Assignment Set 2 (Logic programming using Prolog)

- **Assignment will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions), complete source codes and executable files.**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F for the whole course, irrespective of your performance in the other parts of the lab.**

- **Submission deadline: 15th October, 2017**

- **Total weight = 20%**
- **Marks distribution: 20, 30, 50**

Problem 1: Finding Relationship and Gender (20 Marks)

Consider a database that describe PARENT relationships as well as GENDER relationships. In this database, you can have predicate like parent(jatin,avantika) that interprets as "Jatin is a parent of Avantika". The predicate male(jatin) interprets as "Jatin is a man". Similarly, the predicate female(avantika) interprets as "Avantika is a woman". An example database of facts is:

```
parent(jatin,avantika).
parent(jolly,jatin).
parent(jolly,kattappa).
parent(manisha,avantika).
parent(manisha,shivkami).
parent(bahubali,shivkami).
male(kattappa).
male(jolly).
female(shivkami).
female(avantika).
male(bahubali).
```

Observe that few things are not specified in the database (for instance, male(jatin)).

Question1: UNCLE (10 Marks)

Your task is to write a Prolog program to determine UNCLE relationship using the above two types of facts only. You should not consider uncles "by marriage", i.e., for A to be B's uncle the two must have blood relationship. Your program should be able to generate the following outputs.

```
?- uncle(katappa,avantika).
Yes
```

```
?- uncle(avantika,manisha).
No

?- uncle(katappa,A).
A = avantika ;
No

?- uncle(jatin,avantika).
No

?- uncle(A,B).
A = katappa
B = avantika ;
No
```

Question2: HALF SISTER (10 Marks)

In a similar way, write a Prolog program for HALFSISTER[1] relationship. Your program should be able to generate the following outputs.

```
?- halfsister(avantika,shivkami).
Yes

?- halfsister(A,shivkami).
A=avantika ;
No

?- halfsister(A,B).
A=avantika
B=shivkami ;

A=shivkami
B=avantika ;
No
```

[1] *Half Sister* means a sister with whom one has only one parent in common

Problem 2: Searching for the Details of a Research Scholar (30 Marks)

Create two lists as follows for containing 10 research scholars' data.

roll_number	name	e-mail_address	type[2]

roll_number	name	type	supervisor	co_supervisor[3]

Write a Prolog program to search for a scholar's full details derived from the above two lists.

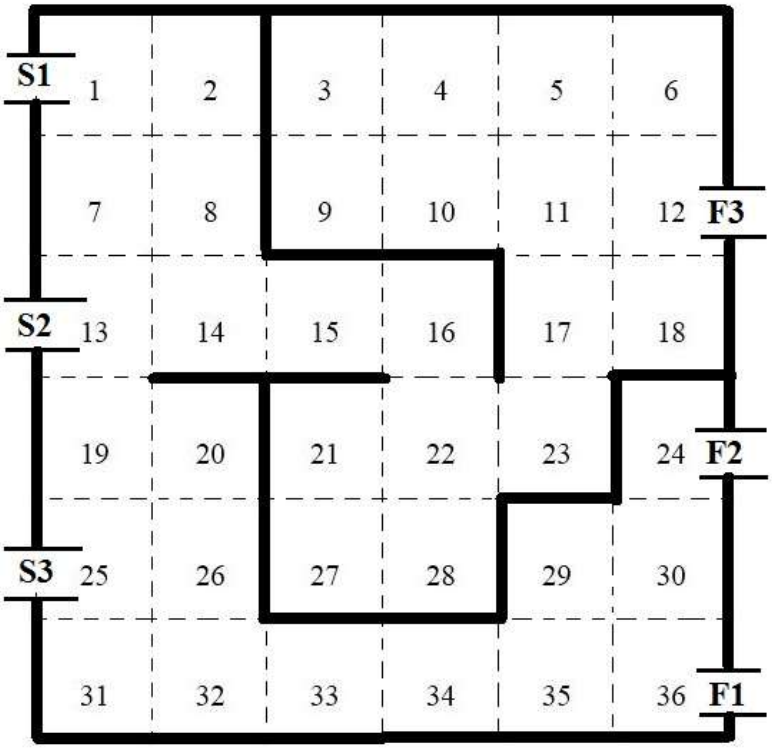
After searching the detail of a particular research scholar, either by her/his name or by her/his roll number, the output should be shown in the following list format (user should have the option to search for a research scholars' detail both by their roll number as well as by their name).

roll_number	name	e-mail_address	supervisor	co_supervisor	type

[2] *type* specifies whether a research scholar is regular/part time/QIP/foreign student.
[3] *co_supervisor* may be empty for some student.

Problem 3: Path Finding Problem (50 Marks)

A maze is a complete system of passages or paths separated by walls and hedges. Consider the following maze. Here, S1, S2 and S3 indicate the starting points and F1, F2 and F3 indicate the finishing points. Bold lines are walls, which cannot be crossed. Dashed line can be crossed directly. For instance, one can go to cell 2 from cell 1 directly, but cannot go to cell 3 from cell 2 directly. One cannot directly go to a cell which is diagonal with respect to the present cell. For instance, one cannot go to cell 8 from cell 1. If someone follows a wrong path and cannot find the destination, s/he is allowed to backtrack.



Your task is to,

A. Write a Prolog program to find all the paths from start to finish through the maze for each and every pair of starting and finishing points.
B. Find also the optimal path[4] for each pair of starting and finishing points.
C. Report prolog facts and rules you assumed to solve the problem.

[4] Consider hop count for finding *optimal path*