Prolog
Some Additional Features
(Read/Write, etc.)

Fariba Sadri

- listing. Lists all user definitions currently loaded.
- listing(p). Lists the definition of a given predicate p.

In Linux to be able to use listing you must use consult(<program>), rather than compile(<program>).

Input / Output

- write(T) writes a term T
- read(X) unifies with X the next term read (the term must be followed by “.” and carriage return if written from keyboad)
- nl writes a new line character
- tab(N) produces N tab spaces

tab may not work in some Prologs. But we can program it easily.

```
tab(0).
tab(N) :- N>0, write(' '), M is N-1, tab(M).
```

Example

A Prolog program that writes the content of a list, one element per line:


```
writelist([]).
writelist([X|L]):- write(X), nl, writelist(L).
```



```
| ?- writelist(['Ready', 'Steady', 'Go']).
Ready
Steady
Go
yes
```

Another Example

/*

A program for practising the squares of number:

- It asks the user if they want to play.
- If they do then it generates an integer at random, and
- asks them to input the square of the integer.
- If they get it right it says correct, otherwise it says wrong and outputs the correct square.

*/

```
% Load the random number generator library.
% This is called a directive.
:- use_module(library(random)).
check_squares:-
        nl, write('Do you want to play ?'),
        nl, read(X),
        (X=no -> (write(goodbye), nl, nl);
                random(1,20,Y),
                % Randomly generates an integer between 1 and 20.
                nl, askabout(Y)).
```

random(+*L*, +*U*, -*R*)

unifies *R* with a random integer in *[L,U)* when *L* and *U* are integers (note that *U* will *never* be generated).

```
askabout(X):-
        writeMessage(['what is the square of', X, '?']),
        read(Y),
        Z is X*X,
        (Y is Z -> write(correct), nl ;
        write(wrong),tab(4),
        writeMessage(['the square of', X, is, Z])),
        check_squares.
```

Example cntd.

```
askabout(X):-
        writeMessage(['what is the square of',
        X, '?']),
        read(Y),
        (Y is X*X -> write(correct),nl ;
        write(wrong),nl),
        check_squares.
```

Example cntd.

```
writeMessage([]).

writeMessage([X|L]):-
        write(X),
        write(' '),
        writeMessage(L),
        nl.
```

Comparison operators in Prolog

Comparison	Definition	Evaluates?
X = Y	succeeds if X and Y unify (match) in the Prolog sense	No
X \= Y	succeeds if X and Y do not unify; i.e. if not (X = Y)	No
T1 == T2	succeeds if terms T1 and T2 are identical; e.g. names of variables have to be the same	No
T1 \== T2	succeeds if terms T1 and T2 are not identical	No
E1 =:= E2	succeeds if values of expressions E1 and E2 are equal	Yes
E1 =\= E2	succeeds if values of expressions E1 and E2 are not equal	Yes
E1 < E2	succeeds if numeric value of expression E1 is < numeric value of E2	Yes
E1 =< E2	succeeds if numeric value of expression E1 is ≤ numeric value of E2	Yes
E1 > E2	succeeds if numeric value of expression E1 is > numeric value of E2	Yes
E1 >= E2	succeeds if numeric value of expression E1 is ≥ numeric value of E2	Yes
T1 @< T2	succeeds if T1 is alphabetically < T2	No
T1 @=< T2	succeeds if T1 is alphabetically ≤ T2	No
T1 @> T2	succeeds if T1 is alphabetically > T2	No
T1 @>= T2	succeeds if T1 is alphabetically ≥ T2	No

Inserting clauses at the terminal

```
| ?- [user].
% consulting user …
| p(X):-q(X).
| p(X):-r(X).
| q(1).
| r(2).
| end_of_file.
% consulted user in module user, 0 msec 560 bytes
yes
| ?-
```

Instead of

| end_of_file.

You can type

| <Ctrl> z

How do you save the terminal entered program to a file?

If you wish to save the program you have entered at the terminal as a text file, you can use the following steps:

- After the "[user]." part you have to declare all your predicates *dynamic*.
- Then enter the program, ending with end_of_file or <Ctrl>z.
- Then use *tell*, *listing*, *told*.

Example

```
| ?- [user].
| dynamic(plus_one/2).
| plus_one( X, Y ) :- Y is X + 1.
| end_of_file. (or '<Ctrl>-Z').
| ?- tell(yourFilename).
| ?- listing.
| ?- told.
yes
```

In windows use the file -> Working directory menu to choose directory where the file will go.

In Linux give the full file name.

Reading and writing from files

(not part of this course)

Not needed in this course.

But Prolog has a variety of predicates for these tasks, such as:

- see(F) opens file F as the input file
- seen closes F as input file
- tell(F) opens file F as the output file
- told closes F as output file