

## EXPERIMENT NO 8

**Aim: Programming in PROLOG**

**Objective: Solving the tower of Hanoi problem and N-queen using PROLOG**

**Name:- Winster Pereira**

**Class :- TE COMPS - A**

**Roll No :- 9569**

### 1. Animals.pl

```
dog(rottweiler).
cat(munchkin).
animal(A) :- cat(A).
```

### 2. Appetizers.pl

```
likes(john, pizza).
likes(sarah, sushi).
likes(mike, pizza).
likes(mike, sushi).
likes(emma, sushi).
likes(emma, chocolate).
likes(emma, ice_cream).
likes(peter, ice_cream).
likes(peter, chocolate).
```

```
likes_similar(X, Y) :-
    likes(X, Z),
    likes(Y, Z),
    X \= Y.
```

### 3. Cars.pl

```
car(chevrolet, camaro, 2019, red).
car(chevrolet, corvette, 2020, blue).
car(ford, mustang, 2018, yellow).
car(ford, focus, 2017, silver).
car(toyota, corolla, 2019, black).
car(toyota, rav4, 2021, white).
```

% Rules

```
manufacturer(Make, Model, Year, Color) :-
    car(Make, Model, Year, Color).
```

```
car_color(Model, Color) :-
    car(_, Model, _, Color).
```

```
car_year(Model, Year) :-
    car(_, Model, Year, _).
```

#### 4. Countries.pl

```
% Facts about countries and their capitals
capital(paris, france).
capital(berlin, germany).
capital(london, uk).
capital(rome, italy).
capital(madrid, spain).

% Facts about languages spoken in countries
language(france, french).
language(germany, german).
language(uk, english).
language(italy, italian).
language(spain, spanish).

% Facts about currencies used in countries
currency(france, euro).
currency(germany, euro).
currency(uk, pound_sterling).
currency(italy, euro).
currency(spain, euro).

speaks_same_language(Country1, Country2) :-
    capital(Capital, Country1),
    capital(Capital, Country2),
    Country1 \= Country2,
    language(Country1, Language),
    language(Country2, Language).

shares_same_currency(Country1, Country2) :-
    currency(Country1, Currency),
    currency(Country2, Currency),
    Country1 \= Country2.
```

#### 5. Siblings.pl

```
parent(john, mary).
parent(john, david).
parent(kate, mary).
parent(kate, david).
male(john).
male(david).
female(kate).
female(mary).

mother(Mother, Child) :-
    parent(Mother, Child),
    female(Mother).

father(Father, Child) :-
    parent(Father, Child),
    male(Father).
```

```
sibling(Sibling1, Sibling2) :-
    parent(Parent, Sibling1),
    parent(Parent, Sibling2),
    Sibling1 \= Sibling2.

grandparent(Grandparent, Grandchild) :-
    parent(Grandparent, Parent),
    parent(Parent, Grandchild).
```

## 6. Food.pl

```
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).

meal(X) :- food(X).

?- food(pizza).
?- meal(X), lunch(X).
?- dinner(sandwich).
```

## 7. Employee.pl

```
employee(john, 2000).
employee(sarah, 2500).
employee(mike, 3000).
employee(emma, 2800).
employee(peter, 2200).

salary_above_average(Employee) :-
    employee(Employee, Salary),
    average_salary(Average),
    Salary > Average.

average_salary(Average) :-
    findall(Salary, employee(_, Salary), Salaries),
    sum_list(Salaries, Total),
    length(Salaries, Count),
    Average is Total / Count.
```

