```
/**
* Shawn Tan
* Benjamin Tan
*
*/
:-lib(eplex).

/*
 * Kirchoff's current rule
 */
preprocess(Circuit,AugCircuit,Points) :-
   (
      foreach(C,Circuit),
      foreach((C,_),AugCircuit),
      fromto([],PIn,POut,PointsDup)
      do
         (C =.. [_,A,B,_] ->
            POut = [A,B|PIn];
            POut = PIn
         )
   ),
   sort(PointsDup,Points).
current_cons(Points,AugCircuit,CurrList) :-
   (
      foreach(P,Points),
      fromto([],ConsIn,ConsOut,CurrList),
      param(AugCircuit) do
         point_con(P,AugCircuit,Currs),
         append(ConsIn,[Currs],ConsOut)
   ).
point_con(Point,AugCircuit,Currs) :-
   (
      foreach((C,I),AugCircuit),
      fromto([],CurIn,CurOut,Currs),param(Point) do
      ((C =.. [_,Point,_,_],Cur = -I ; C =.. [_,_,Point,_],Cur = I) ->
         CurOut = [Cur|CurIn];
         CurOut = CurIn
      )
   ).
/*
 *  Kirchoff's voltage rule
 */
loop((C,_),RestCircuit,[Val|Loop]) :-
   C =.. [battery,A,B,Val],
   (
      fromto(
         (B,  RestCircuit, Loop      ),
```

```
        (P1, InCirc,     [H|OutLoop]),
        (P2, OutCirc,    OutLoop    ),
        (A,  _,          []         )
      ) do
        delete((Seg,I),InCirc,OutCirc),
        ((
          Seg =.. [C,P1,P2,V],
          comp_volt(C,V,I,H)
        );(
          Seg =.. [C,P2,P1,V],
          comp_volt(C,V,I,H1),
          H = -H1
        ))
   ).

comp_volt(resistor,Val,I,-I*Val).
comp_volt(battery,Val,_,Val).
voltage_cons(AugCircuit,VoltList) :-
   (
      foreach(C,AugCircuit),
      fromto([],ConsIn,ConsOut,VoltList),
      param(AugCircuit) do
        (C = (T,_), T =.. [battery|_] ->
          delete(C,AugCircuit,Circuit),
          bagof(Loop,loop(C,Circuit,Loop),Cons),
          append(ConsIn,Cons,ConsOut);
          ConsIn = ConsOut
        )
   ).
compute_voltages(Done,_,Done,Points) :-
   length(Done,N),length(Points,N),!.
compute_voltages(Done,AugCircuit,Voltages,Points) :-
   member((Comp,I),AugCircuit),
   (
      Comp =.. [C,A,B,Val],Dir = 1;
      Comp =.. [C,B,A,Val],Dir = -1
   ),
   not member((B,_),Done),
   member((A,Vpre),Done),!,
   comp_volt(C,Val,I,CompVolt),
   eval(Vpre + Dir*CompVolt,Pot),
   compute_voltages([(B,Pot)|Done],AugCircuit,Voltages,Points).

solve(Circuit,Voltages) :-
   delete(ground(A),Circuit,Circuit1), %assumes only 1.
   preprocess(Circuit1,AugCircuit,Points),
   current_cons(Points,AugCircuit,CurrList),
   voltage_cons(AugCircuit,VoltList),
   eplex_solver_setup(min(0)),
```

```
   (foreach(C,CurrList) do sum(C) $= 0),
   (foreach(C,VoltList) do sum(C) $= 0),
   eplex_solve(_),
   (foreach((_,I),AugCircuit) do eplex_var_get(I,typed_solution,I)),
   print_list(AugCircuit),
   compute_voltages([(A,0)],AugCircuit,Voltages1,Points),
   sort(Voltages1,Voltages).

test(Circuit,Voltages):-
   Circuit = [
      ground(a),
      battery(a,b,10),
      resistor(a,b,10000),
      resistor(b,c,2000),
      resistor(c,a,4000),
      resistor(c,a,8000),
      resistor(c,a,8000)
   ],
   solve(Circuit,Voltages).

/*
   preprocess(Circuit,AugCircuit,Points),
   current_cons(Points,AugCircuit,CurrList),
   voltage_cons(AugCircuit,VoltList),
   eplex_solver_setup(min(0)),
   (foreach(C,CurrList) do sum(C) $= 0),
   (foreach(C,VoltList) do sum(C) $= 0),
   eplex_solve(_),
   (foreach((_,I),AugCircuit) do eplex_var_get(I,typed_solution,I)),
   print_list(AugCircuit),
   compute_voltages([(a,0)],AugCircuit,Voltages,Points).*/
```

/home/shawn/schoolwork/CS4216/PS5/ex2.pl

```
:-lib(ic).
solve(B,L,Prod) :-
   dim(L,[B]),
   %length(L,B),
   length(Prod,B),
   flatten_array(L,ListL),
   Last is B-1,
   ListL :: 0..Last,
   writeln(L),
   (
      foreacharg(A,L),count(I,0,Last),foreach(A*I,Prod),
      param(Last,L) do
         Count = L[A],
         (A $> 0 or Count $=< 0)
   ),
   sum(ListL) #= B,
   sum(Prod) #= B,
   labeling(L).
```

/home/shawn/schoolwork/CS4216/PS5/test.pl

```
gen_dom(D,N,M) :-
   L is M-N+1, length(D,L), ( foreach(E,D),count(I,N,M) do E = I ).

gen(L,D) :-
   foreach(E,L),param(D) do delete(E,D,_).

all_diff(L) :-
   fromto(L,[H|T],T,[]) do \+ member(H,T).

% 1 -> concert
% 2 -> cinema
% 3 -> theatre
% 4 -> exhibition

puzzle(A,B,C,D,E,P,O,S) :-
   gen_dom(DD,1,4), gen([A,B,C,D],DD), gen([E,P,O,S],DD),
   A = 1, B = O, C \= E, P = 2, E = 3,
   all_diff([A,B,C,D]), all_diff([E,P,O,S]).

occ(X,L,R) :- findall(X,member(X,L),Y),length(Y,R).

gen_smart(L,D) :-
   foreach(E,L),fromto(0,Cin,Cout,_),param(D)
   do member(E,D), Cout is Cin+E, Cout =< 10.

solve(L) :-
   length(L1,10),
   gen_dom(D,0,9),
   gen_smart(L1,D),reverse(L1,L),
   ( foreach(E,L),count(I,0,9),param(L) do writeln(L),occ(I,L,E) ).
```