

## My Program:

### 1st program:

```
% facts
m([sam, chris, marshall, steven, jack, sean, james]).
f([jenny, myra, claire, mari, penelope, nicole, elizabeth]).

family([sam, jenny, [steven, claire]]).
family([marshall, myra, [jack, sean]]).
family([chris, claire, [penelope, nicole, elizabeth]]).
family([steven, elizabeth, [james]]).

% rules
male(X) :-
    m(Male),
    member(X, Male).
female(X) :-
    f(Female),
    member(X, Female).

father(F, X) :-
    family([F, _, Children]),
    member(X, Children).
mother(M, X) :-
    family([_, M, Children]),
    member(X, Children).
parent(P, X) :-
    father(P, X).
parent(P, X) :-
    mother(P, X).

siblings1(X, Y) :-
    family([_, _, Children]),
    member(X, Children),
    member(Y, Children),
    X \= Y.
siblings2(X, Y) :-
    siblings1(X, Y).
```

brother1(X, Y) :-  
    male(X),  
    siblings1(X, Y).

brother2(X, Y) :-  
    brother1(X, Y).

sister1(X, Y) :-  
    female(X),  
    siblings1(X, Y).

sister2(X, Y) :-  
    sister1(X, Y).

cousins(X, Y) :-  
    parent(PX, X),  
    parent(PY, Y),  
    siblings1(PX, PY).

uncle(U, Z) :-  
    male(U),  
    parent(P, Z),  
    siblings1(U, P).

aunt(A, Z) :-  
    female(A),  
    parent(P, Z),  
    siblings1(A, P).

grandchild(GC, B) :-  
    parent(B, P),  
    parent(P, GC).

grandson(GS, B) :-  
    male(GS),  
    grandchild(GS, B).

granddaughter(GD, B) :-  
    female(GD),  
    grandchild(GD, B).

greatgrandparent(GGP, G) :-  
    parent(GGP, GP),  
    grandchild(G, GP).

```
ancestor(A, C) :-  
    parent(A, C).  
ancestor(A, C) :-  
    parent(P, C),  
    ancestor(A, P).
```

## 2nd program:

```
membership_test(Element, [Element|_]).  
membership_test(Element, [_|List]) :-  
    membership_test(Element, List).
```

```
first_element(Element, [Element|_]).
```

```
last_element(Last_Element, [Last_Element]).  
last_element(List, [_|Last_Element]) :-  
    last_element(List, Last_Element).
```

```
two_adjacent_elements(Element1, Element2, [Element1, Element2|_]).  
two_adjacent_elements(Element1, Element2, [_|List]) :-  
    two_adjacent_elements(Element1, Element2, List).
```

```
three_adjacent_elements(Element1, Element2, Element3, [Element1, Element2, Element3|_]).  
three_adjacent_elements(Element1, Element2, Element3, [_|List]) :-  
    three_adjacent_elements(Element1, Element2, Element3, List).
```

```
append_list1_to_list2([], List, List).  
append_list1_to_list2([List_Head|List_Tail], List2, [List_Head|Append]) :-  
    append_list1_to_list2(List_Tail, List2, Append).
```

```
delete_element_from_list(Element, [Element|List_Tail], List_Tail).  
delete_element_from_list(Element, [H|List_Tail], [H|Delete]) :-  
    delete_element_from_list(Element, List_Tail, Delete).
```

```
%append_element_to_list([], List, List).
```

```
insert_to_list(Element, 1, List, [Element|List]).  
insert_to_list(Element, Position, [List_Head|List_Tail], [List_Head|Insert]) :-  
    Position > 1,  
    New_Position is Position - 1,
```

```
insert_to_list(Element, New_Position, List_Tail, Insert).
```

```
list_length([], 0).
```

```
list_length([_|List_Tail], Length) :-
```

```
list_length(List_Tail, Updated_Length),
```

```
Length is Updated_Length + 1.
```

```
reverse_list([], []).
```

```
reverse_list([List_Head|List_Tail], Reverse) :-
```

```
reverse_list(List_Tail, Reversed_Tail),
```

```
append_list1_to_list2(Reversed_Tail, [List_Head], Reverse).
```

```
palindrome_list_check([_]).
```

```
palindrome_list_check([List_Head|List_Tail]) :-
```

```
palindrome_list_check(List_Head, List_Tail).
```

```
display_list([List_Head|List_Tail]) :-
```

```
write(List_Head),
```

```
write(' '),
```

```
display_list(List_Tail).
```

