# CSE341 – Programming Languages (Fall 2020)

# Homework #4

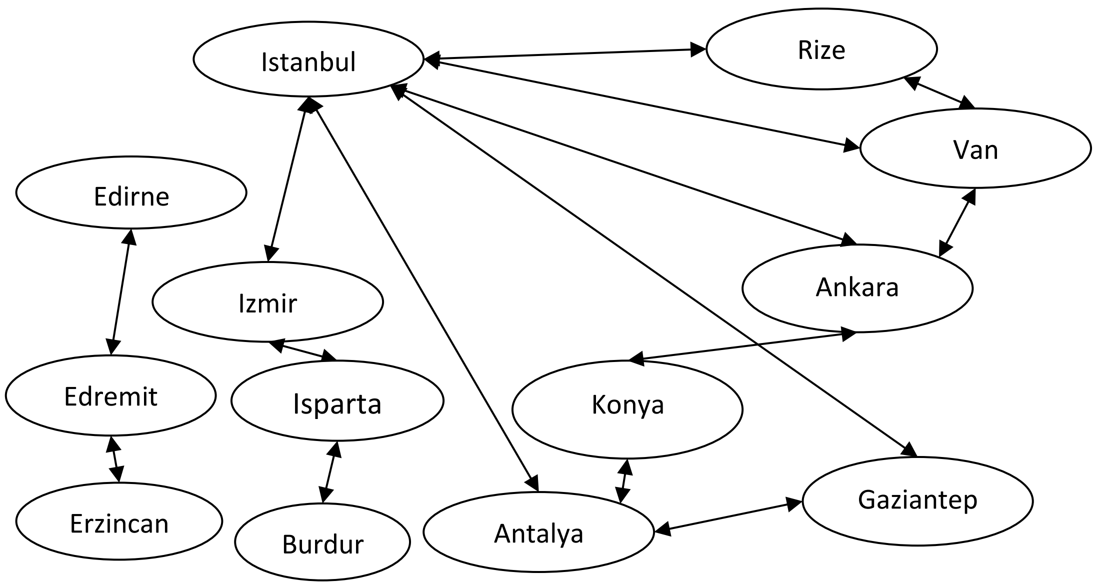
**Handed out**: December 30, 2020.

**Due**: 11:55pm January 10, 2021.

**Hand-in Policy**: Source code and documentation should be submitted online as a single .zip. File with naming convention studentid_lastname_firstname_hw4.zip via Moodle by the submission deadline. No late submissions will be accepted. Each part of the homework will be implemented in a single file named **partx.pl.** Thus we expect 6 different files in this homework.
**Collaboration Policy**: No collaboration is permitted. Any cheating (copying someone elses work in any form) will result in a grade of -100 for the first offense and -200 for the subsequent attempts.
**Grading**: Each homework will be graded on the scale 100. Unless otherwise noted, the questions/parts will be weighed equal.

---

**Part 1.** In the graph below you 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.



Your program should have all the facts and predicates/rules. See the following:

```
% knowledge base
…
flight(istanbul,antalya).  % the fact that Istanbul and Antalya has a flight.
…
% rules
…
route(X,Y) :- flight(X,Y). % a predicate indicating there exist a route between
                           % X and Y if there is flight between X and Y.
…
```

A single query to complete your program should check if there is a direct route between two given cities. Alternatively, it can list all the connected cities for a given city. See the following:

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

Make sure that your predicate implementation handles cycles properly avoiding infinite loops.

**Part 2.** Continuing with the previous problem, you are asked to write a program that checks if a route exists between two cities and if so, provides the shortest route.

In the first step, you are to expand the knowledge by adding distances for the direct flights. E.g.,

```
% knowledge base
…
flight(istanbul, antalya).  % the fact that Istanbul and Antalya has a flight.
distance(istanbul, antalya, 481).  % flight distance – calculated using
                                    % https://www.distancecalculator.net
% complete all the flights and distances …
…
```

A query to complete your program should check if there is a direct route between two given cities and the shortest distance between them. See the following example:

```
?- sroute(edremit,erzincan,X).
X = 1044 ;
```

**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).
```

3.1. Define/write a predicate “schedule(S,P,T)” that associates a student to a place and time of class. See the example query and its result.

```
?- schedule(a,P,T).
P = 102
T = 10 ;


P = 108
T = 12 ;
```

3.2. Define/write another predicate “usage(P,T)” that gives the usage times of a classroom. See the example query and its result.

