LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE
OF COMPUTER APPLICATION, MUMBAI

Artificial Intelligence and Machine Learning

ASST. PROF. **AQUILA SHAIKH,**
DR. **RASHMITA PRADHAN,**
DIVAKAR JHA

Faculty, Master of Computer Application (M.C.A.)
Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application

16LEAVES

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

LATE BHAUSAHEB HIRAY S.S. TRUST'S INSTITUTE OF COMPUTER APPLICATION BANDRA, EAST MUMBAI-51

Assistant Professor Aquila Shaikh, Dr. Rashmita Pradhan, Divakar Jhar

Faculty, Master of Computer Application (M.C.A.)
Late Bhausaheb Hiray S.S. Trust's Institute of Computer Application

16LEAVES

First Edition, 2023

Paperback ISBN: 978-81-19221-07-3
eBook ISBN: 978-81-19221-19-6
WebPDF ISBN: 978-81-19221-13-4

Printed and bound in India by
16Leaves
2/579, Singaravelan Street
Chinna Neelankarai
Chennai – 600 041, India

info@16leaves.com
www.16Leaves.com
Call: 91-9940638999

Contents

Chapter 1 Logic Programming with Prolog and AI Problems

What is GNU:

1. GNU Prolog is a free Prolog compiler with constraint solving over finite domains developed by Daniel Diaz.

2. GNU Prolog accepts Prolog+constraint programs and produces native binaries (like gcc does from a C source). The obtained executable is then stand-alone. The size of this executable can be quite small since GNU Prolog can avoid to link the code of most unused built-in predicates. The performances of GNU Prolog are very encouraging (comparable to commercial systems).

3. Beside the native-code compilation, GNU Prolog offers a classical interactive interpreter (top-level) with a debugger.

4. The Prolog part conforms to the ISO standard for Prolog with many extensions very useful in practice (global variables, OS interface, sockets,...).

5. GNU Prolog also includes an efficient constraint solver over Finite Domains (FD). This opens contraint logic programming to the user combining the power of constraint programming to the declarativity of logic programming.

Prolog: (Programming in logic)

1. It is a logic programming language.

2. Logic means Facts and Rules. And based on Facts and Rules we go to the conclusion.

3. Prolog is a declarative programming language.

4. Prolog was designed by Alain Colmerauer and Robert Kowaski. It was first implemented in 1973.

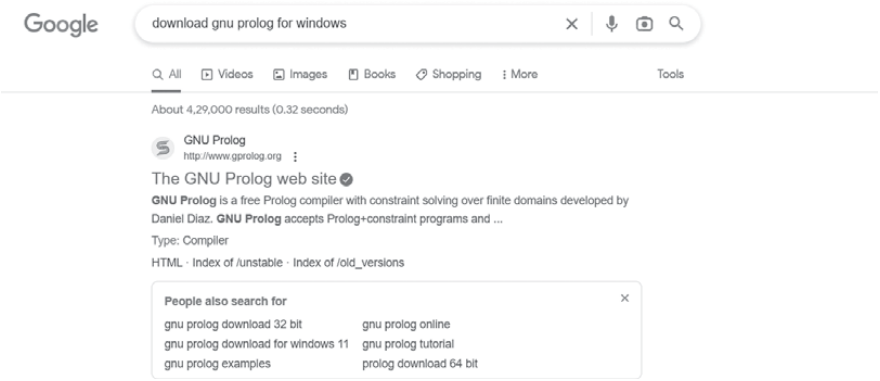
How does GNU Prolog work?

The GNU Prolog compiler is based on the Warren Abstract Machine (WAM). It first compiles a Prolog program to a WAM file which is then translated to a low-level machine independent language called mini-assembly specifically designed for GNU Prolog. The resulting file is then translated to the assembly language of the target machine (from which an object is obtained). This allows GNU Prolog to produce a native stand alone executable from a Prolog source (similarly to what does a C compiler from a C program). The main advantage of this compilation scheme is to produce native code and to be fast. Another interesting feature is that executables are small. Indeed, the code of most unused built-in predicates can be excluded from the executables at link-time.

