Computer Science and Engineering Department

Artificial Intelligence (UCS-521)

Lab Assignment-7

<table>
<tr><td>1</td><td>Prolog application that you need to implement is about solving the following murder mystery:

jean was killed on Tuesday; the only suspects are:

Luc, Paul, Alain, Bernard and Louis.

The rules to follow are:

• The murderer is somebody who has a motive to kill jean, who owns a gun, and who does not have an alibi for Tuesday.
• An alibi provided by a person who is not trustworthy is not accepted.
• Somebody has a motive to kill jean if he has a special interest in killing jean or he wants revenge.
• Somebody has a special interest in killing jean is he is the beneficiary of jean’s fortune, or if he owns money to jean, or if jean surprised him committing a crime.

Here are the facts established by the investigation:

Luc has an alibi for Tuesday given by Bernard
Paul has an alibi for Tuesday given by Bernard
Louis has an alibi for Tuesday given by Luc
Alain has an alibi for Thursday given by Luc
Alain is not a trustworthy person
Paul wants to take revenge on Jean
Luc wants to take revenge on Jean
Bernard is the beneficiary of Jean’s fortune
Jean is the beneficiary of Louis’s fortune
Louis owns money to Jean
Luc owns money to Jean
Jean has seen Alain committing a crime
Luc owns a gun
Louis owns a gun
Alain owns a gun

(i) Implement the above facts and rules for describing the problem.
(You choose the names of these predicates.)

(ii) Have Prolog solve the murder mystery, by using the facts and the rules to answer the question: Who killed jean?</td></tr>
<tr><td>2</td><td>The Prolog application that you are going to implement will provide access to a database of students and a database of courses.</td></tr>
</table>

We encode the database of students using a predicate student (one Prolog fact for each student). The first argument the full name of the student and the second argument a list of courses that the student took. The name of a student is a data structure called name with two arguments: the first name and the last name of the student. A course name is simply a symbol such as csi324.

```
student(name(john, smith), [mat101, csi108, csi148, csi270]).
student(name(jim, roy), []).
student(name(jane, brown), [mat101, csi108]).
student(name(emily, white), [mat101, csi108, mat246]).
student(name(emma, smith), [mat101, csi108, csi148, mat246]).
```

We encode the database of courses and their prerequisites as a predicate prereq, with one Prolog fact for each course. The first argument is the name of the course and the second argument is a list of prerequisites required in order to take that course. A prerequisite can be a course name or a list of alternative prerequisites. For example, the fact that csi324 has the prerequisites csi148 and one of csi238 and mat246 is expressed in the last clause below. A list of alternative prerequisite contains two or more alternatives.

```
prereq(csi108, []).
prereq(csi148, [csi108, mat101]).
prereq(csi238, [csi148]).
prereq(csi260, [csi108]).
prereq(csi270, [csi148]).
prereq(csi310, [[csi260, csi270], [sta107, sta205, sta255]]).
prereq(csi324, [[csi148], [csi238, mat246]]).
```

Implement the following predicates:

1. took has two arguments: a student's first name and a course name. It verifies if the student already took that course. Assume the two arguments are instantiated. Also assume that the first name uniquely identifies a student (this is not the case in reality, students numbers would be used as unique identifiers).
2. is_prereq has two course names as arguments and verifies if the second one is a prerequisite for the first one. Assume that the two arguments are instantiated.
3. can-take with two arguments, a student's first name and a course, and succeeds if the student has all the prerequisites necessary to take the course. If there are alternative prerequisites, the student is required to have at least one of them. A student cannot take a course if she/he already took it.
4. can-take-list has two arguments, a student's first name and a list of courses, and succeeds if the student can take all the courses in the list (fails otherwise).