### 3rd program:

```
safe_movement ([]) .
```

```
safe_movement ([Queen |Not_Queen]) :-
```

```
safe_movement (Not_Queen) ,
```

```
do_not_move (Queen, Not_Queen,1) .
```

```
do_not_move (_,[],_) .
```

```
do_not_move (Current_Position, [Possible_Position|Queen] , Previous_Position) :-
```

```
Possible_Position - Current_Position =\= Previous_Position,
```

```
Current_Position - Possible_Position =\= Previous_Position,
```

```
do_not_move (Current_Position, Queen, Previous_Position).
```

```
possible_movements (_,[],_) .
```

```
bad_movements ([], []) .
```

```
make_columns ( [], _) .
```

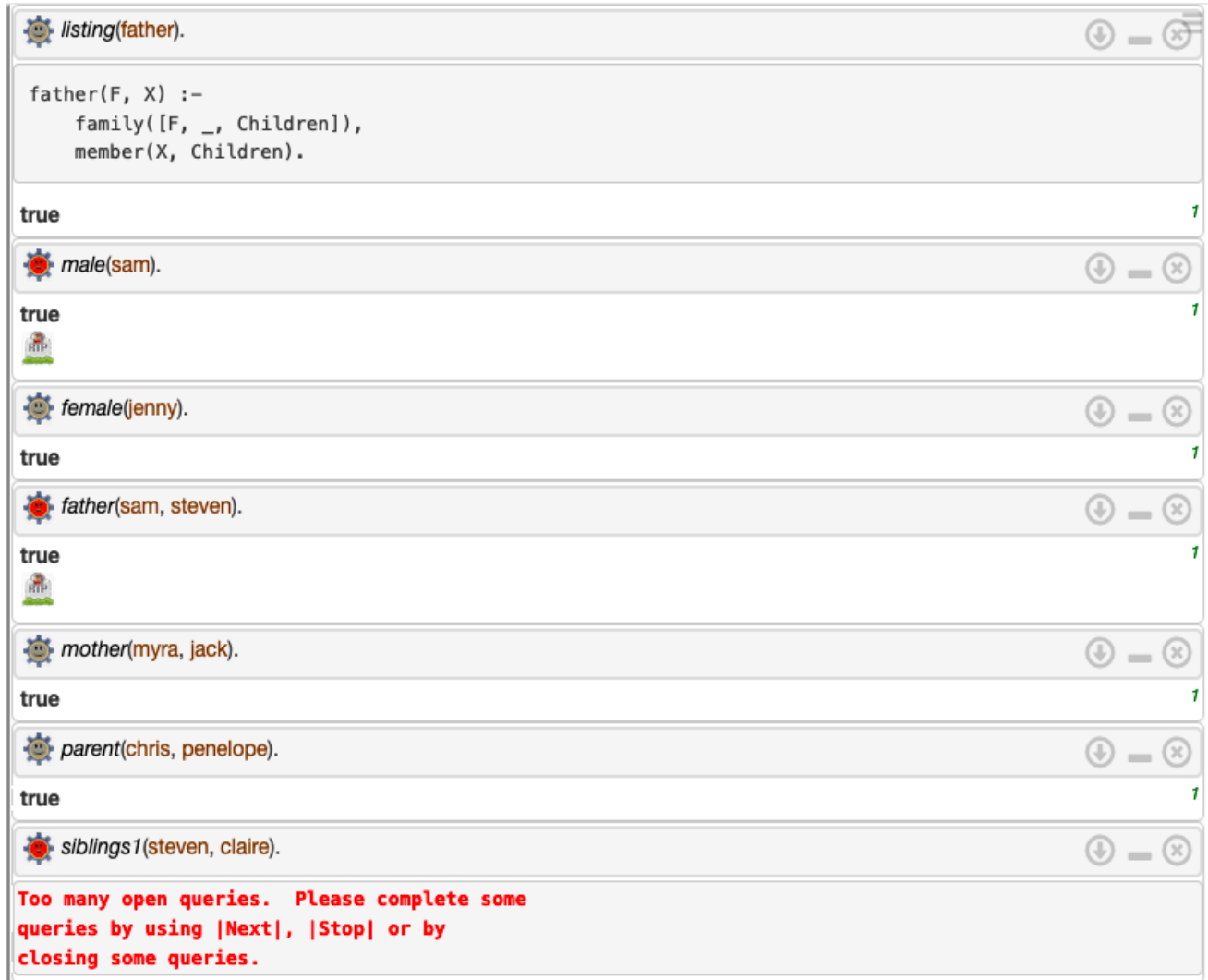
```
make_rows([], _) .
```

```
placing_queens (Eight_Queens) :-  
    possible_movements ([1,2,3,4,5,6,7,8] , Eight_Queens) ,  
    safe_movement (Eight_Queens) .
```

```
make_moves :-  
    make_columns (A, 1) ,  
    make_row (1, A) ,  
    write ("-----") ,  
    placing_queens (Queen) .
```

