// bellman ford and tracking negative cycles

#include<bits/stdc++.h>

using namespace std;

int n,e;

class edge

{

int wt;

int node;

public:

edge(int a,int b)

{

wt=a;node=b;

}

int retwt()

{

return(wt);

}

int retnode()

{

return(node);

}

};

void bellmanford(vector<list<edge>>v,int source)

{

int dist[1000];//array which holds dist from source to all other nodes

for(int i=0;i<1000;i++)

{

dist[i]=INT\_MAX;

}

dist[source]=0;

//edge relaxation should take palce n-1 times

for(int k=0;k<n;k++)

{//vector<list<edge>>::iterator it;

for( int i=1;i<=n;i++)

{

list<edge>::iterator it2;

list<edge>temp=v[i];

for(it2=temp.begin();it2!=temp.end();it2++)

{

//edge relaxation

if(dist[i]+((\*it2).retwt())<dist[(\*it2).retnode()])

{

dist[(\*it2).retnode()]= dist[i]+((\*it2).retwt());

}

}

}

}

for(int i=1;i<=n;i++)

{

cout<<dist[i]<<" ";

}

cout<<"\n";

int dist2[1000];//store all the distances before doing another edge relaxation

for(int i=1;i<=n;i++)

{

dist2[i]=dist[i];

}

// do edge relaxation once more

//vector<list<edge>>::iterator it;

int flag=0;

for( int i=1;i<=n;i++)

{

list<edge>::iterator it2;

list<edge>temp=v[i];

for(it2=temp.begin();it2!=temp.end();it2++)

{

//edge relaxation

if(dist[i]+((\*it2).retwt())<dist[(\*it2).retnode()])

{

dist[(\*it2).retnode()]= dist[i]+((\*it2).retwt());

}

}

}

for(int i=1;i<=n;i++)

{

if(dist[i]!=dist2[i])

{

flag=1;

break;

}

}

if(flag==0)

{

cout<<"no negative cycle:)";

}

else

{

cout<<"negative cycle:(";

}

}

int main()

{

cin>>n>>e;

vector<list<edge>>v;

v.assign(n+1, list<edge>());

for(int i=0;i<e;i++)

{

int a,b;

cin>>a>>b;

int weight;

cin>>weight;

v[a].push\_back(edge(weight,b));

}

bellmanford(v,1);

return 0;

}