Name :- Ashish Verma Soni

Reg no:- RA1911042010102

Section:- CSBS Q2

**Aim:-**

Implementation of constraint satisfaction problems

**Flow of the program:-**

1. Graph / map coloring problems are those where the nodes are assigned colors such that the adjacent connected nodes / regions don’t have the same color assigned.

2. At the same time, it is required to use the minimum number of colors possible – called the chromatic number.

3. Start by coloring the first node with a color.

4. Color the subsequent connected nodes with a different color.

5. Check at every step that it satisfies the condition.

**CODE:-**

class Graph:

    def \_\_init\_\_(self, edges, N):

        self.adj = [[] for \_ in range(N)]

        for (src, dest) in edges:

            self.adj[src].append(dest)

            self.adj[dest].append(src)

def colorGraph(graph):

    result = {}

    for u in range(N):

        assigned = set([result.get(i) for i in graph.adj[u] if i in result])

        color = 1

        for c in assigned:

            if color != c:

                break

            color = color + 1

        result[u] = color

    for v in range(N):

        print("Color assigned to vertex", v, "is", colors[result[v]])

if \_\_name\_\_ == '\_\_main\_\_':

    colors = ["","BLUE", "GREEN", "RED", "YELLOW", "ORANGE"]

    edges = [(0,1), (0,3), (0,4), (1,0), (1,4), (1,2), (2,1), (2,3), (3,0), (3,4), (3,5), (4,0), (4,1), (4,2), (4,3), (4,5), (4,6), (5,3), (5,6), (5,4), (6,4), (6,5)]

    N = 7

    graph = Graph(edges, N)

    colorGraph(graph)

**Output:**

