

Prolog Programs

Name: Vedant Pathare

R.N: 9564, Comps-A

Batch B

1. Houses.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the Prolog code for 'Houses.pl', which includes a list of 15 clues for a logical puzzle and a query 'zebra_owner(Owner)'. The right pane shows the query result: 'zebra_owner(Owner).', 'Owner = japanese'. The interface includes a search bar, a list of open files, and a 'Run!' button.

2. Grammar.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the Prolog code for 'Grammar.pl', which includes a list of 28 rules for a simple English DCG grammar. The right pane shows the query 'phrase(s(Tree), [john, saw, a, man, with, a, telescope])' with the result 'Tree ='. Below the result, a parse tree diagram is shown, illustrating the structure of the sentence 'john saw a man with a telescope'. The diagram shows the root node 's' branching into 'pn' (john) and 'vp' (saw a man with a telescope). The 'vp' node branches into 'v' (saw) and 'np' (a man with a telescope). The 'np' node branches into 'det' (a) and 'n' (man). The 'pp' node branches into 'p' (with) and 'np' (a telescope). The 'np' node branches into 'det' (a) and 'n' (telescope).

3. ExpertSystem.pl

SWISH File Edit Examples Help 364 users online

examples houses_puzzle grammar clpfd_sudoku movies

expert_system queens lists kb

```
1 % A meta-interpreter implementing
2 % a tiny expert-system
3 % -----
4
5
6 prove(true) :- !.
7 prove((B, Bs)) :- !,
8     prove(B),
9     prove(Bs).
10 prove(H) :-
11     clause(H, B),
12     prove(B).
13 prove(H) :-
14     askable(H),
15     writeln(H),
16     read(Answer),
17     Answer == yes.
18
19
20 good_pet(X) :- bird(X), small(X).
21 good_pet(X) :- cuddly(X), yellow(X).
22
23 bird(X) :- has_feathers(X), tweets(X).
24
25 yellow(tweety).
26
27 askable(tweets(_)).
28 askable(small(_)).
```

prove(good_pet(tweety)).

has_feathers(tweety) yes

tweets(tweety) yes

small(tweety) yes

true

Next 10 100 1,000 Stop

?- prove(good_pet(tweety)).

Examples History Solutions table results Run!

4. Sudoku.pl

SWISH File Edit Examples Help 357 users online

examples houses_puzzle grammar clpfd_sudoku movies

expert_system queens lists

```
1 % render solutions nicely.
2 :- use_rendering(sudoku).
3
4 :- use_module(library(clpfd)).
5
6 % Example by Markus Triska, taken from the SWI-Prolog manual.
7
8 sudoku(Rows) :-
9     length(Rows, 9), maplist(same_length(Rows), Rows),
10     append(Rows, Vs), Vs ins 1..9,
11     maplist(all_distinct, Rows),
12     transpose(Rows, Columns),
13     maplist(all_distinct, Columns),
14     Rows = [A,B,C,D,E,F,G,H,I],
15     blocks(A, B, C), blocks(D, E, F), blocks(G, H, I).
16
17 blocks([], [], []).
18 blocks([A,B,C|Bs1], [D,E,F|Bs2], [G,H,I|Bs3]) :-
19     all_distinct([A,B,C,D,E,F,G,H,I]),
20     blocks(Bs1, Bs2, Bs3).
21
22 problem(1, [[_,_,_,_,_,_,_,_,_],
23             [_,_,_3,_,_8,5],
24             [_,_1,_2,_,_,_,_],
25             [_,_5,7,_,_,_],
26             [_,_4,_,_1,_,_],
27             [_9,_,_,_,_,_],
28             [_,_,_,_,_,_],
29             [_,_,_,_,_,_],
30             [_,_,_,_,_,_]]).
```

problem(1, Rows), sudoku(Rows).

