such that is a correct equation.

```
mergeEq(List,Left,Right) :-
    splitLists(List,Left1,Right1),
    term(Left1,Left),
    term(Right1,Right),
    Left =:= Right.
```

MergeEq gets an equation list and splits into 2 different list using its numbers and creates correct equations

```
main :-
    open('input.txt',read,Str),
    read(Str,equat),
    close(Str),
    open('output.txt',write,Stream),
    write(Stream, ""),
    close(Stream),
    mergeEq(equat,LeftSide,RightSide),
    open('output.txt',append,Out),
    write(Out, LeftSide = RightSide ),
    write(Out, "\n"),
    close(Out),
    fail.
```

First, open the input file and read the content into a list and close the file. Then, open the output file and empty it. After that, make the calculation and write into output file. Close the file and finish the program.

How to run?

→ Open prolog from terminal in the directory.

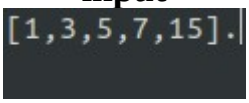
```
melih@melih-SATELLITE-L50-C:~/Desktop/plHW4$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

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

?- consult('part5.pl').
true.

?- main.
false.
```

→ consult(‘part5.pl’).

→ main.

Then, it will read the input from “input.txt” and it will write the correct equations to the “output.txt”

input



output