

AI LAB – 7

Harsh Parmar – 9763 Batch D

Prolog Programmes:

% Some simple test Prolog programs

%

% Knowledge bases

loves(vincent, mia).

loves(marcellus, mia).

loves(pumpkin, honey_bunny).

loves(honey_bunny, pumpkin).

jealous(X, Y) :-

loves(X, Z),

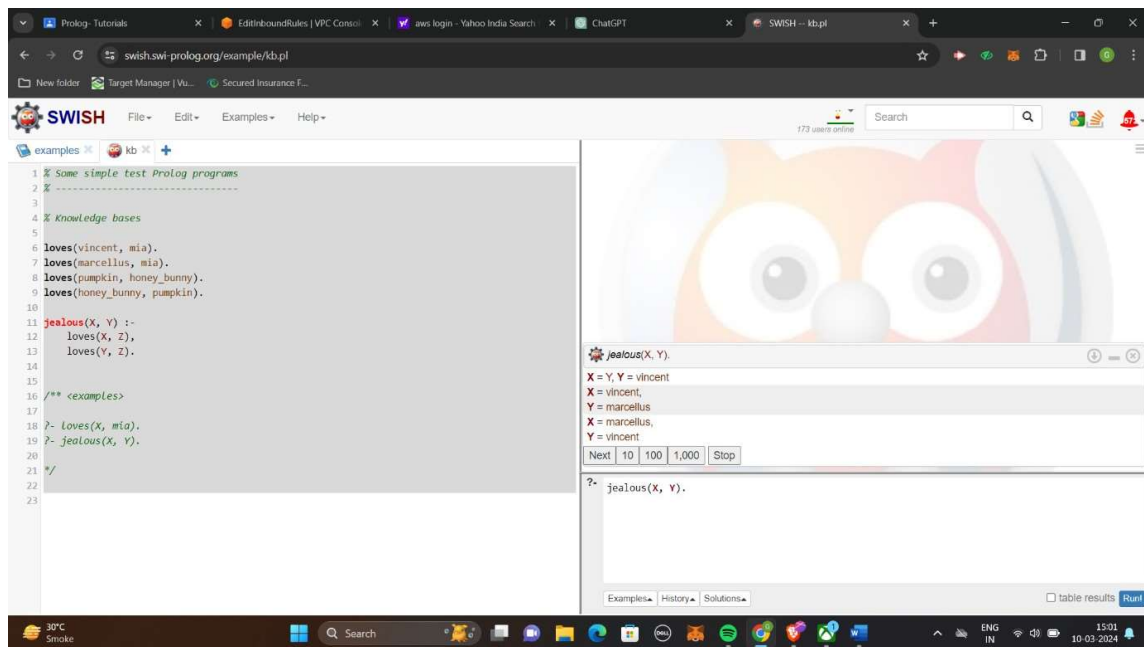
loves(Y, Z).

/** <examples>

?- loves(X, mia).

?- jealous(X, Y).

*/



CODE 2:

% Some simple test Prolog programs

% working with lists

% Also demonstrates timing

%

suffix(Xs, Ys) :-

append(_, Ys, Xs).

prefix(Xs, Ys) :-

append(Ys, _, Xs).

sublist(Xs, Ys) :-

suffix(Xs, Zs),

prefix(Zs, Ys).

nrev([], []).

nrev([H|T0], L) :-

nrev(T0, T),

append(T, [H], L).

/** <examples>

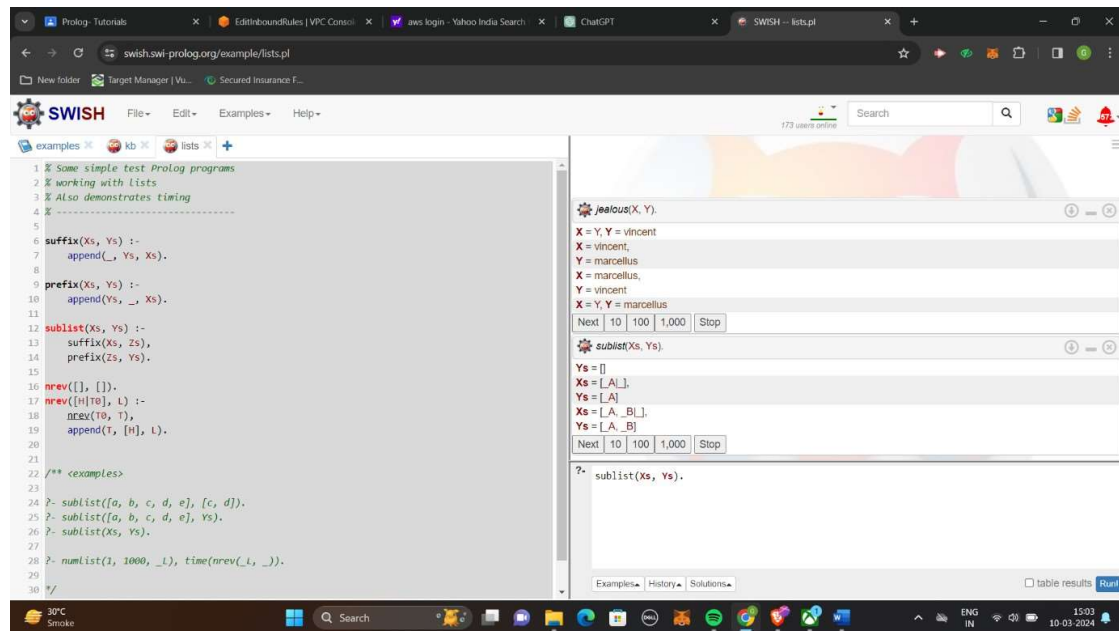
?- sublist([a, b, c, d, e], [c, d]).