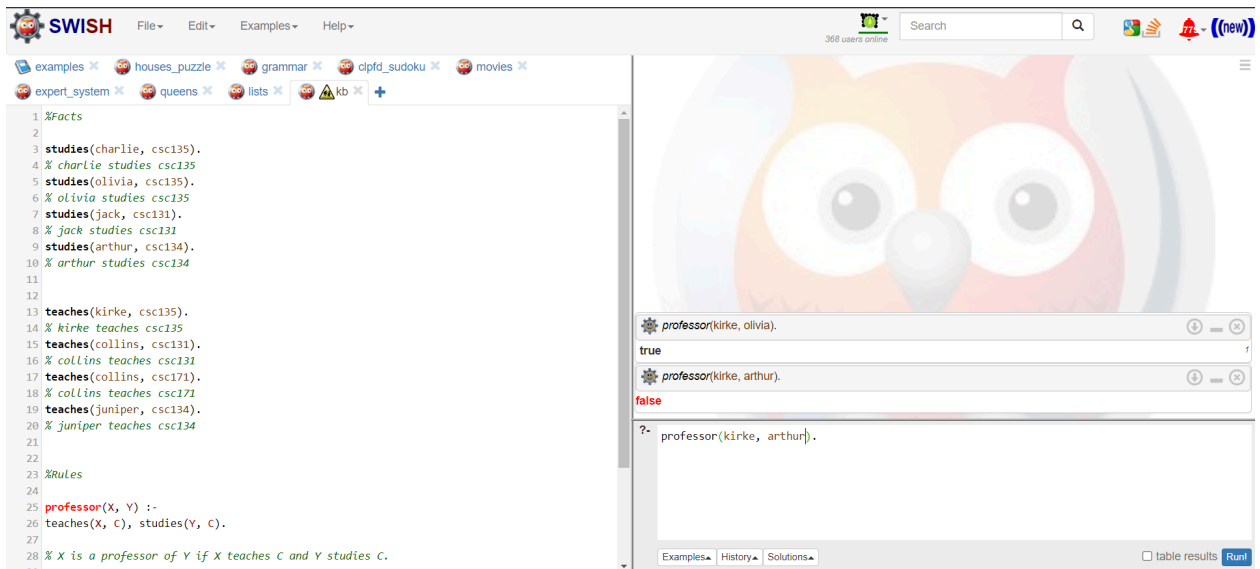
Rows =

9	8	7	6	5	4	3	2	1
2	4	6	1	7	3	9	8	5
3	5	1	9	2	8	7	4	6
1	2	8	5	3	7	6	9	4
6	3	4	8	9	2	1	5	7
7	9	5	4	6	1	8	3	2
5	1	9	2	8	6	4	7	3
4	7	2	3	1	9	5	6	8
8	6	3	7	4	5	2	1	9

?- problem(1, Rows), sudoku(Rows).

Examples History Solutions table results Run!