GNU Prolog also includes an efficient constraint solver over Finite Domains (FD). The key feature of the GNU Prolog solver is the use of a single (low-level) primitive to define all (high-level) FD constraints. There are many advantages of this approach: constraints can be compiled, the user can define his own constraints (in terms of the primitive), the solver is open and extensible (as opposed to black-box solvers like CHIP),...Moreover, the GNU Prolog solver is rather efficient, often more than commercial solvers.

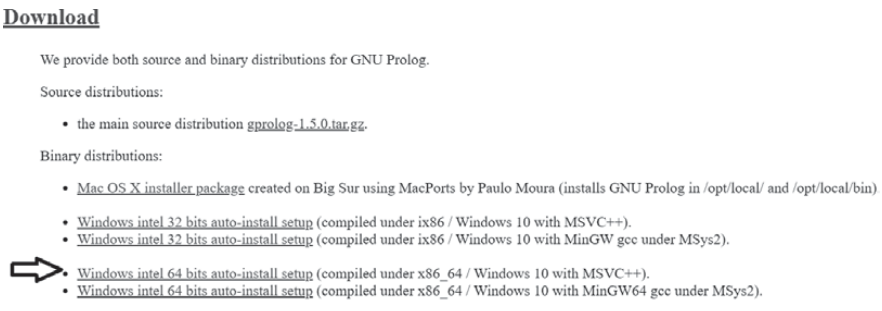
Installation of GNU Prolog:

Step 1: Search “download gnu prolog for windows” in google.

