

Prolog Programs

Name: Devesh Vengurlekar
Roll No: 9766, TE Comps-A
Batch D

1. Houses.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the 'Houses.pl' program code, which includes a logical puzzle about five houses and their owners, pets, cigarettes, and drinks. The right pane shows the execution results for the query `zebra_owner(Owner).`, which returns `Owner = japanese`. The interface also includes a search bar, a list of open files, and a status bar indicating 341 users online.

```
7 /* Houses logical puzzle: who owns the zebra and who drinks water?
8
9      1) Five colored houses in a row, each with an owner, a pet, cigarettes, and a drink.
10     2) The English lives in the red house.
11     3) The Spanish has a dog.
12     4) They drink coffee in the green house.
13     5) The Ukrainian drinks tea.
14     6) The green house is next to the white house.
15     7) The Winston smoker has a serpent.
16     8) In the yellow house they smoke Kool.
17     9) In the middle house they drink milk.
18    10) The Norwegian lives in the first house from the left.
19    11) The Chesterfield smoker lives near the man with the fox.
20    12) In the house near the house with the horse they smoke Kool.
21    13) The Lucky Strike smoker drinks juice.
22    14) The Japanese smokes Kent.
23    15) The Norwegian lives near the blue house.
24
25 Who owns the zebra and who drinks water?
26 */
27
28 % Render the houses term as a nice table.
29 :- use_rendering(table,
30     [header(h('Owner', 'Pet', 'Cigarette', 'Drink', 'color'))]).
31
32 zebra_owner(Owner) :-
33     houses(Hs),
34     member(h(Owner,zebra,_,_,_), Hs).
```

2. Grammar.pl

The screenshot shows the SWISH Prolog IDE interface. The left pane displays the 'Grammar.pl' program code, which defines a simple English DCG grammar for the sentence "John saw a man with a telescope". The right pane shows the execution results for the query `phrase(s(Tree), [john, saw, a, man, with, a, telescope]).`, which returns a parse tree structure for the sentence. The interface also includes a search bar, a list of open files, and a status bar indicating 350 users online.

```
1 % Render parse trees using a tree, but ignore Lists Relies on native SVG
2 % support in the browser. IF THE ANSWER LOOKS EMPTY, COMMENT OR REMOVE
3 % THE LINE BELOW.
4 :- use_rendering(svgtree, [list(false)]).
5
6 % A simple English DCG grammar
7 % =====
8
9 s(s(np,VP)) --> np(np, Num), vp(VP, Num).
10
11 np(np, Num) --> pn(np, Num).
12 np(np(det,H), Num) --> det(det, Num), n(H, Num).
13 np(np(det,H,PP), Num) --> det(det, Num), n(H, Num), pp(PP).
14
15 vp(vp(V,np), Num) --> v(V, Num), np(np, _).
16 vp(vp(V,np,PP), Num) --> v(V, Num), np(np, _), pp(PP).
17
18 pp(pp(p,np)) --> p(p), np(np, _).
19
20 det(det(a), sg) --> [a].
21 det(det(the), _) --> [the].
22
23 pn(pn(john), sg) --> [john].
24
25 n(n(man), sg) --> [man].
26 n(n(men), pl) --> [men].
27 n(n(telescope), sg) --> [telescope].
28
```

3. ExpertSystem.pl

The SWISH Prolog IDE interface displays the code for `ExpertSystem.pl` on the left and the query result on the right. The code defines a meta-interpreter for a tiny expert system.

```
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   writeIn(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(_)).
```

The query `prove(good_pet(tweety)).` is executed, resulting in the following output:

```
has_feathers(tweety)
yes
tweets(tweety)
yes
small(tweety)
yes
true
```

Navigation buttons: Next, 10, 100, 1,000, Stop. A "table results" checkbox is present.

4. Sudoku.pl

The SWISH Prolog IDE interface displays the code for `Sudoku.pl` on the left and the query result on the right. The code defines a Sudoku solver using Prolog.

```
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             [_:_:_:_:_]).
```