## Output:

### 1st Program's Output:



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and the text `listing(father).`. The window has three buttons on the right: a download icon, a minus icon, and a close icon. The main area displays a series of queries and their results, each preceded by a gear icon and followed by a green '1' in the right margin.

```
father(F, X) :-  
    family([F, _, Children]),  
    member(X, Children).
```

true 1

```
male(sam).
```

true 1

```
female(jenny).
```

true 1

```
father(sam, steven).
```

true 1

```
mother(myra, jack).
```































true 1




























```
parent(chris, penelope).
```

true 1

```
siblings1(steven, claire).
```







































Too many open queries. Please complete some queries by using |Next|, |Stop| or by closing some queries.

 <code>siblings1(steven, claire).</code>	 
<code>true</code>	<code>1</code>
	
 <code>siblings2(penelope, nicole).</code>	 
<code>true</code>	<code>1</code>
	
 <code>brother1(jack, sean).</code>	 
<code>true</code>	<code>1</code>
 <code>brother2(jack, sean).</code>	 
<code>true</code>	<code>1</code>
 <code>sister1(penelope, nicole).</code>	 
<code>true</code>	<code>1</code>
 <code>sister2(penelope, claire).</code>	 
<code>false</code>	
 <code>cousins(steven,nicole).</code>	 
<code>false</code>	
 <code>cousins(james,penelope).</code>	 
<code>true</code>	<code>1</code>
 <code>uncle(steven,nicole).</code>	 
<code>true</code>	<code>1</code>
	

 <code>uncle(steven,nicole).</code>	 
true	1
 <code>aunt(claire,james).</code>	 
true	1
 <code>grandchild(sam,penelope).</code>	 
false	
 <code>grandchild(penelope,sam).</code>	 
true	1
 <code>grandson(james,chris).</code>	 
true	1
 <code>granddaughter(nicole,sam).</code>	 
true	1
 <code>greatgrandparent(sam,nicole).</code>	 
false	
 <code>greatgrandparent(sam,james).</code>	 
true	1
 <code>ancestor(sam,elizabeth).</code>	 
true	1



## 2nd Program's Output:

 <code>membership_test(2, [1, 2, 4, 5]).</code>	  
<code>true</code>	 <span>1</span>
 <code>membership_test(3, [1, 2, 4, 5]).</code>	  
<code>false</code>	
 <code>first_element(1, [1, 2, 4, 5]).</code>	  
<code>true</code>	 <span>1</span>
 <code>first_element(2, [1, 2, 4, 5]).</code>	  
<code>false</code>	
 <code>last_element(5, [1, 2, 4, 5]).</code>	  
<code>true</code>	 <span>1</span>
 <code>last_element(2, [1, 2, 4, 5]).</code>	  
<code>false</code>	
 <code>two_adjacent_elements(<b>Element1</b>, <b>Element2</b>, [1, 2, 4, 5, 7, 8, 9]).</code>	  
<code><b>Element1</b> = 1,</code>	
<code><b>Element2</b> = 2</code>	
<code><b>Element1</b> = 2,</code>	
<code><b>Element2</b> = 4</code>	
<code><b>Element1</b> = 4,</code>	
<code><b>Element2</b> = 5</code>	
<code><b>Element1</b> = 5,</code>	
<code><b>Element2</b> = 7</code>	
<code><b>Element1</b> = 7,</code>	
<code><b>Element2</b> = 8</code>	
<code><b>Element1</b> = 8,</code>	
<code><b>Element2</b> = 9</code>	
<code>false</code>	
 <code>three_adjacent_elements(<b>Element1</b>, <b>Element2</b>, <b>Element3</b>, [1, 2, 4, 5, 7, 8, 9]).</code>	  
<code><b>Element1</b> = 1,</code>	
<code><b>Element2</b> = 2,</code>	
<code><b>Element3</b> = 4</code>	
<code><b>Element1</b> = 2,</code>	
<code><b>Element2</b> = 4,</code>	
<code><b>Element3</b> = 5</code>	
<code><b>Element1</b> = 4,</code>	
<code><b>Element2</b> = 5,</code>	
<code><b>Element3</b> = 7</code>	
<code><b>Element1</b> = 5,</code>	
<code><b>Element2</b> = 7,</code>	
<code><b>Element3</b> = 8</code>	
<code><b>Element1</b> = 7,</code>	
<code><b>Element2</b> = 8,</code>	
<code><b>Element3</b> = 9</code>	
<code>false</code>	
 <code>append_list1_to_list2([1, 2], [3, 4], <b>Append</b>).</code>	  
<code><b>Append</b> = [1, 2, 3, 4]</code>	