## 8. Student.pl

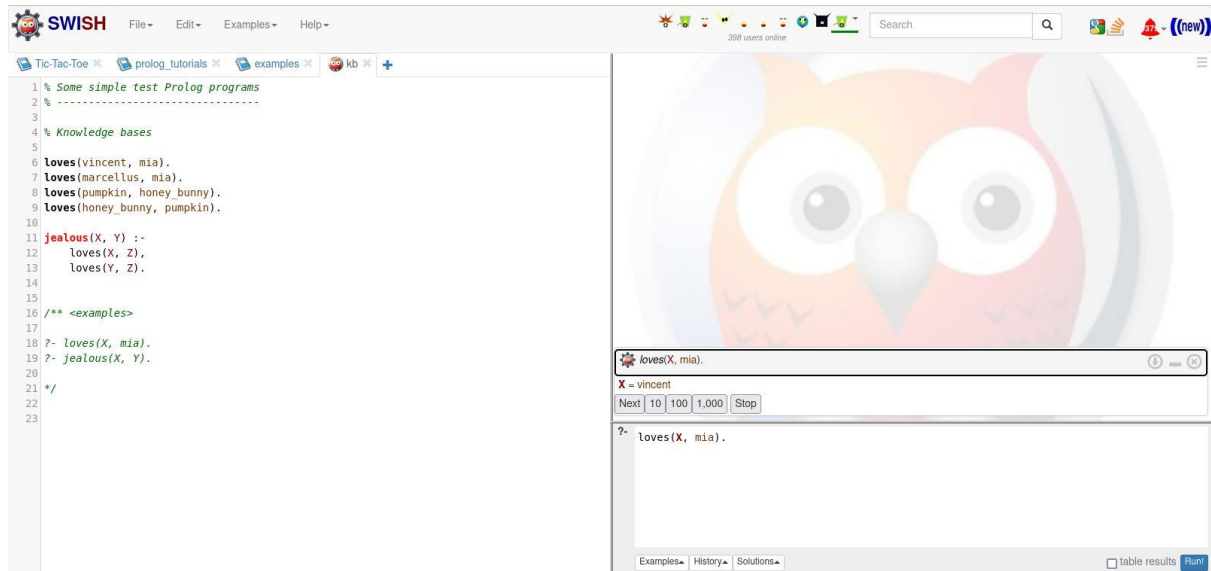
```
studies(charlie, csc135).
studies(olivia, csc135).
studies(jack, csc131).
studies(arthur, csc134).

teaches(kirke, csc135).
teaches(collins, csc131).
```

```
teaches(collins, csc171).
teaches(juniper, csc134).
```

```
professor(X, Y) :-
teaches(X, C), studies(Y, C).
```

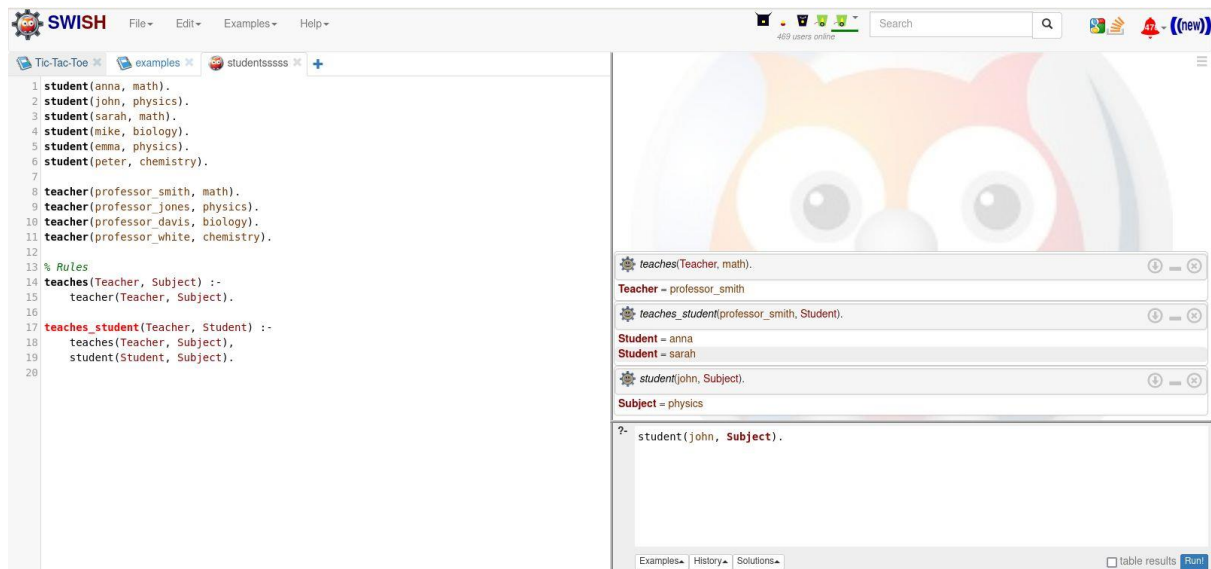
## Outputs:



The screenshot shows the SWISH Prolog IDE interface. The left pane contains a Prolog program with the following code:

```
1 % Some simple test Prolog programs
2 %
3
4 % Knowledge bases
5
6 loves(vincent, mia).
7 loves(marcellus, mia).
8 loves(pumpkin, honey_bunny).
9 loves(honey_bunny, pumpkin).
10
11 jealous(X, Y) :-
12   loves(X, Z),
13   loves(Y, Z).
14
15
16 /** <examples>
17
18 ?- loves(X, mia).
19 ?- jealous(X, Y).
20
21 */
22
23
```

The right pane displays the output of the query `?- loves(X, mia).`. The result is `X = vincent`. Below the result, there are buttons for `Next`, `10`, `100`, `1,000`, and `Stop`. The bottom of the right pane shows tabs for `Examples`, `History`, and `Solutions`, along with a checkbox for `table results` and a `Run!` button.



The screenshot shows the SWISH Prolog IDE interface. The left pane contains a Prolog program with the following code:

```
1 student(anna, math).
2 student(john, physics).
3 student(sarah, math).
4 student(mike, biology).
5 student(emma, physics).
6 student(peter, chemistry).
7
8 teacher(professor_smith, math).
9 teacher(professor_jones, physics).
10 teacher(professor_davis, biology).
11 teacher(professor_white, chemistry).
12
13 % Rules
14 teaches(Teacher, Subject) :-
15   teacher(Teacher, Subject).
16
17 teaches_student(Teacher, Student) :-
18   teaches(Teacher, Subject),
19   student(Student, Subject).
20
```

The right pane displays the output of the query `?- teaches(Teacher, math).`. The result is `Teacher = professor_smith`. Below the result, there are buttons for `Next`, `10`, `100`, `1,000`, and `Stop`. The bottom of the right pane shows tabs for `Examples`, `History`, and `Solutions`, along with a checkbox for `table results` and a `Run!` button.

File Edit Examples Help

Search
484 users online

((new))

Tic-Tac-Toe examples kb appetizerssss salariessss +

```

1 employee(john, 2000).
2 employee(sarah, 2500).
3 employee(mike, 3000).
4 employee(emma, 2800).
5 employee(peter, 2200).
6
7 salary_above_average(Employee) :-
8   employee(Employee, Salary),
9   average_salary(Average),
10  Salary > Average.
11
12
13 average_salary(Average) :-
14   findall(Salary, employee(_, Salary), Salaries),
15   sum_list(Salaries, Total),
16   length(Salaries, Count),
17   Average is Total / Count.

```

```

salary_above_average(Employee).
Employee = mike
Employee = emma
false

average_salary(Average).
Average = 2500

employee(john, Salary).
Salary = 2000

```

? employee(john, salary).

Examples History Solutions
☐ table results
Run!

File Edit Examples Help

Search
477 users online

((new))

Tic-Tac-Toe examples kb appetizerssss +

```

1 likes(john, pizza).
2 likes(sarah, sushi).
3 likes(mike, pizza).
4 likes(mike, sushi).
5 likes(emma, sushi).
6 likes(emma, chocolate).
7 likes(emma, ice_cream).
8 likes(peter, ice_cream).
9 likes(peter, chocolate).
10
11 likes_similar(X, Y) :-
12   likes(X, Z),
13   likes(Y, Z),
14   X \= Y.
15

```

```

likes(X, pizza).
X = john
X = mike

likes(sarah, sushi).
true

likes_similar(X, Y).
X = john,
Y = mike
X = sarah,
Y = mike
X = sarah,
Y = emma
Next 10 100 1,000 Stop

likes(X, chocolate), likes(X, ice_cream), \+ likes(X, ice_cream).
false

```

? likes(X, chocolate), likes(X, ice\_cream), \+ likes(X, ice\_cream).

