**def** bfs(stNode,desNode,graph,nodes,edges):

mainQ=[]

vL=[**False for** i **in** range(nodes)]

cost=[0 **for** i **in** range(nodes)]

mainQ.append(stNode)

**while** mainQ:

stNode=mainQ[0]

**if** stNode == desNode:

print(**"Minimum steps lagche "**, cost[stNode], **" ta"**)

mainQ.pop(0)

**else**:

vL[mainQ.pop(0)] = **True**

**for** i **in** range(len(graph[stNode])):

**if** vL[graph[stNode][i]]!=**True**:

mainQ.append(graph[stNode][i])

vL[graph[stNode][i]]=**True**

cost[graph[stNode][i]]=cost[stNode]+1

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

f = open(**"bfs.txt"**)

nodes = int(f.readline())

edges = int(f.readline())

graph = [[] **for** i **in** range(nodes)]

**for** i **in** range(edges):

u, v = map(int, f.readline().split())

graph[u].append(v)

graph[v].append(u)

destination = int(f.readline())

bfs(0,destination,graph,nodes,edges)

**def** bfs(stNode, desNode, graph, nodes, edges):

mainQ = []

vL = [**False for** i **in** range(nodes)]

cost = [0 **for** i **in** range(nodes)]

mainQ.append(stNode)

**while** mainQ:

stNode = mainQ[0]

**if** stNode == desNode:

**return** cost[stNode]

**else**:

vL[mainQ.pop(0)] = **True**

**for** i **in** range(len(graph[stNode])):

**if** vL[graph[stNode][i]] != **True**:

mainQ.append(graph[stNode][i])

vL[graph[stNode][i]] = **True**

cost[graph[stNode][i]] = cost[stNode] + 1

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

f = open(**"bfs2.txt"**)

nodes = int(f.readline())

edges = int(f.readline())

graph = [[] **for** i **in** range(nodes)]

**for** i **in** range(edges):

u, v = map(int, f.readline().split())

graph[u].append(v)

graph[v].append(u)

destination = int(f.readline())

noraPos=int(f.readline())

laraPos=int(f.readline())

noraSteps=bfs(noraPos, destination, graph, nodes, edges)

laraSteps=bfs(laraPos, destination, graph, nodes, edges)

**if** laraSteps > noraSteps:

print(**"Nora Wins"**)

**elif** laraPos == noraPos :

print(**"It is a tie"**)

**else**:

print(**"Lara Wins"**)

**def** bfs(stNode, desList, graph, nodes, edges):

mainQ = []

vL = [**False for** i **in** range(nodes)]

cost = [0 **for** i **in** range(nodes)]

mainQ.append(stNode)

**while** mainQ:

stNode = mainQ[0]

**if** stNode **in** desList:

print(**"First to catch Lina is K"**,stNode)

**return** cost[stNode] *#returning the number of steps*

**else**:

vL[mainQ.pop(0)] = **True**

**for** i **in** range(len(graph[stNode])):

**if** vL[graph[stNode][i]] != **True**:

mainQ.append(graph[stNode][i])

vL[graph[stNode][i]] = **True**

cost[graph[stNode][i]] = cost[stNode] + 1

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

f = open(**"bfs3.txt"**)

nodes = int(f.readline())

edges = int(f.readline())

graph = [[] **for** i **in** range(nodes)]

**for** i **in** range(edges):

u, v = map(int, f.readline().split())

graph[v].append(u)

**for** i **in** range(nodes):

print(i, **"=> "**,graph[i])

source = int(f.readline()) *#since Linas position is our source now*

desList=[] *#because we need to search for more than 1 destinations this time, we're taking a list*

partiNum=int(f.readline())

**for** i **in** range(partiNum):

desList.append(int(f.readline()))

steps=bfs(source, desList, graph, nodes, edges)

print(**"Steps needed - "**,steps)