# Adjacent Matrix  
G = [[ 0, 1, 1, 0, 1, 0],  
 [ 1, 0, 1, 1, 0, 1],  
 [ 1, 1, 0, 1, 1, 0],  
 [ 0, 1, 1, 0, 0, 1],  
 [ 1, 0, 1, 0, 0, 1],  
 [ 0, 1, 0, 1, 1, 0]]  
  
  
  
# inisiate the name of node.  
node = "abcdef"  
t\_={}  
for i in range(len(G)):  
 t\_[node[i]] = i  
  
# count degree of all node.  
degree =[]  
for i in range(len(G)):  
 degree.append(sum(G[i]))  
  
# inisiate the posible color  
colorDict = {}  
for i in range(len(G)):  
 colorDict[node[i]]=["Blue","Red","Yellow","Green"]  
  
  
# sort the node depends on the degree  
sortedNode=[]  
indeks = []  
  
# use selection sort  
for i in range(len(degree)):  
 \_max = 0  
 j = 0  
 for j in range(len(degree)):  
 if j not in indeks:  
 if degree[j] > \_max:  
 \_max = degree[j]  
 idx = j  
 indeks.append(idx)  
 sortedNode.append(node[idx])  
  
# The main process  
theSolution={}  
for n in sortedNode:  
 setTheColor = colorDict[n]  
 theSolution[n] = setTheColor[0]  
 adjacentNode = G[t\_[n]]  
 for j in range(len(adjacentNode)):  
 if adjacentNode[j]==1 and (setTheColor[0] in colorDict[node[j]]):  
 colorDict[node[j]].remove(setTheColor[0])  
  
  
# Print the solution  
for t,w in sorted(theSolution.items()):  
 print("Node",t," = ",w)