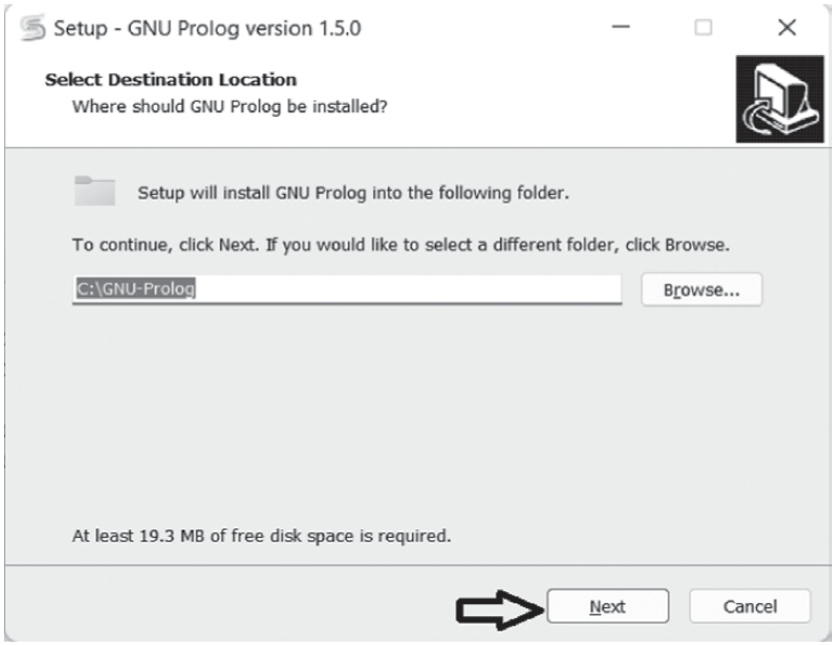


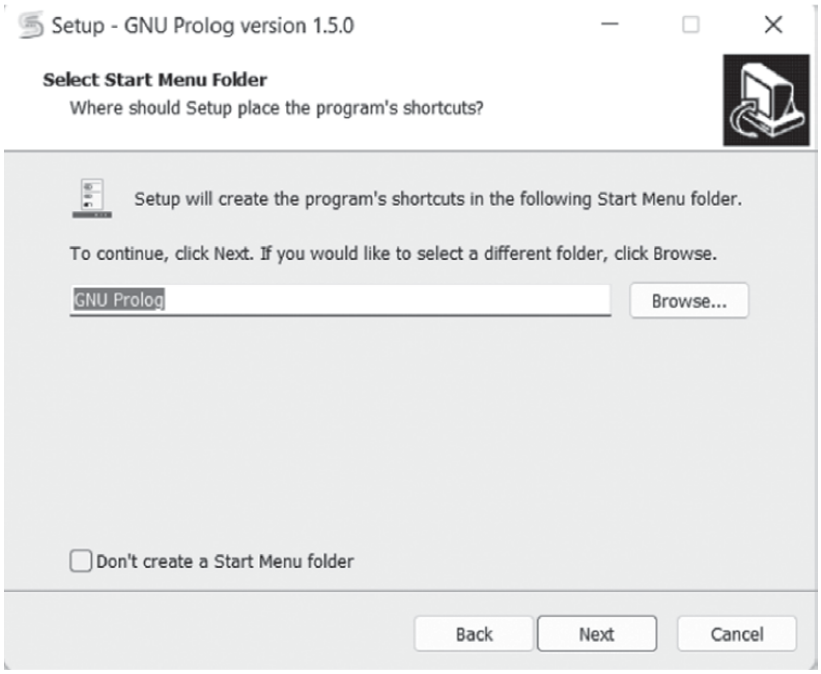
Step 2: Go to “GNU prolog website”.

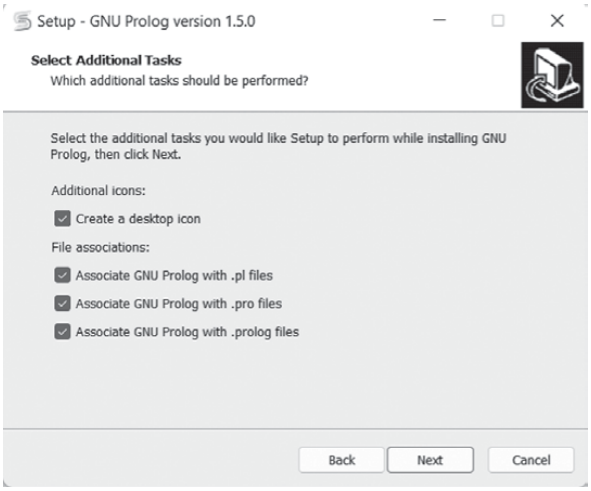
Step 3: Click on the following link:

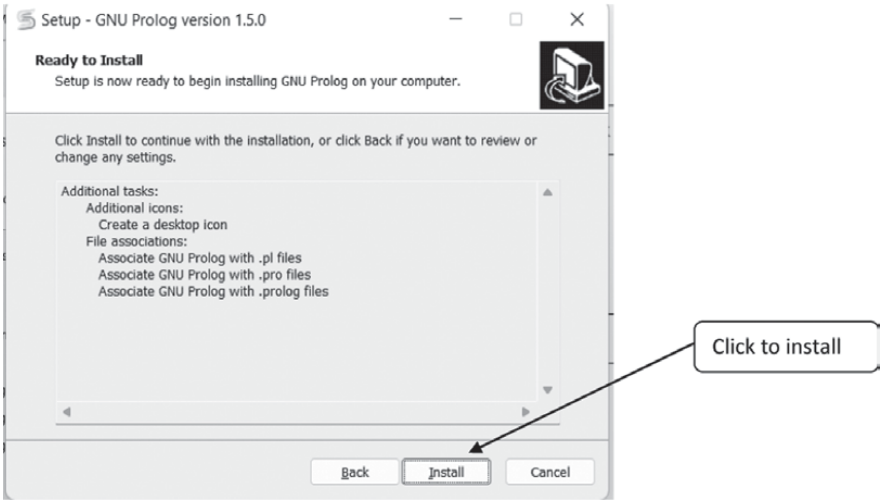


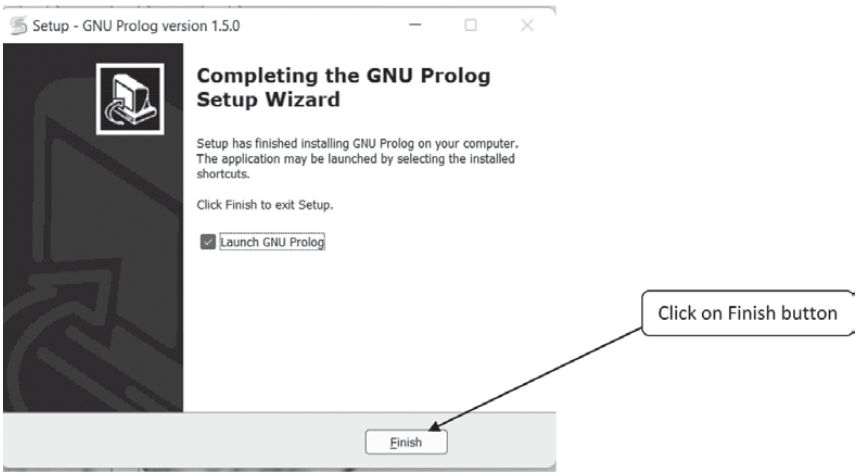
Step 4: Double-click on the "exe" file and install gnu as follows:

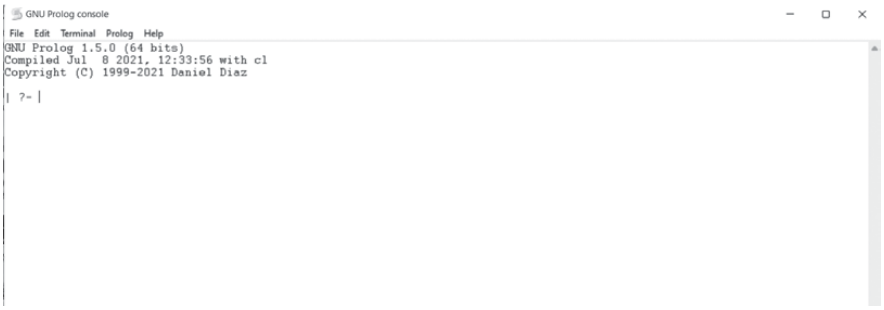





Setup - GNU Prolog version 1.5.0
Select Additional Tasks
Which additional tasks should be performed?
Select the additional tasks you would like Setup to perform while installing GNU Prolog, then click Next.
Additional icons:
Create a desktop icon
File associations:
Associate GNU Prolog with .pl files
Associate GNU Prolog with .pro files
Associate GNU Prolog with .prolog files
Back
Next
Cancel


Setup - GNU Prolog version 1.5.0
Ready to Install
Setup is now ready to begin installing GNU Prolog on your computer.
Click Install to continue with the installation, or click Back if you want to review or change any settings.
Additional tasks:
Additional icons:
Create a desktop icon
File associations:
Associate GNU Prolog with .pl files
Associate GNU Prolog with .pro files
Associate GNU Prolog with .prolog files
Back
Install
Cancel
Click to install


Setup - GNU Prolog version 1.5.0
Completing the GNU Prolog Setup Wizard
Setup has finished installing GNU Prolog on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch GNU Prolog
Finish
Click on Finish button



1.1: Demonstration of basic commands in prolog (Facts, Rule, and queries)

Facts: Facts can be defined as explicit relationships between objects and properties of objects. Facts are unconditionally true in nature.

Example:

1. Tom is a cat. → **cat(tom)** → This factual expression is called as clause.

2. Harry is a student.

Note: Anything that is within the parameter that we pass is called as argument and that argument is an object.

Relation: Relation defines the way in which a collection of objects or variables belong together.

Example: student(harry) → here student is a relation with harry.