5. Teacher.pl



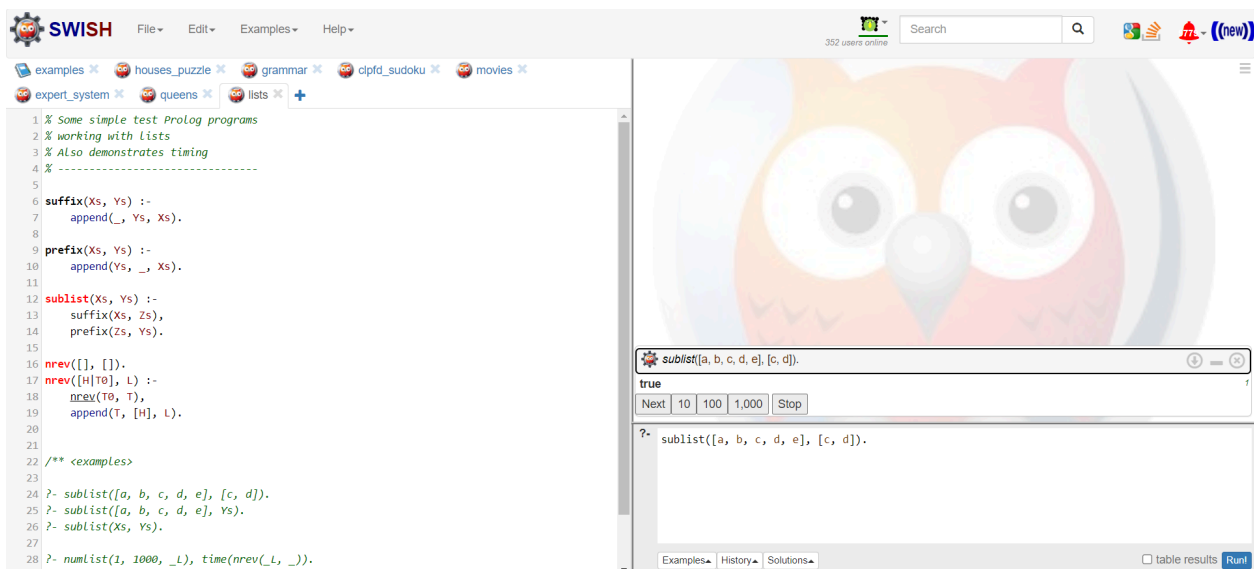
The screenshot shows the SWISH Prolog IDE interface. The left pane displays the source code for `Teacher.pl`, which defines facts about people studying and teaching, and a rule for the `professor` predicate. The right pane shows the execution results, including the query `professor(kirke, olivia).` returning `true`, `professor(kirke, arthur).` returning `false`, and a prompt for `professor(kirke, arthur).`

```
1 %Facts
2
3 studies(charlie, csc135).
4 % charlie studies csc135
5 studies(olivia, csc135).
6 % olivia studies csc135
7 studies(jack, csc131).
8 % jack studies csc131
9 studies(arthur, csc134).
10 % arthur studies csc134
11
12
13 teaches(kirke, csc135).
14 % kirke teaches csc135
15 teaches(collins, csc131).
16 % collins teaches csc131
17 teaches(collins, csc171).
18 % collins teaches csc171
19 teaches(juniper, csc134).
20 % juniper teaches csc134
21
22
23 %Rules
24
25 professor(X, Y) :-
26   teaches(X, C), studies(Y, C).
27
28 % X is a professor of Y if X teaches C and Y studies C.
```

Execution results:

```
professor(kirke, olivia).
true
professor(kirke, arthur).
false
?- professor(kirke, arthur).
```

6. Lists.pl



The screenshot shows the SWISH Prolog IDE interface. The left pane displays the source code for `Lists.pl`, which defines predicates for list operations: `suffix`, `prefix`, `sublist`, `nrev`, and `numList`. The right pane shows the execution results, including the query `sublist([a, b, c, d, e], [c, d]).` returning `true`, and a prompt for `sublist([a, b, c, d, e], [c, d]).`

```
1 % Some simple test Prolog programs
2 % working with Lists
3 % Also demonstrates timing
4 % -----
5
6 suffix(Xs, Ys) :-
7   append(_, Ys, Xs).
8
9 prefix(Xs, Ys) :-
10  append(Ys, _, Xs).
11
12 sublist(Xs, Ys) :-
13   suffix(Xs, Zs),
14   prefix(Zs, Ys).
15
16 nrev([], []).
17 nrev([_|T0], L) :-
18   nrev(T0, T),
19   append(T, [_], L).
20
21
22 /** <examples>
23
24 ?- sublist([a, b, c, d, e], [c, d]).
25 ?- sublist([a, b, c, d, e], Ys).
26 ?- sublist(Xs, Ys).
27
28 ?- numList(1, 1000, _L), time(nrev(_L, _)).
```

Execution results:

```
sublist([a, b, c, d, e], [c, d]).
true
Next 10 100 1,000 Stop
?- sublist([a, b, c, d, e], [c, d]).
```

7. Meal.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the code for `Meal.pl`:

```
1 %Facts
2
3 food(burger). % burger is a food
4 food(sandwich). % sandwich is a food
5 food(pizza). % pizza is a food
6 lunch(sandwich). % sandwich is a lunch
7 dinner(pizza). % pizza is a dinner
8
9
10 %Rules
11 meal(X) :- food(X).
12
13 %Every food is a meal OR
14 %Anything is a meal if it is a food
15
16
17 /* Queries / Goals
18 ?- food(pizza).
19 % Is pizza a food?
20 ?- meal(X), lunch(X).
21 % Which food is meal and lunch?
22 ?- dinner(sandwich).
23 % Is sandwich a dinner?
24 */
```

The right pane shows the execution results for the query `meal(pizza).`, which returns `true`. Below it, the query `lunch(pizza).` returns `false`. At the bottom, there is a query prompt `?- lunch(pizza).` and a `Run!` button.

