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

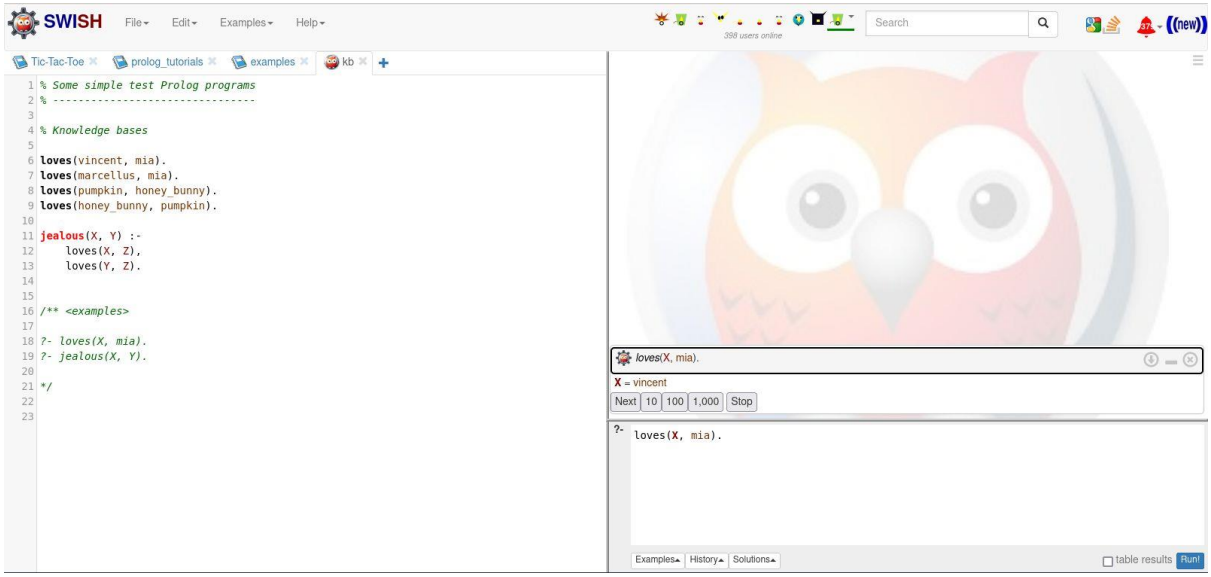
```
employee(john, 2000).
employee(sarah, 2500).
employee(mike, 3000).
employee(emma, 2800).
employee(peter, 2200).

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.
```