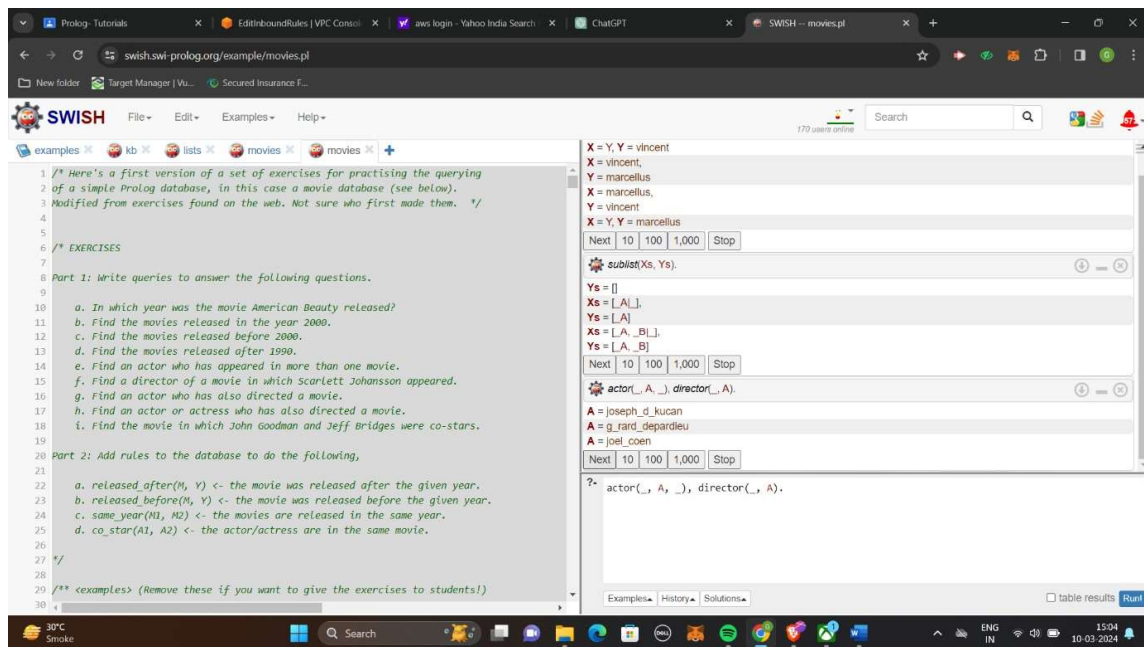
?- sublist([a, b, c, d, e], Ys).

?- sublist(Xs, Ys).

?- numlist(1, 1000, _L), time(nrev(_L, _)).

*/





Code 5:

% A meta-interpreter implementing

% a tiny expert-system

%

prove(true) :- !.

prove((B, Bs)) :- !,

 prove(B),

 prove(Bs).

prove(H) :-

 clause(H, B),

 prove(B).

prove(H) :-

 askable(H),

 writeln(H),

```
read(Answer),  
    Answer == yes.
```

```
good_pet(X) :- bird(X), small(X).  
good_pet(X) :- cuddly(X), yellow(X).
```

```
bird(X) :- has_feathers(X), tweets(X).
```

```
yellow(tweety).
```

```
askable(tweets(_)).  
askable(small(_)).  
askable(cuddly(_)).  
askable(has_feathers(_)).
```

```
/** <examples>
```

```
?- prove(good_pet(tweety)).
```

```
*/
```

Code 6:

```
%% eliza(+Stimuli, -Response) is det.  
% @param Stimuli is a list of atoms (words).
```

```
% @author Richard A. O'Keefe (The Craft of Prolog)
```

```
eliza(Stimuli, Response) :-
```

```
    template(InternalStimuli, InternalResponse),  
    match(InternalStimuli, Stimuli),  
    match(InternalResponse, Response),  
    !.
```

```
template([s([i,am]),s(X)], [s([why,are,you]),s(X),w('?')]).
```

```
template([w(i),s(X),w(you)], [s([why,do,you]),s(X),w(me),w('?')]).
```

```
match([],[]).
```

```
match([Item | Items],[Word | Words]) :-
```

```
    match(Item, Items, Word, Words).
```

```
match(w(Word), Items, Word, Words) :-
```

```
    match(Items, Words).
```

```
match(s([Word | Seg]), Items, Word, Words0) :-
```

```
    append(Seg, Words1, Words0),
```

