Homework 4 Report File

Part1:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **route(X,Y).** or you can give any other constant for X and Y parameter like **route(istanbul,Y).** or **route(istanbul,rize).** or **route(X, rize).**
  If query has no unknown variable like **route(istanbul,rize).** it will return true or false depend on the knowledge base. Otherwise it will return what unknown variables can be and to see another possibilities you should press **;** in SWI-Prolog.

Examples:

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

?- route(edirne,erzincan).
true .

?- route(edirne,istanbul).
false.
```

Part2:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **sroute(edremit,erzincan,X).**
  It will return X as shortest path between two given cities

Examples:

```
?- sroute(edirne,erzincan,X).
X = 1301.

?- sroute(istanbul,konya,X).
X = 579 .
```

Part3:

3.1:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **schedule(S,P,T).** where S is student, P is place and T is time. If query has no unknown variable like **schedule(a,102,10).** it will return true or false depend on the knowledge base. Otherwise it will return what unknown variables can be and to see other possibilities you should press **;** in SWI-Prolog.

Examples:

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

?- schedule(a,z23,10).
true .

?- schedule(a,z23,5).
false.
```

3.2:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **usage(P,T).** where P is place and T is time. If query has no unknown variable like **usage (z23,10).** it will return true or false depend on the knowledge base. Otherwise it will return what unknown variables can be and to see other possibilities you should press **;** in SWI-Prolog.

Examples:

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

?- usage(z23,5).
false.

?- usage(z07,T).
T = 16 ;
T = 17.
```

3.3:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **conflict(X,Y).** where X and Y is class.
  it will return true if there is a conflict, otherwise return false

Examples:

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

?- conflict(455,102).
false.
```

3.4:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **meet(X,Y).** where X and Y is student.
  it will return true if student X and student Y are present in the same classroom at the same time.

Examples:

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

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

Part4:

4.1:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **element(E,S).**
  it will return true if E is in S.

Examples:

```
?- element(5, [2,3,5]).
true .

?- element(5, [2,3,4]).
false.
```

4.2:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **union(S1,S2,S3).**
  it will return true if S3 is the union of S1 and S2.

Examples:

```
?- union([1,2],[5,3,7],[1,2,3,5,7]).
true .

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

4.3:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **intersect(S1,S2,S3).**
  it will return true if S3 is the intersect of S1 and S2.

Examples:

```
?- intersect([1,2,3,5],[5,3,7],[3,5]).
true .

?- intersect([1,2,3,5],[5,3,7],[3,5,7]).
false.
```

4.4:

Implementation:

- İt is done as ask for in the pdf

Execution:

- After compiled it, you can type **equivalent(S1,S2).**
  it will return true if S1 and S2 are equivalent.

Examples:

```
?- equivalent([1,2,5],[1,2,5]).
true .

?- equivalent([1,2,5],[1,2,5,7]).
false.
```

Part5:

Implementation:

- İt is done as ask for in the pdf
- İt reads numbers form of [number,number…..] as specified in the pdf from input.txt file
- İt writes the result equation to the output.txt file
- Because of SWI-Prolog, output.txt and input.txt must be on the prolog's working directory. (in my computer belgeler/prolog)

Execution:

- After compiled it, you can type **program().**
  it will write the equation to the output.txt file.

Examples:

Input: [2,4,5,8,-26]

Output: 2-4*5-8=-26