Examples History Solutions
☐ table results
Run!

SWISH File Edit Examples Help 487 users online

Tic-Tac-Toe Hangout familiesssss New tab

```
1 parent(john, mary).
2 parent(john, david).
3 parent(kate, mary).
4 parent(kate, david).
5 male(john).
6 male(david).
7 female(kate).
8 female(mary).
9
10 mother(Mother, Child) :-
11     parent(Mother, Child),
12     female(Mother).
13
14 father(Father, Child) :-
15     parent(Father, Child),
16     male(Father).
17
18 sibling(Sibling1, Sibling2) :-
19     parent(Parent, Sibling1),
20     parent(Parent, Sibling2),
21     Sibling1 \= Sibling2.
22
23 grandparent(Grandparent, Grandchild) :-
24     parent(Grandparent, Parent),
25     parent(Parent, Grandchild).
```

parent(john, Child).  
Child = mary  
Child = david

mother(Mother, mary).  
Mother = kate

father(Father, david).  
Father = john  
false

sibling(Sibling1, Sibling2).  
Sibling1 = mary.  
Sibling2 = david  
Sibling1 = david,  
Sibling2 = mary  
Next 10 100 1,000 Stop

grandparent(Grandparent, david).  
false

?- grandparent(Grandparent, david).

Examples History Solutions table results Run

SWISH File Edit Examples Help 488 users online

Tic-Tac-Toe Hangout Program countriessss animalssss

```
1 dog(rottweiler).
2 cat(munchkin).
3 animal(A) :- cat(A).
```


dog(rottweiler).  
true

cat(munchkin).  
true

animal(munchkin).  
true

?- animal(munchkin).


Examples History Solutions table results Run

 **SWISH**

File Edit Examples Help

455 users online

Search



Tic-Tac-Toe

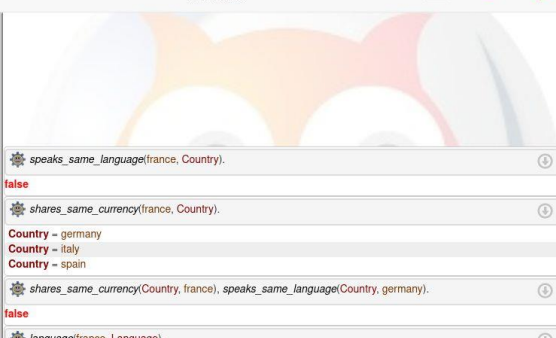
countries

Hangout

+

```
2 capital(paris, france).
3 capital(berlin, germany).
4 capital(london, uk).
5 capital(rome, italy).
6 capital(madrid, spain).
7
8 % Facts about languages spoken in countries
9 language(france, french).
10 language(germany, german).
11 language(uk, english).
12 language(italy, italian).
13 language(spain, spanish).
14
15 % Facts about currencies used in countries
16 currency(france, euro).
17 currency(germany, euro).
18 currency(uk, pound_sterling).
19 currency(italy, euro).
20 currency(spain, euro).
21
22 speaks_same_language(Country1, Country2) :-
23     capital(Capital, Country1),
24     capital(Capital, Country2),
25     Country1 \= Country2,
26     language(Country1, Language),
27     language(Country2, Language).
```

Type chat message here ...  
Send



speaks\_same\_language(france, Country).

false

shares\_same\_currency(france, Country).

Country = germany  
Country = italy  
Country = spain

shares\_same\_currency(Country, france), speaks\_same\_language(Country, germany).

false


language(france, Language).

Language = french

?- language(france, Language).

Examples History Solutions


☐ table results Run!

 **SWISH**

File Edit Examples Help

451 users online

Search

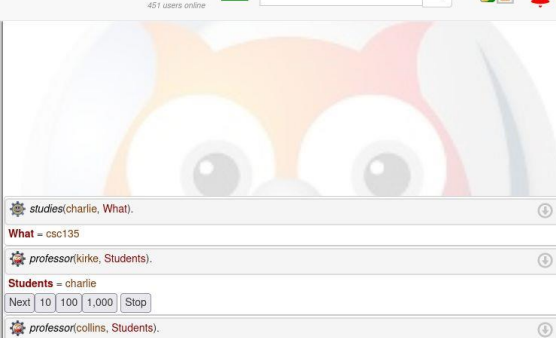


Tic-Tac-Toe

teacherrrr

+

```
1 studies(charlie, csc135).
2 studies(olivia, csc135).
3 studies(jack, csc131).
4 studies(arthur, csc134).
5
6
7 teaches(kirke, csc135).
8 teaches(collins, csc131).
9 teaches(collins, csc171).
10 teaches(juniper, csc134).
11
12 professor(X, Y) :-
13     teaches(X, C), studies(Y, C).
14
```



studies(charlie, What).

What = csc135

professor(kirke, Students).

Students = charlie  
Next 10 100 1,000 Stop

professor(collins, Students).

Students = jack  
Next 10 100 1,000 Stop

?- professor(collins, Students).

Examples History Solutions


☐ table results Run!

SWISH File Edit Examples Help

454 users online

Tic-Tac-Toe prolog\_tutorials examples kb foodddddd

```
1 food(burger).
2 food(sandwich).
3 food(pizza).
4 lunch(sandwich).
5 dinner(pizza).
6
7 meal(X) :- food(X).
8
9
```



food(pizza).  
true

lunch(pizza).  
false

?- lunch(pizza).

Examples History Solutions

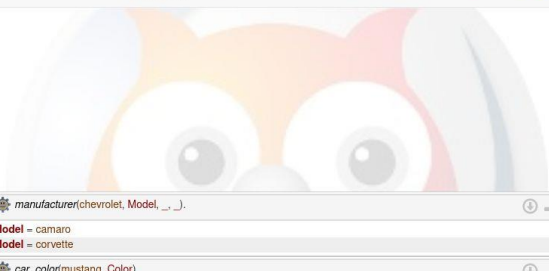
table results Run

SWISH File Edit Examples Help

450 users online

Tic-Tac-Toe examples carsssss

```
1 car(chevrolet, camaro, 2019, red).
2 car(chevrolet, corvette, 2020, blue).
3 car(ford, mustang, 2018, yellow).
4 car(ford, focus, 2017, silver).
5 car(toyota, corolla, 2019, black).
6 car(toyota, rav4, 2021, white).
7
8 % Rules
9 manufacturer(Make, Model, Year, Color) :-
10     car(Make, Model, Year, Color).
11
12 car_color(Model, Color) :-
13     car(_, Model, _, Color).
14
15 car_year(Model, Year) :-
16     car(_, Model, Year, _).
```



manufacturer(chevrolet, Model, \_).

Model = camaro  
Model = corvette

car\_color(mustang, Color).

Color = yellow

car\_year(Model, 2019).

Model = camaro  
Model = corolla

?- car\_year(Model, 2019).