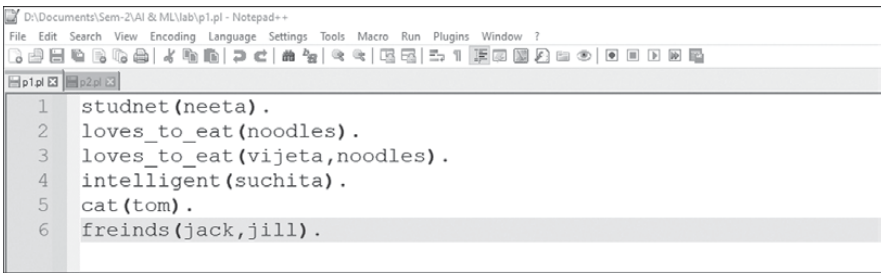
We can define more than one objects as: **relation (object1, object2, object3,.....)**

Rules: Rules can be defined as explicit relationship between objects. Rules are conditionally true.

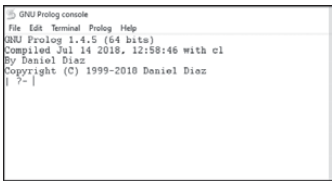
Example: Seema is happy if she sings.

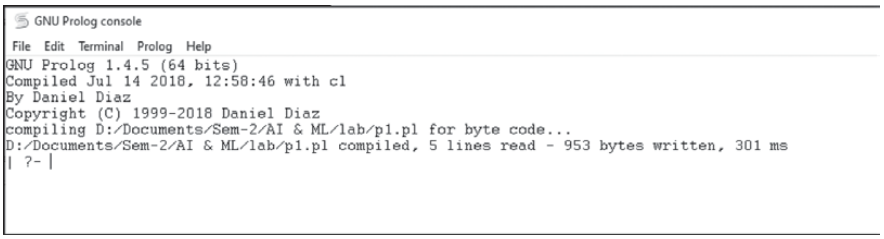
We can write above rules as: happpy(seema):-sings(seema)

Create one file in notepad with.pl extension and **file type =All type.**



Go to GNU → File → Change dir → Select the folder where notepad file is saved.



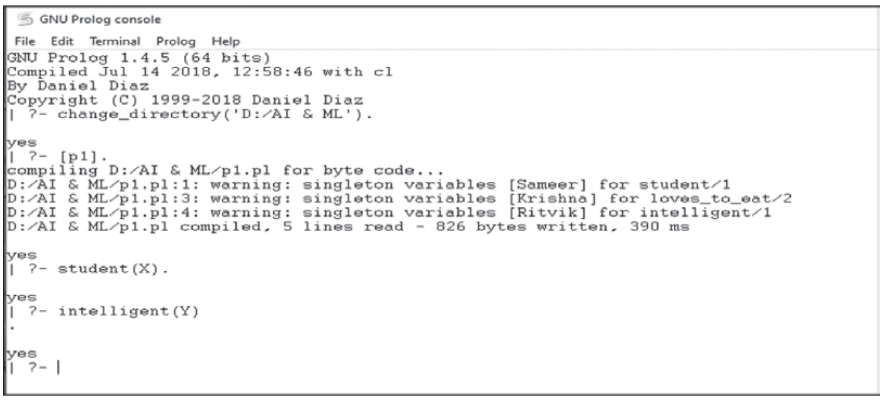


Code:-

→[p1].

→ Student(X).

→Intelligent(Y).



1.2: Demonstration of AND & OR Function in Prolog:

AND function(;)

OR function(,)

Create new notepad file.

→likes(pooja,geeta).

likes(geeta,pooja).

likes(neha,aliya).

freindship(X,Y):- likes(X,Y);likes(Y,X).

```
| ?- [p2].
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 884 bytes written, 17 ms

yes
| ?- freindship(X,Y).

X = pooja
Y = geeta ? ;

X = geeta
Y = pooja ? ;

X = neha
Y = aliya ? ;

X = geeta
Y = pooja ? ;

X = pooja
Y = geeta ? ;

X = aliya
Y = neha

(46 ms) yes
| ?-
```

Practice 1.1:

→ likes(pooja,geeta).

likes(geeta,pooja).

likes(neha,aliya).

freindship(X,Y):- likes(X,Y),likes(Y,X).

```
| ?- [p2].
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 763 bytes written, 16 ms

yes
| ?- freindship(X,Y).

X = pooja
Y = geeta ? ;

X = geeta
Y = pooja ? ;

no
| ?-
```

Practice 1.2:

→ next_to(mumbai,pune).

next_to(pune,satara).

next_to(mumbai,nashik).

travel(A,C):- next_to(A,B),next_to(B,C).

```
| ?- [p2].
compiling D:/AI & ML/p2.pl for byte code...
D:/AI & ML/p2.pl compiled, 5 lines read - 738 bytes written, 16 ms

yes
| ?- freindship(X,Y).

X = pooja
Y = geeta ? ;

X = geeta
Y = pooja ? ;

(31 ms) no
```

1.3: Relationship in prolog: - Specify relationship between object and properties of objects.

Relationship can also be a rule.

Relationship in Prolog

"Harry owns the book." ⟶ owner relationship

Harry and the book.

Relationship can also be a rule:

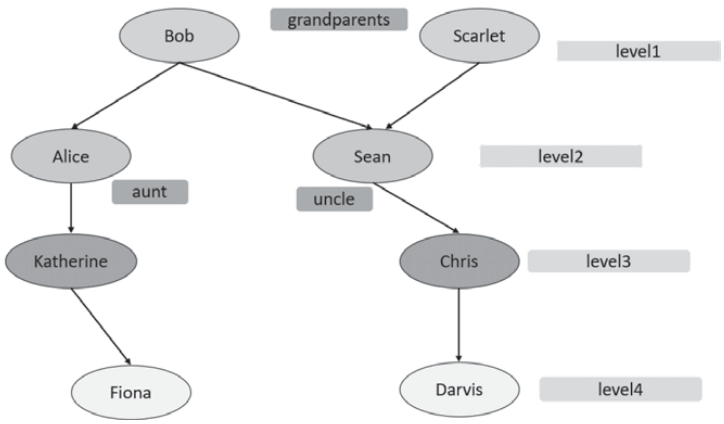
Two people are sisters if they both are female and they have same parents.

parent(Smith,Reema).
parent(Smith,Seema).
female(Reema).
female(Seema).

```
sisters(a,b) :- parent(z,a) , parent(z,b) , female(a) , female(b)
```

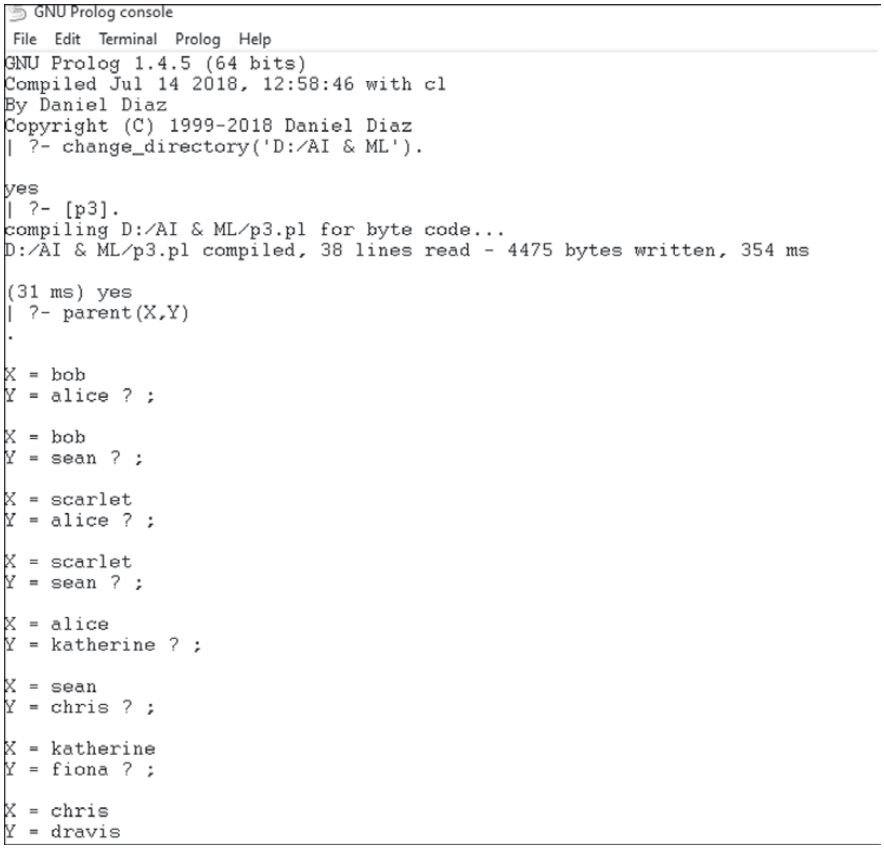
```
sisters(a,b) :- parent(z,a) , daughter(b,z) , female(a) , female(b)
```

Example:



→ **p3.pl**

```
female(scarlet).
female(alice).
female(katherine).
female(fiona).
male(bob).
male(sean).
male(chris).
male(dravis).
parent(bob, alice).
parent(bob, sean).
parent(scarlet,alice).
parent(scarlet, sean).
parent(alice, katherine).
parent(sean, chris).
parent(katherine,fiona).
parent(chris,dravis).
granparent(X,Y):- parent(X,Z),parent(Z,Y).
sister(X,Y):- parent(Z,X), parent(Z,Y), female(X), X\=Y.
brother(X,Y):- parent(Z,X), parent(Z,Y), male(X), female(Y).
uncle(X,Y):- parent(Z,Y), brother(X,Z).
aunt(X,Y):- parent(Z,Y),sister(X,Z).
daughter(X,Y):- parent(Y,X), female(X).
son(X,Y):- parent(Y,X), male(X).
mother(X,Y):- parent(X,Y), female(X).
father(X,Y):- parent(X,Y), male(X).
```


GNU Prolog console
File Edit Terminal Prolog Help
GNU Prolog 1.4.5 (64 bits)
Compiled Jul 14 2018, 12:58:46 with cl
By Daniel Diaz
Copyright (C) 1999-2018 Daniel Diaz
| ?- change_directory('D:/AI & ML').
yes
| ?- [p3].
compiling D:/AI & ML/p3.pl for byte code...
D:/AI & ML/p3.pl compiled, 38 lines read - 4475 bytes written, 354 ms
(31 ms) yes
| ?- parent(X,Y)
.
X = bob
Y = alice ? ;
X = bob
Y = sean ? ;
X = scarlet
Y = alice ? ;
X = scarlet
Y = sean ? ;
X = alice
Y = katherine ? ;
X = sean
Y = chris ? ;
X = katherine
Y = fiona ? ;
X = chris
Y = dravis

```
| ?- brother(X,Y).

X = sean
Y = alice ? ;

X = sean
Y = alice ? ;

no
| ?- sister(X,Y).

X = alice
Y = sean ? ;

X = alice
Y = sean ? ;

(16 ms) no
| ?- grandparent(X,Y).

X = bob
Y = katherine ? ;

X = bob
Y = chris ? ;

X = scarlet
Y = katherine ? ;

X = scarlet
Y = chris ? ;

X = alice
Y = fiona ? ;

X = sean
Y = dravis ? ;
```

```
| ?- uncle(X,Y).

X = sean
Y = katherine ? ;

X = sean
Y = katherine ? ;

(15 ms) no
| ?- aunt(X,Y).

X = alice
Y = chris ? ;

X = alice
Y = chris ? ;

(31 ms) no
| ?- mother(X,Y).

X = scarlet
Y = alice ? ;

X = scarlet
Y = sean ? ;

X = alice
Y = katherine ? ;

X = katherine
Y = fiona ? ;

(47 ms) no
```