```
:- use_module(library(pce)).

% Sample graph edges
edge(a, b).
edge(a, c).
edge(b, c).
edge(b, d).
edge(c, d).
edge(d, e).

% Get all unique nodes
nodes(Nodes) :-
   findall(Node, (edge(Node,_); edge(_,Node)), AllNodes),
   sort(AllNodes, Nodes).

% Graph coloring algorithm
graph_coloring(Colors, Coloring) :-
   nodes(Nodes),
   assign_colors(Nodes, Colors, [], Coloring).

assign_colors([], _, Partial, Partial).
assign_colors([Node|Rest], Colors, Partial, Coloring) :-
   member(Color, Colors),
   \+ conflict(Node, Color, Partial),
   assign_colors(Rest, Colors, [Node-Color|Partial], Coloring).

conflict(Node, Color, Partial) :-
   edge(Node, Adj),
   member(Adj-Color, Partial).
conflict(Node, Color, Partial) :-
   edge(Adj, Node),
   member(Adj-Color, Partial).

% GUI to select number of colors
go :-
   nodes(_),  % just to load nodes
   new(D, dialog('Graph Coloring GUI')),
   send(D, append, new(ColorMenu, menu(color_count, cycle))),
   send_list(ColorMenu, append, ['2','3','4','5']),
   send(D, append, button(run, message(@prolog, run_coloring, ColorMenu?selection))),
   send(D, open).

% Run coloring and display result
run_coloring(ColorCountAtom) :-
   atom_number(ColorCountAtom, ColorCount),
   numlist(1, ColorCount, Colors),
   (graph_coloring(Colors, Coloring) ->
      format_coloring(Coloring, Str),
```

```
    display_result(Str)
  ;
    display_result('No coloring possible with given colors')
  ).

% Convert coloring list to readable string
format_coloring(Coloring, Str) :-
  maplist(node_color_to_string, Coloring, StrList),
  atomic_list_concat(StrList, ', ', Str).

node_color_to_string(Node-Color, S) :-
  format(atom(S), '~w-~w', [Node, Color]).

% Display result in GUI dialog
display_result(Str) :-
  new(D, dialog('Graph Coloring Result')),
  send(D, append, text(Str)),
  send(D, open).
```