**EXP NO: 04**

**DATE:**

**CONSTRAINT SATISFACTION PROBLEM**

**NAME: NAVEENKUMAR M**

**ROLLNO: 1905097**

**AIM:**

To solve the map coloring problem .

**CODE:**

class Territory:

def \_\_init\_\_(self, name, neighbors, color=None):

self.name = name

self.neighbors = neighbors

self.color = color

map = {

"A1": Territory("A1", ["A2", "H"]),

"A2": Territory("A2", ["A3", "H","A1"]),

"A3": Territory("A3", ["A4", "H", "A2"]),

"A4": Territory("A4", ["H", "A3"]),

"H": Territory("H", ["T", "A1", "A2", "A3", "A4"]),

"T": Territory("T", ["F1", "F2"]),

"F1": Territory("F1", ["T"]),

"F2": Territory("F2", ["T"])

}

m=3

def checkneighbors(x, c):

node=map[list(map.keys())[x]]

for i in node.neighbors:

r=list(map.keys()).index(i)

if color[r] == c:

return False

return True

visited = set()

res = []

color=[0]\*len(map)

def solve(start,color,v):

if len(map)==v:

print(color)

return

elif len(visited)>len(map):

print("Problem can't be solved")

return

for c in range(1,start+1):

if checkneighbors(v, c) :

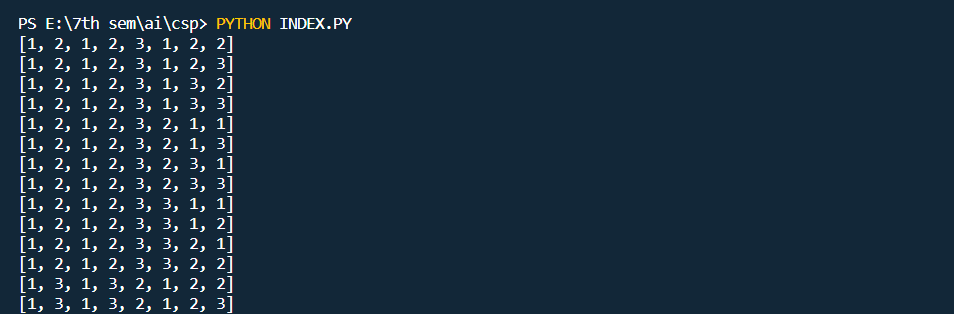
color[v]=c

solve(start, color, v+1)

color[v]=0

solve(m,color,0)

**OUTPUT:**

****

**RESULT:**

Map coloring problem has been executed successfully.