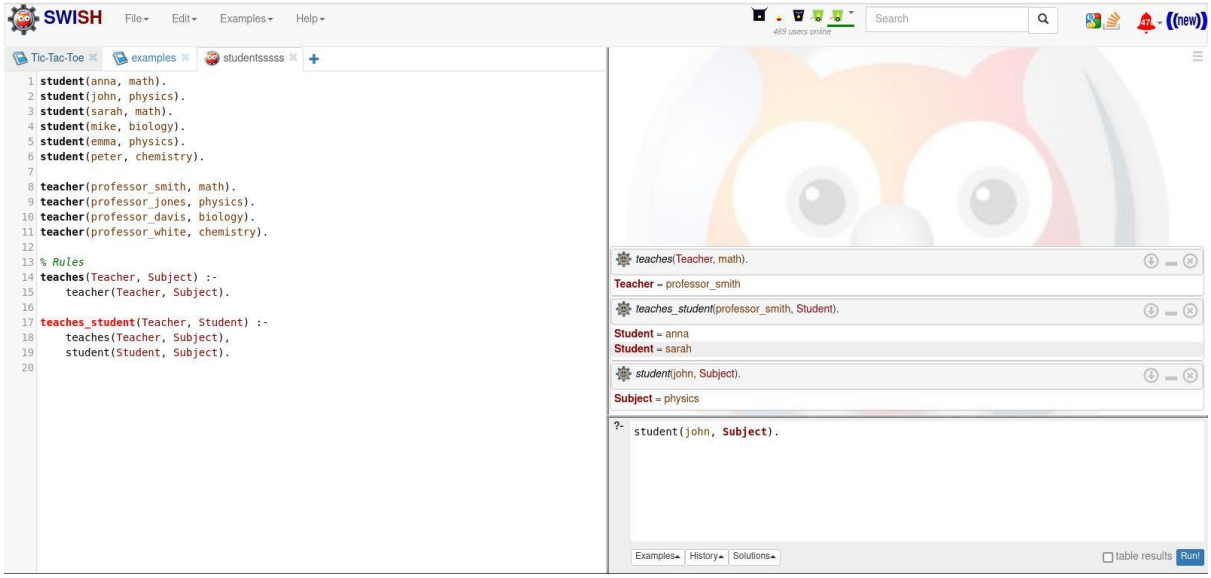
8. Student.pl

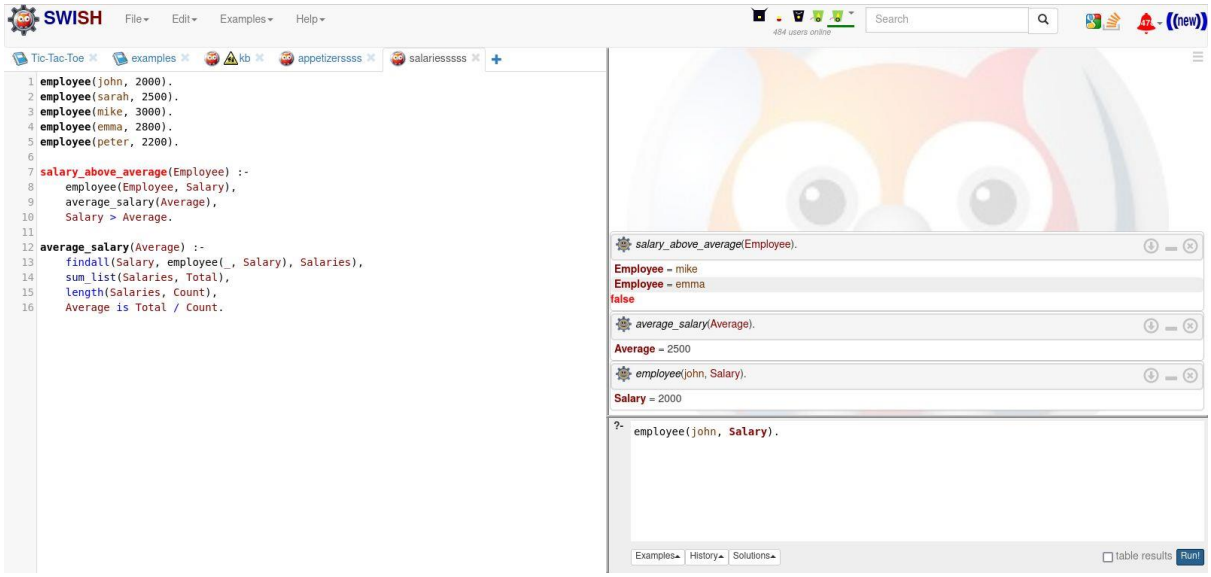
```
studies(charlie, csc135).
studies(olivia, csc135).
studies(jack, csc131).
studies(arthur, csc134).


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

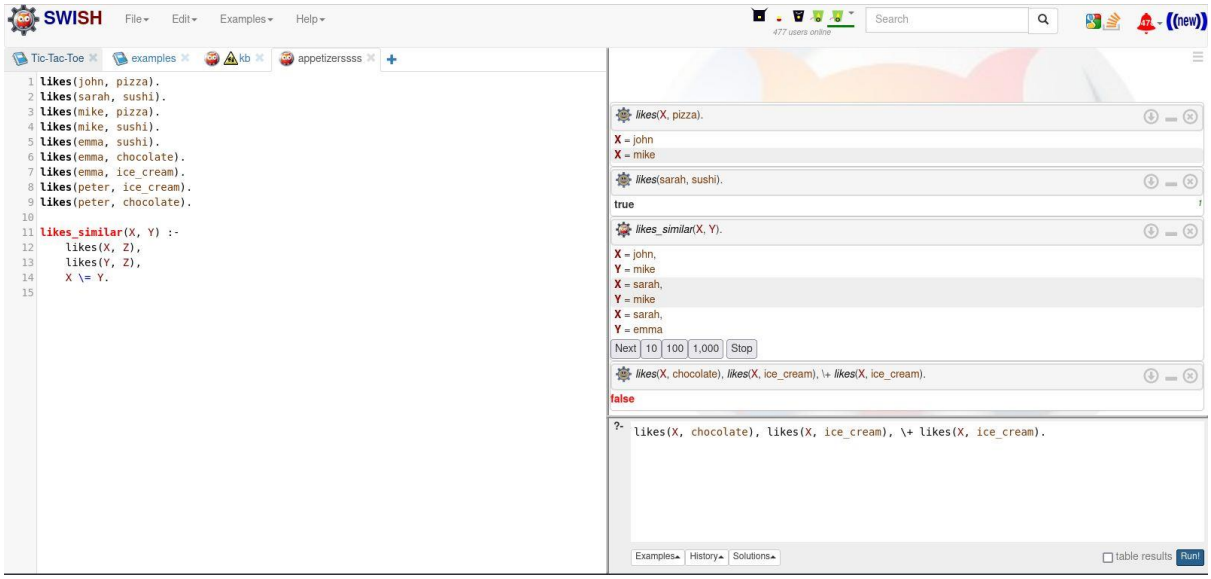
```
teaches(collins, csc171).
teaches(juniper, csc134).