8. Movies.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the code for `Movies.pl`:

```
42 /* DATABASE
43
44 movie(M, Y) <- movie M came out in year Y
45 director(M, D) <- movie M was directed by director D
46 actor(M, A, R) <- actor A played role R in movie M
47 actress(M, A, R) <- actress A played role R in movie M
48
49 */
50
51 :- discontiguous
52     movie/2,
53     director/2,
54     actor/3,
55     actress/3.
56
57 movie(american_beauty, 1999).
58 director(american_beauty, sam_mendes).
59 actor(american_beauty, kevin_spacey, lester_burnham).
60 actress(american_beauty, annette_bening, carolyn_burnham).
61 actress(american_beauty, thora_birch, jane_burnham).
62 actor(american_beauty, wes_bentley, ricky_fitts).
63 actress(american_beauty, mena_suvari, angela_hayes).
64 actor(american_beauty, chris_cooper, col_frank_fitts_usmc).
65 actor(american_beauty, peter_gallagher, buddy_kane).
66 actress(american_beauty, allison_janney, barbara_fitts).
67 actor(american_beauty, scott_bakula, jim_olmeyer).
68 actor(american_beauty, sam_robards, jim_berkley).
69 actor(american_beauty, harry_del_sherman, brad_dunne).
```

The right pane shows the execution results for the query `movie(american_beauty, Y).`, which returns `Y = 1999`. Below it, there is a query prompt `?- movie(american_beauty, Y).` and a `Run!` button.

9. NQueens.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the Prolog code for `NQueens.pl`. The code includes comments about rendering solutions nicely, a parameter `Queens` for column numbers, and a reference to Richard A. O'Keefe's *The Craft of ProLog*. The code defines `queens(N, Queens)` to find solutions for the N-Queens problem, using constraints to ensure no two queens share the same row or diagonal. The right pane shows the execution of `queens(8, Queens)`, displaying an 8x8 chessboard with 8 queens placed on different rows and columns. Below the board are buttons for 'Next', '10', '100', '1,000', and 'Stop'. The bottom pane shows the query `?- queens(8, Queens).` and a 'Run!' button.

```
1 % render solutions nicely.
2 :- use_rendering(chess).
3
4 %% queens(+N, -Queens) is nondet.
5 %
6 % @param Queens is a list of column numbers for placing the queens.
7 % @author Richard A. O'Keefe (The Craft of ProLog)
8
9 queens(N, Queens) :-
10     length(Queens, N),
11     board(Queens, Board, 0, N, _, _),
12     queens(Board, 0, Queens).
13
14 board([], [], N, N, _, _).
15 board([_Queens], [Col-Vars|Board], Col0, N, [_VR], VC) :-
16     Col is Col0+1,
17     functor(Vars, f, N),
18     constraints(N, Vars, VR, VC),
19     board(Queens, Board, Col, N, VR, [_VC]).
20
21 constraints(0, _, _) :- !.
22 constraints(N, Row, [R|Rs], [C|Cs]) :-
23     arg(N, Row, R-C),
24     M is N-1,
25     constraints(M, Row, Rs, Cs).
26
27 queens([], _, []).
28 queens([C|Cs], Row0, [Col|Solution]) :-
```

10. Jealous.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the Prolog code for `Jealous.pl`. The code includes comments about simple test Prolog programs and knowledge bases. It defines `loves` facts for Vincent, Mia, Marcelus, Pumpkin, and Honey Bunny. The `jealous(X, Y)` predicate is defined to be true if `X` loves `Y` and `Y` loves someone else. The right pane shows the execution of `?- jealous(X, Y).`, displaying the following results:

```
X = Y, Y = vincent
X = vincent,
Y = marcellus
X = marcellus,
Y = vincent
X = Y, Y = marcellus
X = Y, Y = pumpkin
X = Y, Y = honey_bunny
```

The bottom pane shows the query `?- jealous(X, Y).` and a 'Run!' button.

```
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
```