```
#!/usr/bin/python3

# Topological sort by means of a Depth-First Search (DFS) using an
# adjacency-list representation of a graph.

# Graph representation: V(G) = {0,1,2,...,n-1}, The graph is
# represented by an array of arrays G such that G[u]=[v1,v2,...]
# means that the graph contains edges u-->v1, u-->v2, etc.

def topological_sort(G):
    # return an array of nodes, sorted in topological order
    # or None if the graph contains a cycle
    n = len(G)
    T = []                      # T: topological order (array of nodes)
    S = []                      # S: stack used for iterative DFS

    # D[v]: discovery status of v: 0: unknown; 1: discovered; 2: finished
    D = [0]*n

    for u in range(n):
        if D[u] == 0:
            S.append(u)
        while len(S) > 0:
            u = S[-1]           # u = top of the stack
            if D[u] == 0:
                D[u] = 1
                for v in G[u]:
                    if D[v] == 0:
                        S.append(v)
                    elif D[v] == 1:
                        return None
            elif D[u] == 1:
                D[u] = 2
                T.append(u)
                S.pop()
            else:
                S.pop()
    return T

import sys

# We read a directed graph from the standard input. The input consists
# of n lines, each containing the adjacency list of a node.  For
# example:
#
# shirt tie belt                # shirt-->tie, shirt-->belt
# boxers pants shoes
# socks shoes
# watch
# tie jacket
# pants belt shoes
# belt jacket

Adj = []                        # adjacency list
Idx = {}                        # Idx: node name --> node index in Adj
Name = []                       # Name[v] is the name of node v

# Utility function to add a node u to the graph.  u is the node name.
def add_vertex(u_name):
    global Name, Idx, Adj
    if u_name in Idx:
        u = Idx[u_name]
    else:
        u = len(Adj)
        Idx[u_name] = u
        Adj.append([])
        Name.append(u_name)
    return u

import sys

for l in sys.stdin:
    L = l.strip().split()
    if len(L) == 0:
        continue;
    u = add_vertex (L[0])
    for i in range(1,len(L)):
        v = add_vertex(L[i])
        Adj[u].append(v)

S = topological_sort(Adj)
if S == None:
    print('graph contains a cycle')
else:
    for i in range(len(S)-1, -1, -1):
        print(Name[S[i]])
```