## Screenshots of my code:

### 1st Program:

The screenshot shows the SWISH Prolog IDE interface. The main editor displays a Prolog program defining a family database. The program includes facts, rules for gender, parentage, and siblings, and a query for grandchild relationships.

```
1 % facts
2 m([sam, chris, marshall, steven, jack, sean, james]).
3 f([jenny, myra, claire, mari, penelope, nicole, elizabeth]).
4
5 family([sam, jenny, [steven, claire]]).
6 family([marshall, myra, [jack, sean]]).
7 family([chris, claire, [penelope, nicole, elizabeth]]).
8 family([steven, elizabeth, [james]]).
9
10 % rules
11 male(X) :-
12     m(Male),
13     member(X, Male).
14 female(X) :-
15     f(Female),
16     member(X, Female).
17
18 father(F, X) :-
19     family([F, _, Children]),
20     member(X, Children).
21 mother(M, X) :-
22     family([_, M, Children]),
23     member(X, Children).
24 parent(P, X) :-
25     father(P, X).
26 parent(P, X) :-
27     mother(P, X).
28
29 siblings1(X, Y) :-
30     family([_, _, Children]),
31     member(X, Children),
32     member(Y, Children).
```

On the right side, the execution results are shown for the query `grandchild(sam, penelope).`. The results are as follows:

- `uncle(steven, n`: true
- `aunt(claire, james).`: true
- `grandchild(sam, penelope).`: false
- `grandchild(per sam).`: true

The interface also shows a search bar, a menu with File, Edit, Examples, and Help, and a status bar indicating 188 users online.

← → ↻ <https://swish.swi-prolog.org> ☆ 189 users online Search

**SWISH** File Edit Examples Help

Program

```

29 siblings1(X, Y) :-
30     family([_, _], Children),
31     member(X, Children),
32     member(Y, Children),
33     X \= Y.
34 siblings2(X, Y) :-
35     siblings1(X, Y).
36
37 brother1(X, Y) :-
38     male(X),
39     siblings1(X, Y).
40 brother2(X, Y) :-
41     brother1(X, Y).
42
43 sister1(X, Y) :-
44     female(X),
45     siblings1(X, Y).
46 sister2(X, Y) :-
47     sister1(X, Y).
48
49 cousins(X, Y) :-
50     parent(PX, X),
51     parent(PY, Y),
52     siblings1(PX, PY).
53
54 uncle(U, Z) :-
55     male(U),
56     parent(P, Z),
57     siblings1(U, P).
58 aunt(A, Z) :-
59     female(A),
60     parent(P, Z),

```

uncle(steven,n  
true  
aunt(claire,  
james).  
true  
grandchild(san  
penelope).  
false  
grandchild(per  
sam).  
true  
Examples  
Run! Solutions

← → ↻ <https://swish.swi-prolog.org> ☆ 189 users online Search

**SWISH** File Edit Examples Help

Program


```

50 parent(PX, X),
51 parent(PY, Y),
52 siblings1(PX, PY).
53
54 uncle(U, Z) :-
55     male(U),
56     parent(P, Z),
57     siblings1(U, P).
58 aunt(A, Z) :-
59     female(A),
60     parent(P, Z),
61     siblings1(A, P).
62
63 grandchild(GC, B) :-
64     parent(B, P),
65     parent(P, GC).
66 grandson(GS, B) :-
67     male(GS),
68     grandchild(GS, B).
69 granddaughter(GD, B) :-
70     female(GD),
71     grandchild(GD, B).
72
73 greatgrandparent(GGP, G) :-
74     parent(GGP, GP),
75     grandchild(G, GP).
76
77 ancestor(A, C) :-
78     parent(A, C).
79 ancestor(A, C) :-
80     parent(P, C),
81     ancestor(A, P).

