PROLOG-8-Queens – Documentation

Csc 600-01

Keawa Rozet

Code available at: <https://github.com/krozet/PROLOG-8-Queens>

Prompt

Write a PROLOG program solves the 8 queens problem (location of 8 queens on a chess board so that no queens have each other in check, i.e. are not located in the same row/column/diagonal).

What my code does

In my code you will see that everything is largely broken up into two section. The first section handles printing to the screen, while the second section handles the actual logic of the program. I only included two different commands for the user to input, “print\_coordinates” and “print\_board” because I felt these two commands were sufficient enough to show case my solution.

print\_coordinates does exactly what states. It prints off the coordinates for the 8 different queens on the board. You will notice, however, that I decided to use coordinates native to chess rather than cartesian values. This felt more natural and can easily be translated in cartesian coordinates with a=1 to h=8.

print\_board is only slightly more complicated. In “solution” I return a list of the queens’ row coordinates, with their indexes representing the column coordinate they are located at. I recursively traverse each row and write an empty square where there is no queen, and write and a queen square on the coordinate it is located. I then recursively continue this process for the rest of the columns until I have a fully 8-queen-populated chess board. The board is viewed as if it were from the perspective of black.

My solution is simple enough and, yet again, takes full advantage of recursion. First, I create all the permutations of 8 queens on the chess board. I then check if each queen is safe and is not being attacked. If the queen is safe, I continue to the next queen. If a queen is not safe, then I delete that value and check if the next permutation is successful.

Code

commands :-

write("List of commands:"), nl, nl,

write("print\_coordinates"), nl,

write("print\_board"), nl,

nl.

print\_coordinates :-

solution(A),

display\_coordinates(A, 1).

print\_board :-

solution(A),

print\_boundry,

print\_columns(A, 1),

print\_column\_letters,

nl.

print\_square :- write("| ").

print\_queen :- write("|Q").

print\_end\_quare :- write("|"), nl.

print\_boundry :- write(" - - - - - - - - "), nl.

print\_column\_letters :- write(" h g f e d c b a "), nl.

print\_columns([], \_).

print\_columns([H|T], C) :-

write(C), write(" "),

print\_row(H, 1),

C1 is C + 1,

print\_columns(T, C1).

print\_row(\_, 9) :-

print\_end\_quare,

print\_boundry, !.

print\_row(X, Col) :-

X is Col, print\_queen,

Col1 is Col + 1, print\_row(X, Col1).

print\_row(X, Col) :-

X =\= Col, print\_square,

Col1 is Col + 1, print\_row(X, Col1).

solution(Queens) :-

permutation([1,2,3,4,5,6,7,8], Queens),

safe(Queens).

permutation([],[]).

permutation([H|T], P) :-

permutation(T, P1),

mydelete(H, P, P1).

mydelete(X, [X|L], L).

mydelete(X, [Y|L], [Y|L1]) :-

mydelete(X, L, L1).

safe([]).

safe([Queen|Others]) :-

safe(Others),

noattack(Queen,Others,1).

noattack(\_,[],\_).

noattack(Y, [Y1|L], X1) :-

Y1 - Y =\= X1,

Y - Y1 =\= X1,

X2 is X1 + 1,

noattack(Y, L, X2).

display\_coordinates([], 9).

display\_coordinates([H|T], Y) :-

nth1(H, [h,g,f,e,d,c,b,a], X),

write("["), write(X),

write(Y), write("]"), nl,

Y1 is Y + 1,

display\_coordinates(T, Y1).

Example

> swipl

Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)

SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.

Please run ?- license. for legal details.

For online help and background, visit [http://www.swi-prolog.org](http://www.swi-prolog.org/)

For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- [chess].

true.

?- commands.

List of commands:

print\_coordinates

print\_board

true.

?- print\_coordinates.

[d1]

[g2]

[c3]

[h4]

[b5]

[e6]

[a7]

[f8]

true ;

[c1]

[f2]

[d3]

[b4]

[h5]

[e6]

[g7]

[a8]

true ;

[c1]

[e2]

[b3]

[h4]

[f5]

[d6]

[g7]

[a8]

true .

?- print\_board.

- - - - - - - -

1 | | | | |Q| | | |

- - - - - - - -

2 | |Q| | | | | | |

- - - - - - - -

3 | | | | | |Q| | |

- - - - - - - -

4 |Q| | | | | | | |

- - - - - - - -

5 | | | | | | |Q| |

- - - - - - - -

6 | | | |Q| | | | |

- - - - - - - -

7 | | | | | | | |Q|

- - - - - - - -

8 | | |Q| | | | | |

- - - - - - - -

h g f e d c b a

true ;

- - - - - - - -

1 | | | | | |Q| | |

- - - - - - - -

2 | | |Q| | | | | |

- - - - - - - -

3 | | | | |Q| | | |

- - - - - - - -

4 | | | | | | |Q| |

- - - - - - - -

5 |Q| | | | | | | |

- - - - - - - -

6 | | | |Q| | | | |

- - - - - - - -

7 | |Q| | | | | | |

- - - - - - - -

8 | | | | | | | |Q|

- - - - - - - -

h g f e d c b a

true ;

- - - - - - - -

1 | | | | | |Q| | |

- - - - - - - -

2 | | | |Q| | | | |

- - - - - - - -

3 | | | | | | |Q| |

- - - - - - - -

4 |Q| | | | | | | |

- - - - - - - -

5 | | |Q| | | | | |

- - - - - - - -

6 | | | | |Q| | | |

- - - - - - - -

7 | |Q| | | | | | |

- - - - - - - -

8 | | | | | | | |Q|

- - - - - - - -

h g f e d c b a

true .