```
?- usage(207,T).
T = 455 ;
T = 456 ;
```

3.3. Define/write another predicate “conflict(X,Y)” that gives true if X and Y conflicts due to classroom or time.

3.4. Define/write another predicate “meet(X,Y)” that gives true if student X and student Y are present in the same classroom at the same time.

**Part 4.** Write the following predicates operating on sets.

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

4.2. Define a Prolog predicate “union(S1,S2,S3)” that returns true if S3 is the union of S1 and S2.

4.3. Define a Prolog predicate “intersect(S1,S2,S3)” that returns true if S3 is the intersection of of S1 and S2.

4.3. Define a Prolog predicate “equivalent(S1,S2)” that returns true if S1 and S2 are equivalent sets.

**Part 5.** 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.**

**Part 6.** In this part you will solve a puzzle, Essentially, each row and column of a rectangular bitmap is annotated with the respective lengths of its distinct strings of occupied cells. The person who solves the puzzle must complete the bitmap given only these lengths.

```
Problem statement:          Solution:

|_|_|_|_|_|_|_|_| 3         |_|X|X|X|_|_|_|_| 3
|_|_|_|_|_|_|_|_| 2 1       |X|X|_|X|_|_|_|_| 2 1
|_|_|_|_|_|_|_|_| 3 2       |_|X|X|X|_|_|X|X| 3 2
|_|_|_|_|_|_|_|_| 2 2       |_|_|X|X|_|_|X|X| 2 2
|_|_|_|_|_|_|_|_| 6         |_|_|X|X|X|X|X|X| 6
|_|_|_|_|_|_|_|_| 1 5       |X|_|X|X|X|X|X|_| 1 5
|_|_|_|_|_|_|_|_| 6         |X|X|X|X|X|X|_|_| 6
|_|_|_|_|_|_|_|_| 1         |_|_|_|_|X|_|_|_| 1
|_|_|_|_|_|_|_|_| 2         |_|_|_|X|X|_|_|_| 2
 1 3 1 7 5 3 4 3             1 3 1 7 5 3 4 3
 2 1 5 1                     2 1 5 1
```

For the example above, the problem can be stated as the two lists [[3],[2,1],[3,2],[2,2],[6],[1,5],[6],[1],[2]] and [[1,2],[3,1],[1,5],[7,1],[5],[3],[4],[3]] which give the "solid" lengths of the rows and columns, top-to-bottom and left-to-right, respectively.

**Test Cases: Top-to-bottom or Left-to-right**

1. ([[3], [2,1], [3,2], [2,2], [6], [1,5], [6], [1], [2]],
   [[1,2], [3,1], [1,5], [7,1], [5], [3], [4], [3]])
2. ( [[3,1], [2,4,1], [1,3,3], [2,4], [3,3,1,3], [3,2,2,1,3], [2,2,2,2,2], [2,1,1,2,1,1], [1,2,1,4], [1,1,2,2], [2,2,8], [2,2,2,4], [1,2,2,1,1,1], [3,3,5,1], [1,1,3,1,1,2], [2,3,1,3,3], [1,3,2,8], [4,3,8], [1,4,2,5], [1,4,2,2], [4,2,5], [5,3,5], [4,1,1], [4,2], [3,3]],
   [[2,3], [3,1,3], [3,2,1,2], [2,4,4], [3,4,2,4,5], [2,5,2,4,6], [1,4,3,4,6,1], [4,3,3,6,2], [4,2,3,6,3], [1,2,4,2,1], [2,2,6], [1,1,6], [2,1,4,2], [4,2,6], [1,1,1,1,4], [2,4,7], [3,5,6], [3,2,4,2], [2,2,2], [6,3]])

3. ( [[5], [2,3,2], [2,5,1], [2,8], [2,5,11], [1,1,2,1,6], [1,2,1,3],[2,1,1], [2,6,2], [15,4], [10,8], [2,1,4,3,6], [17], [17], [18], [1,14], [1,1,14], [5,9], [8], [7]], [[5], [3,2], [2,1,2], [1,1,1], [1,1,1], [1,3], [2,2], [1,3,3], [1,3,3,1], [1,7,2], [1,9,1], [1,10], [1,10], [1,3,5], [1,8], [2,1,6],[3,1,7], [4,1,7], [6,1,8], [6,10], [7,10], [1,4,11], [1,2,11], [2,12], [3,13]] )