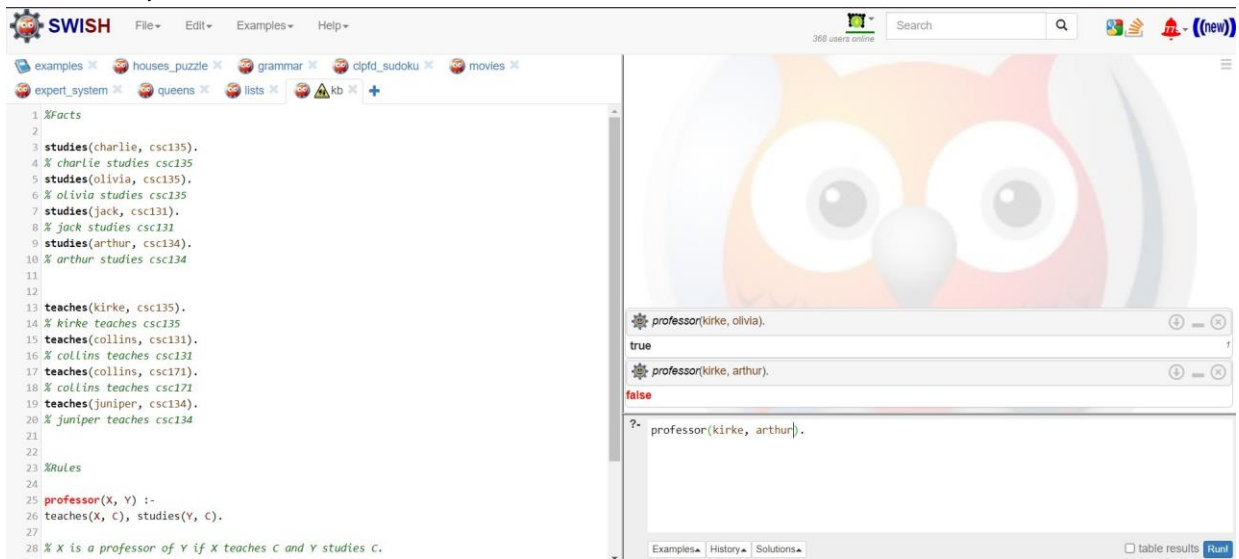
The query `problem(1, Rows), sudoku(Rows).` is executed, resulting in the following output:

```
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

Navigation buttons: Examples, History, Solutions. A "table results" checkbox is present.

5. Teacher.pl



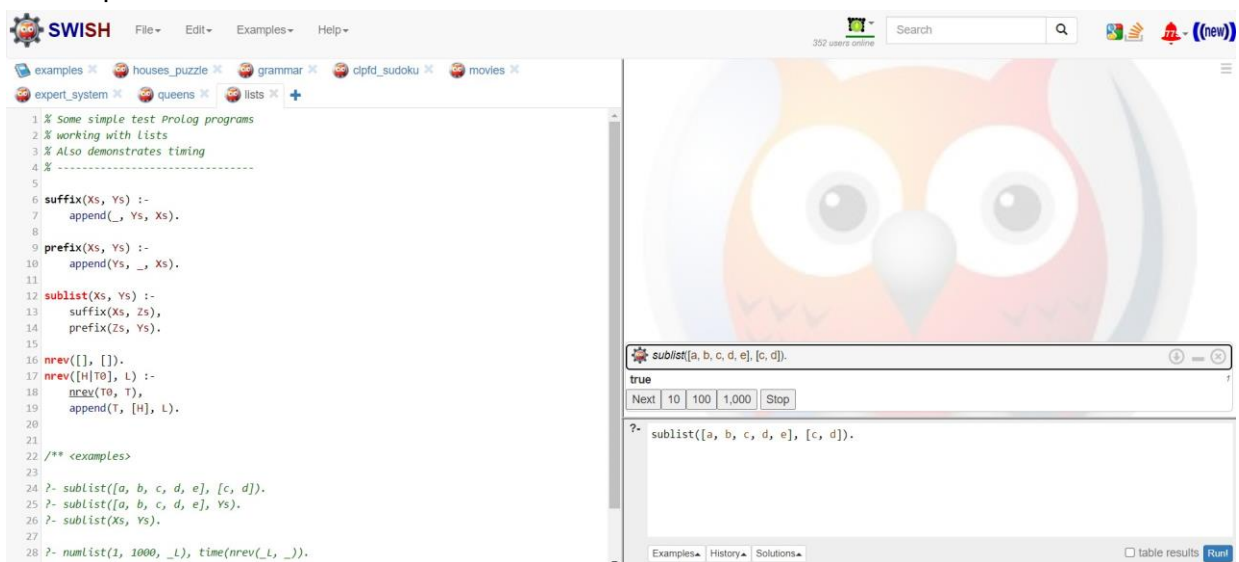
The SWISH Prolog IDE interface displays the file `Teacher.pl`. The left pane shows the following Prolog code:

```
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.
```

The right pane shows the query results for the query `professor(kirke, arthur).`:

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

6. Lists.pl



The SWISH Prolog IDE interface displays the file `Lists.pl`. The left pane shows the following Prolog code:

```
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([H|T0], L) :-
18  nrev(T0, T),
19  append(T, [H], 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, _)).
```

The right pane shows the query results for the query `sublist([a, b, c, d, e], [c, d]).`:

```
?- sublist([a, b, c, d, e], [c, d]).
true
```

7. Meal.pl

The screenshot shows the SWISH Prolog IDE interface. The top bar includes the SWISH logo, menu items (File, Edit, Examples, Help), a search bar, and a user count (377 users online). The left pane displays the 'Meal.pl' file with the following code:

```
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', and 'lunch(pizza).', which returns 'false'. The bottom of the right pane has tabs for 'Examples', 'History', and 'Solutions', and a 'Run' button.

8. Movies.pl

The screenshot shows the SWISH Prolog IDE interface. The top bar includes the SWISH logo, menu items (File, Edit, Examples, Help), a search bar, and a user count (352 users online). The left pane displays the 'Movies.pl' file with the following code:

```
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'. The bottom of the right pane has tabs for 'Examples', 'History', and 'Solutions', 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 the number of queens, and a `queens(N, Queens)` predicate that uses `board/4` and `constraints/4` to find solutions. The right pane shows the execution results for the query `queens(8, Queens).`. It displays a chessboard with 8 queens placed on different rows and columns, and a list of solutions for `Queens`.

```
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]) :-
```

Execution results for `queens(8, Queens).`:

Queens =

Next 10 100 1,000 Stop

?- queens(8, Queens).

Examples History Solutions table results Run

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, and a `jealous(X, Y)` predicate that uses `loves/3` to find solutions. The right pane shows the execution results for the query `jealous(X, Y).`. It displays a list of solutions for `X` and `Y`.

```
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 /** <examples>
16
17 :- loves(X, mia).
18 :- jealous(X, Y).
19
20 */
21
22
23
```

Execution results for `jealous(X, Y).`:

jealous(X, Y).

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

?- jealous(X, Y).

Examples History Solutions table results Run