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

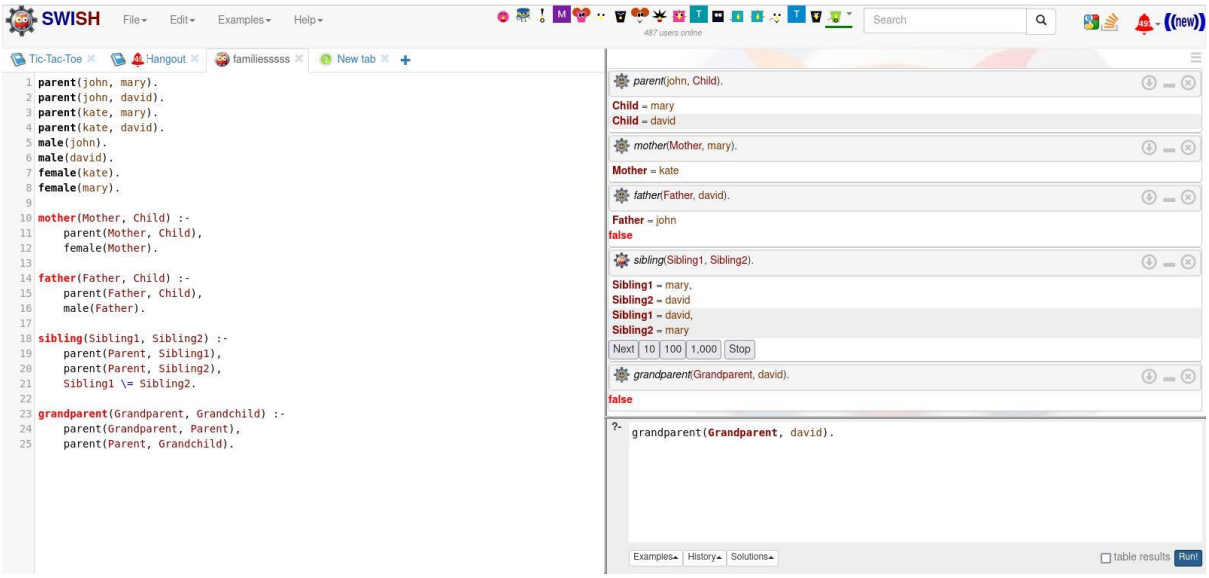
Outputs:

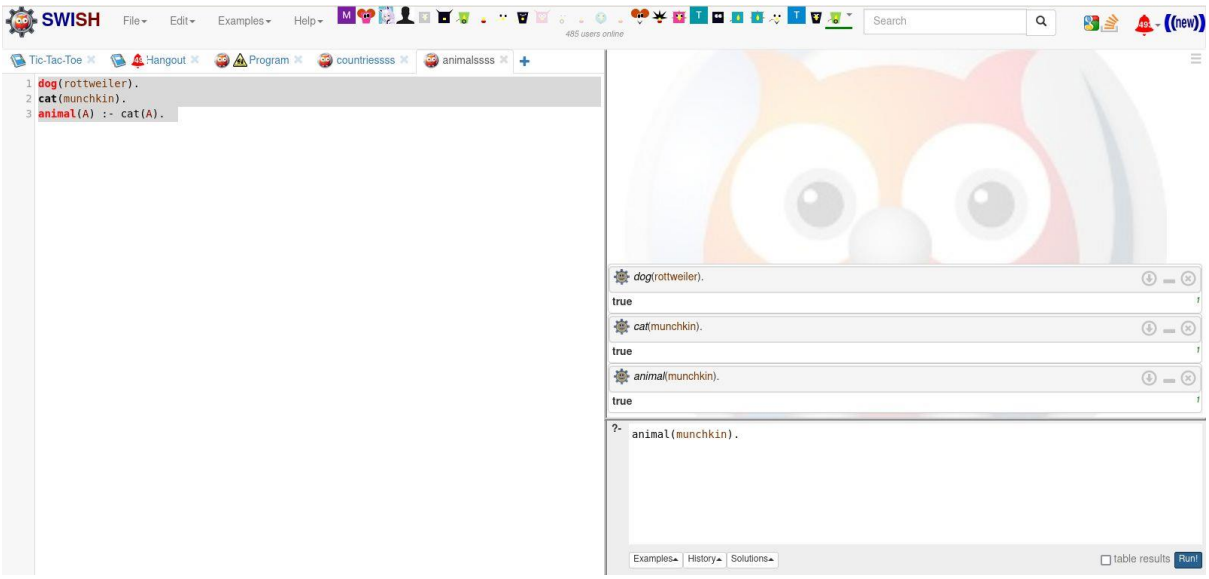


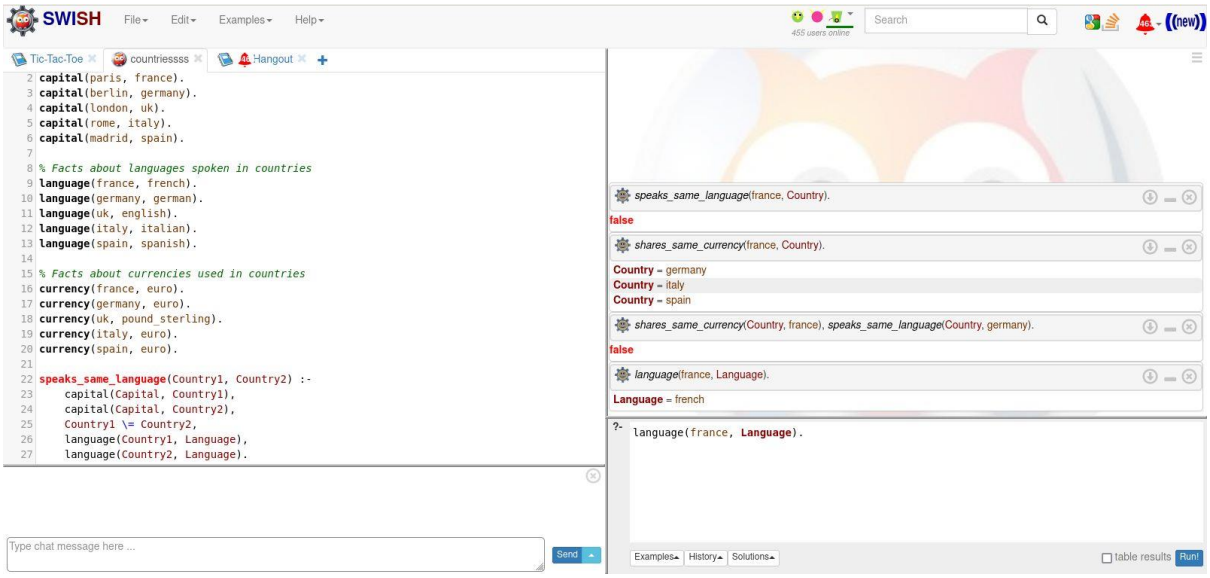


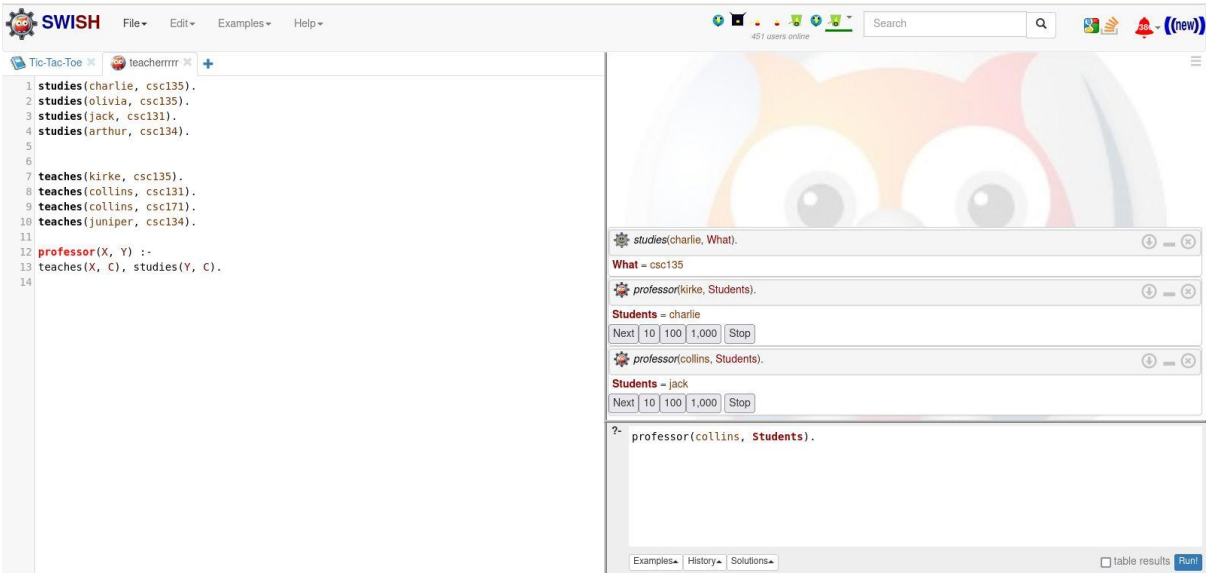

SWISH
File
Edit
Examples
Help
Search
Tic-Tac-Toe
examples
kb
appetizerssss
salariessssss
employee(john, 2000).
employee(sarah, 2500).
employee(mike, 3000).
employee(emma, 2800).
employee(peter, 2200).
salary_above_average(Employee) :-
employee(Employee, Salary),
average_salary(Average),
Salary > Average.
average_salary(Average) :-
findall(Salary, employee(_, Salary), Salaries),
sum_list(Salaries, Total),
length(Salaries, Count),
Average is Total / Count.
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!