```





uncle(steven,n  
true  
aunt(claire,  
james).  
true  
grandchild(san  
penelope).  
false  
grandchild(per  
sam).  
true  
Examples  
Run! Solutions

## 2nd Program:

**SWISH** File Edit Examples Help  
136 users online Search  
Program +

```
1 membership_test(Element, [Element|_]).
2 membership_test(Element, [_|List]) :-
3   membership_test(Element, List).
4
5 first_element(Element, [Element|_]).
6
7 last_element(Last_Element, [Last_Element]).
8 last_element(List, [_|Last_Element]) :-
9   last_element(List, Last_Element).
10
11 two_adjacent_elements(Element1, Element2, [Element1, Element2|_]).
12 two_adjacent_elements(Element1, Element2, [_|List]) :-
13   two_adjacent_elements(Element1, Element2, List).
14
15 three_adjacent_elements(Element1, Element2, Element3, [Element1, Element2, Element3|_]).
16 three_adjacent_elements(Element1, Element2, Element3, [_|List]) :-
17   three_adjacent_elements(Element1, Element2, Element3, List).
18
19 append_list1_to_list2([], List, List).
20 append_list1_to_list2([List_Head|List_Tail], List2, [List_Head|Append]) :-
21   append_list1_to_list2(List_Tail, List2, Append).
22
23 delete_element_from_list(Element, [Element|List_Tail], List_Tail).
24 delete_element_from_list(Element, [_|List_Tail], [_|Delete]) :-
25   delete_element_from_list(Element, List_Tail, Delete).
26
27 %append_element_to_list([], List, List).
28
29 insert_to_list(Element, 1, List, [Element|List]).
30 insert_to_list(Element, Position, [List_Head|List_Tail], [List_Head|Insert]) :-
31   Position > 1,
32   New_Position is Position - 1,
33   insert_to_list(Element, New_Position, List_Tail, Insert).
34
35 list_length([], 0).
36 list_length([_|List_Tail], Length) :-
37   list_length(List_Tail, Updated_Length),
38   Length is Updated_Length + 1.
39
40 reverse_list([], []).
41 reverse_list([List_Head|List_Tail], Reverse) :-
42   reverse_list(List_Tail, Reversed_Tail),
43   append_list1_to_list2(Reversed_Tail, [List_Head], Reverse).
44
45 palindrome_list_check([_]).
46 palindrome_list_check([List_Head|List_Tail]) :-
47   palindrome_list_check(List_Head, List_Tail).
48
49
50 display_list([List_Head|List_Tail]) :-
51   write(List_Head),
52   write(' '),
53   display_list(List_Tail).
```

### 3rd Program:

 **SWISH** File Edit Examples Help  
175 users online Search   
Program  

```
1 safe_movement([]).
2 safe_movement([Queen|Not_Queen]) :-
3   safe_movement(Not_Queen),
4   do_not_move(Queen,Not_Queen,1).
5
6 do_not_move(_,[],_).
7 do_not_move(Current_Position, [Possible_Position|Queen], Previous_Position) :-
8   Possible_Position - Current_Position =\= Previous_Position,
9   Current_Position - Possible_Position =\= Previous_Position,
10  do_not_move(Current_Position, Queen, Previous_Position).
11
12 possible_movements([], []).
13 bad_movements([], []).
14 make_columns([],_).
15 make_row([],_).
16
17 placing_queens(Eight_Queens) :-
18   possible_movements([1,2,3,4,5,6,7,8], Eight_Queens),
19   safe_movement(Eight_Queens).
20
21 make_moves :-
22   make_columns(A, 1),
23   make_row(1, A),
24   write("-----"),
25   placing_queens(Queen).
26
27
28
29
30
31
```