Lab 3 – 08226 Artificial Intelligence

This lab tutorial will introduce you to using rules in your Prolog programs.

1.0 Rules using Comer (and)

Start SWI-Prolog-Editor from the Windows Menu system and create a new Prolog file called **Lab3.pl** and store it in **G:/08226/Lab 3/**.

Rules in Prolog are extensions of Facts. We add sub-goals that have to be satisfied for the rule to return true. A rule has a **head** and a **body**.

Consider the following code:

```
weather(bad) :-
        temperature(low),
        cloud(high).
```

The above rule has the head **`weather(bad)`** and the body **`temperature(low),cloud(high)`**. This rule is stating that for it to be bad weather, there **MUST** be low temperature **AND** high cloud.

Enter the following code into your program and then select **consult**:

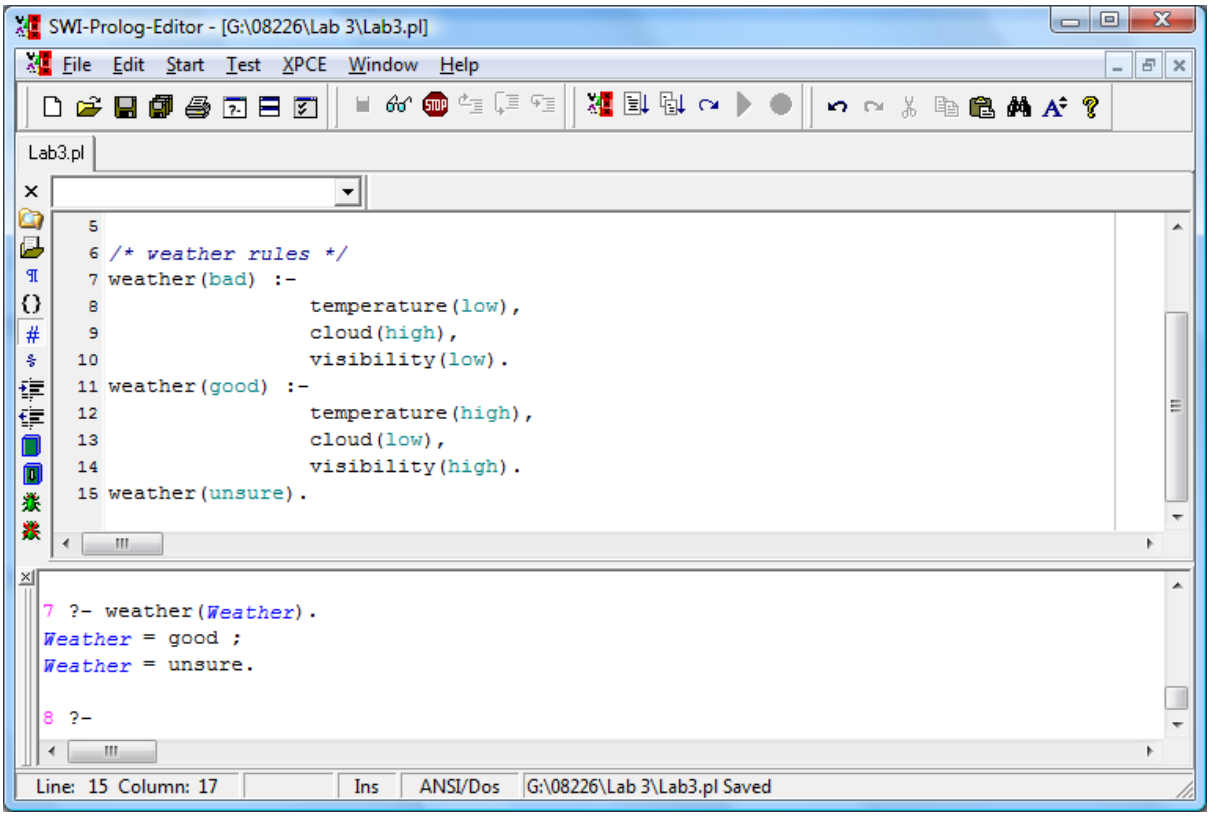
```
/* weather facts */
temperature(high).
cloud(low).
visibility(high).


/* weather rules */
weather(bad) :-
        temperature(low),
        cloud(high),
        visibility(low).
weather(good) :-
        temperature(high),
        cloud(low),
        visibility(high).
weather(unsure).
```