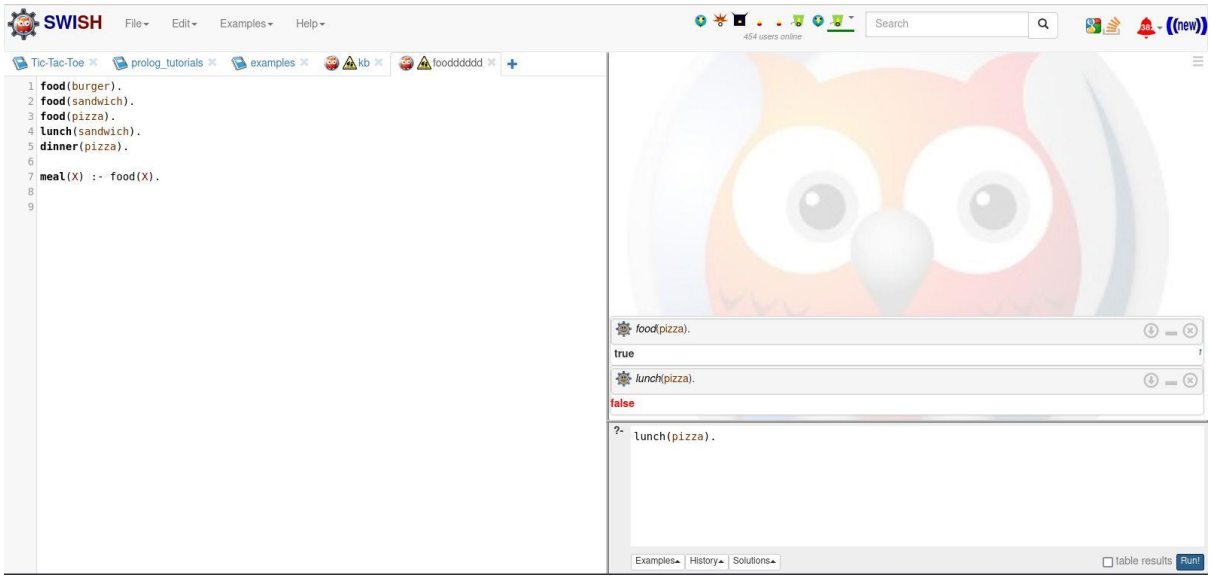

SWISH
File
Edit
Examples
Help
Search
Tic-Tac-Toe
examples
kb
appetizerssss
likes(john, pizza).
likes(sarah, sushi).
likes(mike, pizza).
likes(mike, sushi).
likes(emma, sushi).
likes(emma, chocolate).
likes(emma, ice_cream).
likes(peter, ice_cream).
likes(peter, chocolate).
likes_similar(X, Y) :-
likes(X, Z),
likes(Y, Z),
X \= Y.
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!