Examples History Solutions

table results Run



## 1. Tower Of Hanoi

### Code:

```
% Move N disks from the From pole to the To pole using the Aux pole
% hanoi(+N, +From, +To, +Aux, -Moves)

:- initialization(hanoi).
hanoi(0, _, _, _, []).
hanoi(N, From, To, Aux, Moves) :-
    N1 is N - 1,
    hanoi(N1, From, Aux, To, Moves1),
    move(From, To, Moves2),
    hanoi(N1, Aux, To, From, Moves3),
    append(Moves1, Moves2, Temp),
    append(Temp, Moves3, Moves).

% Move a single disk from the From pole to the To pole
% move(+From, +To, -Moves)
move(From, To, [move(From, To)]).

/*
To use this program, you can call hanoi(N, From, To, Aux, Moves) where N
is the
number of disks, From is the starting pole, To is the destination pole,
Aux is the auxiliary pole,
and Moves is the list of moves required to solve the puzzle. For example:

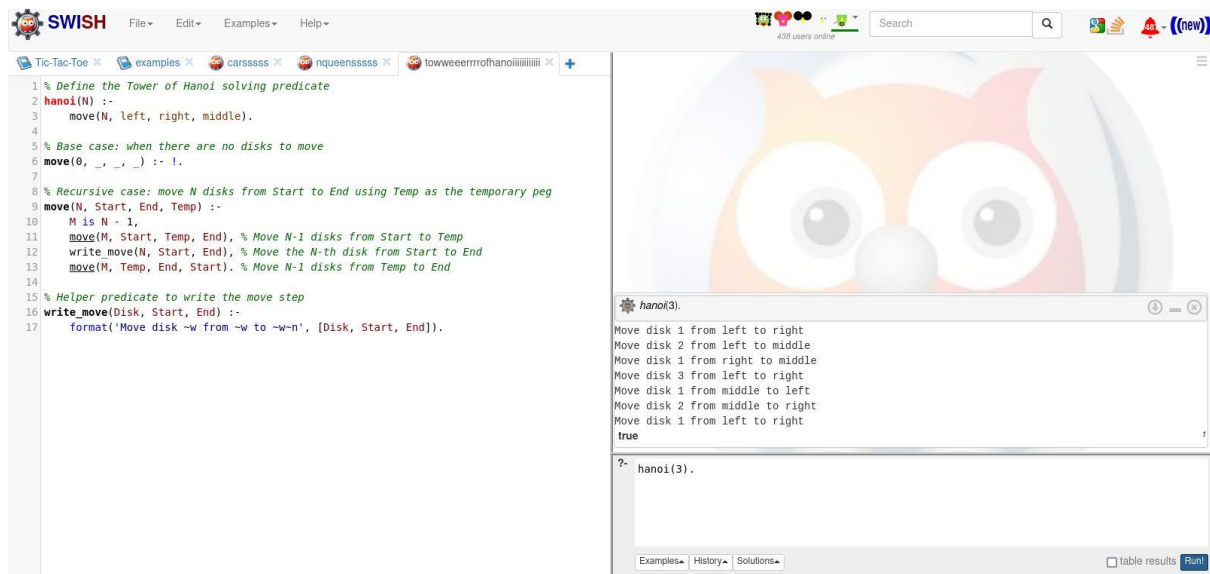
?- consult("D:/App Develop/AI/hanoi.pl").
true.

?- hanoi(3, left, right, middle, Moves).
Moves = [move(left, middle), move(left, right), move(middle, right),
move(left, middle), move(right, left), move(right, middle), move(left,
middle)]

This shows that to solve the puzzle with 3 disks, we need to move the top
2 disks to the
auxiliary pole, then move the bottom disk to the destination pole, and
finally move the 2
disks from the auxiliary pole to the destination pole. The move predicate
is used to
represent a single move, and the append predicate is used to concatenate
the lists of
moves generated by the recursive calls.

*/
```

## Output:



## 2. N-Queen Problem

### Code:

```
:- initialization(queens).
queens(N, Queens) :-
    length(Queens, N),
    board(Queens, Board, 0, N, _, _),
    queens(Board, 0, Queens).

board([], [], N, N, _, _).
board([_|Queens], [Col-Vars|Board], Col0, N, [_|VR], VC) :-
    Col is Col0+1,
    functor(Vars, f, N),
    constraints(N, Vars, VR, VC),
    board(Queens, Board, Col, N, VR, [_|VC]).

constraints(0, _, _, _) :- !.
constraints(N, Row, [R|Rs], [C|Cs]) :-
    arg(N, Row, R-C),
    M is N-1,
    constraints(M, Row, Rs, Cs).

queens([], _, []).
queens([C|Cs], Row0, [Col|Solution]) :-
    Row is Row0+1,
    select(Col-Vars, [C|Cs], Board),
    arg(Row, Vars, Row-Row),
    queens(Board, Row, Solution).

/* <To run>
1 ?- consult("D:/App Develop/AI/nqueens.pl").
true.
```

```
2 ?- queens(8, Queens).
Queens = [1, 5, 8, 6, 3, 7, 2, 4]
```

```
*/
```