Ask Prolog what are the possible weather conditions.

```
?- weather(Weather).
```

You should have the following result:



Prolog has stated that the weather is either **good** or **unsure**.

Why is it possibly good weather? The good weather rule will return true if there is high temperature and low cloud and high visibility. If you look at the facts, we indeed have these three facts, therefore the good weather rule returns true.

Why is it possibly unsure weather? The unsure weather rule will return true because it has no sub-goals to satisfy.

1.1 Rules Test Exercise 1

Write down your answers to the following questions without using Prolog, and then check your answers at the end of this document and/or using Prolog.

Given the following questions, write down the appropriate Prolog queries and which value will be returned?

1) Is the weather good? ***true / false***
2) Is the weather bad? ***true / false***

Check your answers here: **Rules Test Exercise 1 Answers**

1.2 Rules Test Exercise 2

Write a Prolog program using facts in the form **grade/2** that represents the following MSc Degree grade classifications:

MSc Degree grade	Rule
Distinction	Mark >= 70
Pass	Mark >= 40 and Mark < 70
Fail	Mark < 40

Your program should answer the following queries:

```
?- grade(Grade,75).
```
Grade = distinction

```
?- grade(Grade,60).
```
Grade = pass

```
?- grade(Grade,39).
```
Grade = fail

Check your answers here: **Rules Test Exercise 2 Answers**

2.0 Rules using Semi-Colon (or)

We can use the **semi-colon** to produce an option within Prolog. If we wanted to produce a rule that returned true if someone is tired, we could write the following:

```
person(tired) :-
        yawning ;
        eyes_are_closed.
```

We read this rule as a person is tired if they are either yawning **or** their eyes are closed.

2.1 Rules Test Exercise 3

Write a Prolog program using facts in the form **car_fault/1** that represents the following rules:

A car fault is electrical if the car will not start or the lights will not work
A car fault is electrical if the car will not start and the lights will not work, or the battery is flat

Your program should answer the following query:

```
?- car_fault(electrical).
```
true

Check your answers here: **Rules Test Exercise 3 Answers**

3.0 Rules Test Exercise 1 Answers

Given the following questions, write down the appropriate Prolog queries and which value will be returned?

1) Is the weather good? `?- weather(good).` ***true***
2) Is the weather bad? `?- weather(bad).` ***false***

Check the answers using Prolog.

Return to your exercise here: **Rules Test Exercise 1**

4.0 Rules Test Exercise 2 Answers

Write a Prolog program using facts in the form **grade/2** that represents the following MSc Degree grade classifications

MSc Degree grade	Rule
Distinction	Mark >= 70
Pass	Mark >= 40 and Mark < 70
Fail	Mark < 40

```
grade(distinction, Mark) :-
                    Mark >= 70.
grade(pass, Mark) :-
                    Mark >= 40, Mark < 70.
grade(fail, Mark) :-
                    Mark < 40.
```

Return to your exercise here: **Rules Test Exercise 2**

5.0 Rules Test Exercise 3 Answers

Write a Prolog program using facts in the form **car_fault/1** that represents the following rules:

A car fault is electrical if the car will not start or the lights will not work
A car fault is electrical if the car will not start and the lights will not work, or the battery is flat

```
car_fault(electrical) :-
                        car_will_not_start ; lights_will_not_work.

car_fault(electrical) :-
                        (car_will_not_start , lights_will_not_work) ;
                        battery_is_flat.
```

Return to your exercise here: **Rules Test Exercise 3**