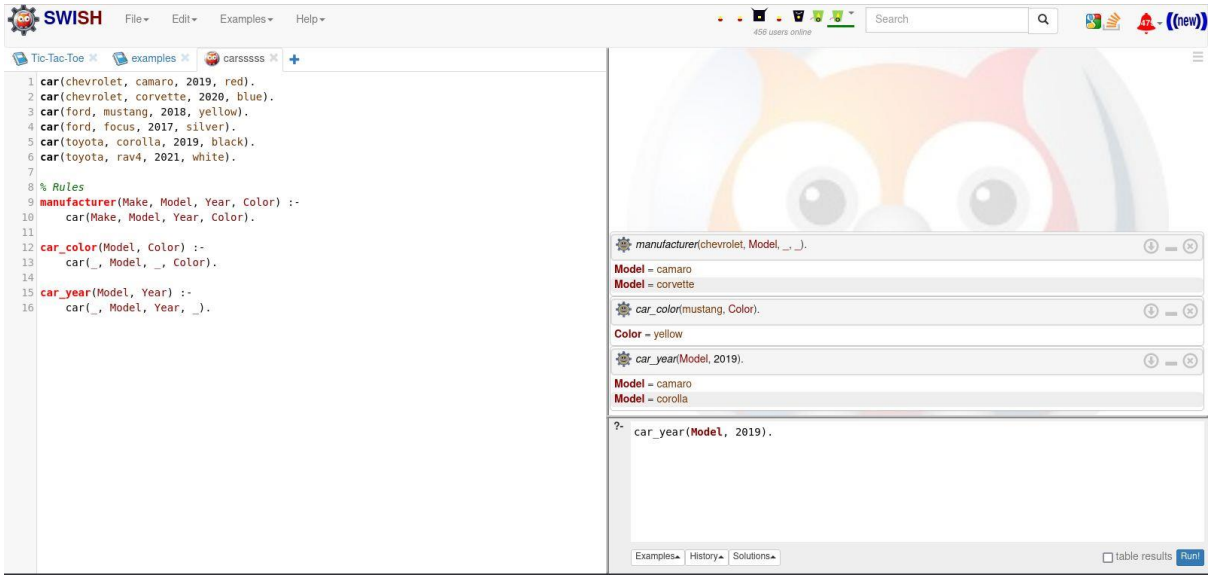

SWISH
File
Edit
Examples
Help
Search
Tic-Tac-Toe
Hangout
familiesssss
New tab
parent(john, mary).
parent(john, david).
parent(kate, mary).
parent(kate, david).
male(john).
male(david).
female(kate).
female(mary).
mother(Mother, Child) :-
parent(Mother, Child),
female(Mother).
father(Father, Child) :-
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).
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
Search
Tic-Tac-Toe
Hangout
Program
countriessss
animalssss
dog(rottweiler).
cat(munchkin).
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
countriessss
Hangout
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).
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).
Type chat message here ...
Send
Examples
History
Solutions
table results
Run!


SWISH
File
Edit
Examples
Help
451 users online
Search
Tic-Tac-Toe
teacherrrrr
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).
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
Search
Tic-Tac-Toe
prolog_tutorials
examples
kb
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
meal(X) :- food(X).
food(pizza).
true
lunch(pizza).
false
?-
lunch(pizza).
Examples
History
Solutions
table results
Run!

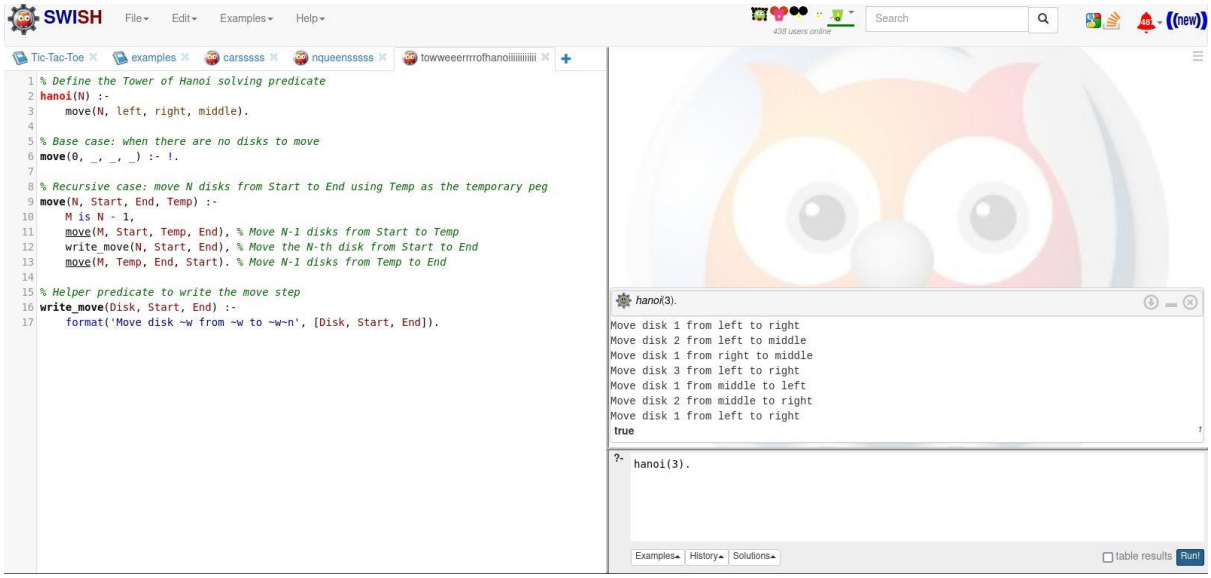

SWISH
File
Edit
Examples
Help
456 users online
Search
Tic-Tac-Toe
examples
carssssss
car(chevrolet, camaro, 2019, red).
car(chevrolet, corvette, 2020, blue).
car(ford, mustang, 2018, yellow).
car(ford, focus, 2017, silver).
car(toyota, corolla, 2019, black).
car(toyota, rav4, 2021, white).
% Rules
manufacturer(Make, Model, Year, Color) :-
car(Make, Model, Year, Color).
car_color(Model, Color) :-
car(_, Model, _, Color).
car_year(Model, Year) :-
car(_, Model, Year, _).
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: