Prolog Supervision 2
Amrita Panesar (ap949)
Supervisor: Ian Lewis

Qn1 - allpaths

```
% arcs(X,L) succeeds if L is the list of nodes to which node X connects
arcs(a,[b,c]).
arcs(b,[d,e]).
arcs(c,[b,f,g]).
arcs(d,[h,i]).
arcs(e,[d,j]).
arcs(f,[b,j]).
arcs(g,[k,l]).
%arcs(i,[]).
arcs(h,[a]).
%arcs(j,[]).
arcs(k,[f]).
arcs(l,[k]).

% contains(X,Y) succeeds if list X contains the element Y
contains([H|T],H).
%contains([H|T],H2) :- H \= H2, contains(T,H2).
% above doesn't work because of b \= _45.. check fails and hence arc fails and hence path fails and hence path(a,g,Path) fails.
contains([H|T],H2) :- contains(T,H2).

% arc(X,Y) succeeds if there is a single-hop connection from X to Y i.e. arcs(X,L) holds and Y is an item in list L
arc(X,Y) :- arcs(X,L), contains(L,Y).

% path(X,Y,Path) succeeds if Path is an ordered list of nodes between X and Y, [X,...,Y]
path(X,Y,Path) :- path2(X,Y,[],Path).

% path2(X,Y,Acc, Path) succeeds if there is a path, Path, from X to Y X,...,Y that does not include any node in Acc
%path2(X,Y,Acc,[X,Y]) :- arc(X,Y),!.
%path2(X,Y,Acc,[X|R]) :- arc(X,X2), not(contains(Acc,X2)), path2(X2,Y,[X2|Acc],R).
path2(X,Y,Acc,[X,Y]) :- arc(X,Y),!.
path2(X,Y,Acc,[X|R]) :- arc(X,X2), not(contains(Acc,X2)), path2(X2,Y,[X,X2|Acc],R).

cost(a,1).
cost(b,2).
```

```
cost(c,3).
cost(d,4).
cost(e,5).
cost(f,6).
cost(g,7).
cost(h,8).
cost(i,9).
cost(j,10).
cost(k,11).
cost(l,12).

% pathcost(Path,Cost) succeeds if Cost is the sum of the costs of all of the elements in list Path
pathcost([],0).
pathcost([H|T],Cost) :- cost(H,C1), pathcost(T,C2), Cost is C1+C2.

% allpaths(X,Y,L) succeeds if L is the list of all paths between nodes X and Y
allpaths(X,Y,L) :- allpaths2(X,Y,[],L),!.
allpaths2(X,Y,Acc,[R|S]) :- path(X,Y,R), not(contains(Acc,R)), allpaths2(X,Y,[R|Acc],S).
allpaths2(X,Y,Acc,Acc).

graph(
    [
        arcs(a,[b,c]),
        arcs(b,[d,e]),
        arcs(c,[b,f,g]),
        arcs(d,[h,i]),
        arcs(e,[d,j]),
        arcs(f,[b,j]),
        arcs(g,[k,l]),
        % arcs(i,[]),
        arcs(h,[a]),
        % arcs(j,[]),
        arcs(k,[f]),
        arcs(l,[k])
    ]).

% g_arc(X,Y) succeeds if graph G contains arcs(X,[..,Y,..])
g_contains(graph([H|T]),X) :-
g_arc2(X,Y) :- g_contains(G, arcs(X,L)), contains(L,Y),!.
g_arc2(X,Y) :- g_contains(G, arcs(Y,L)), contains(L,X).
g_arc(X,Y) :- g_arc2(graph(
    [
        arcs(a,[b,c]),
```

```
        arcs(b,[d,e]),
        arcs(c,[b,f,g]),
        arcs(d,[h,i]),
        arcs(e,[d,j]),
        arcs(f,[b,j]),
        arcs(g,[k,l]),
        % arcs(i,[]),
        arcs(h,[a]),
        % arcs(j,[]),
        arcs(k,[f]),
        arcs(l,[k])
    ]), X,Y).

% g_path(X,Y,Path) succeeds if graph G contains a path from X to Y
g_path(X,Y,Path) :- g_path2(X,Y,[],Path).
g_path2(X,Y,Acc,[X,Y]) :- g_arc(X,Y),!.
g_path2(X,Y,Acc,[X|R]) :- g_arc(X,X2), not(contains(Acc,X2)), g_path2(X2,Y,[X,X2|Acc],R).
```

Qn2 - flatDiffLists

```
% append(R,S,T) succeeds if T is list S appended onto list R.
append([],L,L).
append([X|T],L,[X|R]) :- append(T,L,R).

% flat(X,Y) succeeds if Y is the flattened version of list X.
flat([],[]).
flat([H|T],R) :- flat(H,H2), flat(T,T2),!, append(H2,T2,R).
flat([H|R],[H|S]) :- flat(R,S).

% d_flat(X,Y) succeeds if Y is the flattened difference list of prolog list Y).
d_append(A-B,B-C,A-C).
d_flat([],[]-[]).
% H2, T2 = [] so works. append(H1-H2,T1-T2,R-S) fails.
d_flat([H|T],H1-T1) :- d_flat(H,H1-H2), d_flat(T,T1-T2),!.
d_flat([H|R],[H|T1]-T2) :- d_flat(R,T1-T2).
```