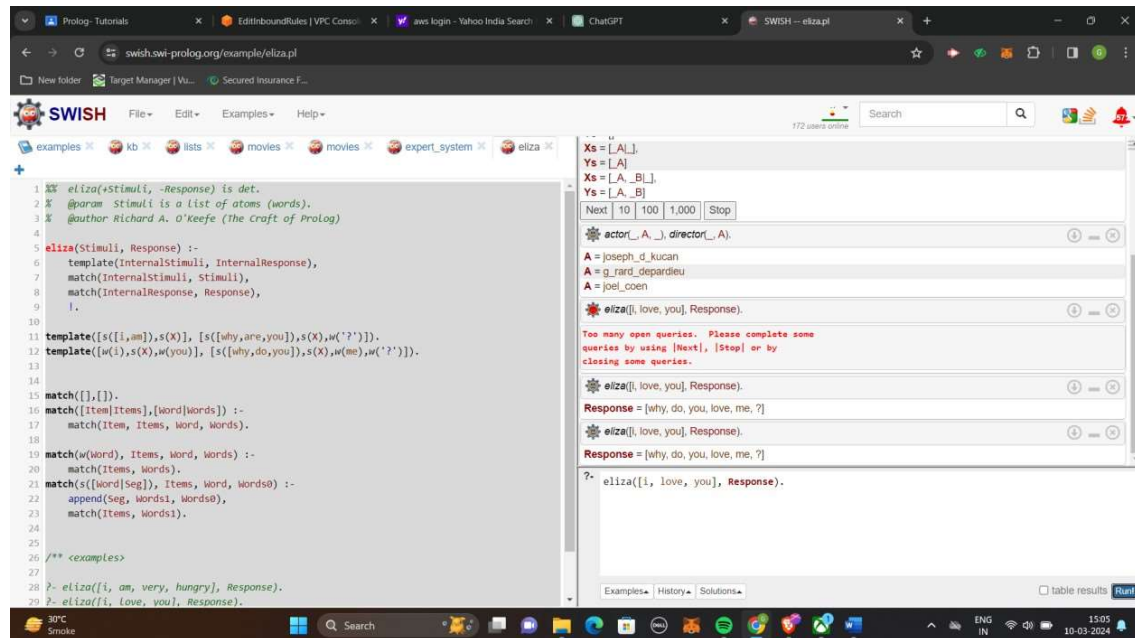
```
    match(Items, Words1).
```

```
/** <examples>
```

```
?- eliza([i, am, very, hungry], Response).
```

?- eliza([i, love, you], Response).

*/



Code :

% Render parse trees using a tree, but ignore lists Relies on native SVG

% support in the browser. IF THE ANSWER LOOKS EMPTY, COMMENT OR REMOVE

% THE LINE BELOW.

:- use_rendering(svgtree, [list(false)]).

% A simple English DCG grammar

% =====

s(s(NP,VP)) --> np(NP, Num), vp(VP, Num).

np(NP, Num) --> pn(NP, Num).

np(np(Det,N), Num) --> det(Det, Num), n(N, Num).

np(np(Det,N,PP), Num) --> det(Det, Num), n(N, Num), pp(PP).

vp(vp(V,NP), Num) --> v(V, Num), np(NP, _).

vp(vp(V,NP,PP), Num) --> v(V, Num), np(NP, _), pp(PP).

pp(pp(P,NP)) --> p(P), np(NP, _).

det(det(a), sg) --> [a].

det(det(the), _) --> [the].

pn(pn(john), sg) --> [john].

n(n(man), sg) --> [man].

n(n(men), pl) --> [men].

n(n(telescope), sg) --> [telescope].

v(v(sees), sg) --> [sees].

v(v(see), pl) --> [see].

v(v(saw), _) --> [saw].

p(p(with)) --> [with].

/** <examples>

?- phrase(s(Tree), [john, saw, a, man, with, a, telescope]).


```
?- phrase(s(Tree), Sentence).
```

```
?- between(1, 8, N), length(S, N), phrase(s(_), S), writeln(S), sleep(0.2), false.
```

```
*/
```

```
Code:
```

```
% render solutions nicely.
```

```
:- use_rendering(chess).
```

```
%% queens(+N, -Queens) is nondet.
```

```
%
```

```
% @param Queens is a list of column numbers for placing the queens.
```

```
% @author Richard A. O'Keefe (The Craft of Prolog)
```

```
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).
```

```
/** <examples>
```

```
?- queens(8, Queens).
```

```
*/
```

Prolog - Tutorials | VPC Console | aws login - Yahoo India Search | ChatGPT | SWISH -- queens.pl

swish.swi-prolog.org/example/queens.pl


169 users online

examples kb lists movies expert_system eliza

queens

```
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]) :-
29   Row is Row0+1,
30   select(Col-Vars, [C|Cs], Board),
31   arg(Row, Vars, Row-Row),
32   queens(Board, Row, Solution).
33
34
35 /** <examples>
36
37 ?- queens(8, Queens).
38
39 */
40
```

Queens =



Next 10 100 1,000 Stop

?- queens(8, Queens).

Examples History Solutions

table results Run

30°C Smoke

Search

ENG IN

1516

10-03-2024