Output:

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the Prolog code for the 8-queens problem, including initialization, board setup, constraints, and the query execution. The right pane shows the output of the query, listing 20 solutions for the 8-queens problem. The solutions are listed as Queens = [Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8].

```
1 :- initialization(queens).
2 queens(N, Queens) :-
3   length(queens, N),
4   board(queens, Board, 0, N, _, _),
5   queens(Board, 0, Queens).
6
7 board([], [], N, N, _, _).
8 board([_Queens], [Col-Vars|Board], Col0, N, [_VR], VC) :-
9   Col is Col0+1,
10  functor(Vars, f, N),
11  constraints(N, Vars, VR, VC),
12  board(queens, Board, Col, N, VR, [_VC]).
13
14 constraints(0, _, _, _) :- !.
15 constraints(N, Row, [R|Rs], [C|Cs]) :-
16   arg(N, Row, R-C),
17   M is N-1,
18   constraints(M, Row, Rs, Cs).
19
20 queens([], _, []).
21 queens([C|Cs], Row0, [Col|Solution]) :-
22   Row is Row0+1,
23   select(Col-Vars, [C|Cs], Board),
24   arg(Row, Vars, Row-Row),
25   queens(Board, Row, Solution).
26
27 /* <To run>
28 1 ?- consult("D:/App Develop/AI/nqueens.pl").
29 true.
30
31 2 ?- queens(8, Queens).
32 Queens = [1, 5, 8, 6, 3, 7, 2, 4]
33
34 */
```

Queens = [1, 5, 8, 6, 3, 7, 2, 4]  
Queens = [1, 6, 8, 3, 7, 4, 2, 5]  
Queens = [1, 7, 4, 6, 8, 2, 5, 3]  
Queens = [1, 7, 5, 8, 2, 4, 6, 3]  
Queens = [2, 4, 6, 8, 3, 1, 7, 5]  
Queens = [2, 4, 7, 1, 3, 8, 6, 4]  
Queens = [2, 5, 7, 4, 1, 8, 6, 3]  
Queens = [2, 6, 1, 7, 4, 8, 3, 5]  
Queens = [2, 6, 8, 3, 1, 4, 7, 5]  
Queens = [2, 7, 3, 6, 8, 5, 1, 4]  
Queens = [2, 7, 5, 8, 1, 4, 6, 3]  
Queens = [2, 8, 6, 1, 3, 5, 7, 4]  
Queens = [3, 1, 7, 5, 8, 2, 4, 6]  
Queens = [3, 5, 2, 8, 1, 7, 4, 6]  
Queens = [3, 5, 2, 8, 6, 4, 7, 1]  
Queens = [3, 5, 7, 1, 4, 2, 8, 6]  
Queens = [3, 5, 8, 4, 1, 7, 2, 6]  
Queens = [3, 6, 2, 5, 8, 1, 7, 4]  
Queens = [3, 6, 2, 7, 1, 4, 8, 5]  
Queens = [3, 6, 2, 7, 5, 1, 8, 4]  
Queens = [3, 6, 4, 1, 8, 5, 7, 2]  
Queens = [3, 6, 4, 2, 8, 5, 7, 1]  
Queens = [3, 6, 8, 1, 4, 7, 5, 2]  
Queens = [3, 6, 8, 1, 5, 7, 2, 4]  
Queens = [3, 6, 8, 2, 4, 1, 7, 5]  
Queens = [3, 7, 2, 8, 5, 1, 4, 6]  
Queens = [3, 7, 2, 8, 6, 4, 1, 5]  
Queens = [3, 8, 4, 7, 1, 6, 2, 5]  
Queens = [4, 1, 5, 8, 2, 7, 3, 6]

?- queens(8, Queens)

Examples History Solutions table results Run