**PRACTICAL No.-12 :To implement a map coloring constraint satisfaction problem (CSP) using a simple backtracking approach.**

**Code:**

# Define the regions and their domains

regions = ['WA', 'NT', 'SA', 'Q', 'NSW', 'V']

colors = ['Red', 'Green', 'Blue']

# Define the constraint function

def const\_different(color1, color2):

return color1 != color2

# Define the constraints as pairs of (region1, region2)

constraints = [

(('WA', 'NT'), const\_different),

(('WA', 'SA'), const\_different),

(('SA', 'NT'), const\_different),

(('SA', 'Q'), const\_different),

(('NT', 'Q'), const\_different),

(('SA', 'NSW'), const\_different),

(('Q', 'NSW'), const\_different),

(('SA', 'V'), const\_different),

(('NSW', 'V'), const\_different),

]

# Function to check if the current assignment is valid

def is\_valid(assignment, region, color):

for (region1, region2), constraint in constraints:

if region == region1 and region2 in assignment:

if not constraint(color, assignment[region2]):

return False

elif region == region2 and region1 in assignment:

if not constraint(color, assignment[region1]):

return False

return True

# Backtracking function to solve the CSP

def backtrack(assignment):

if len(assignment) == len(regions):

return assignment # All regions are assigned

# Select the next unassigned region

unassigned\_region = [r for r in regions if r not in assignment][0]

for color in colors:

if is\_valid(assignment, unassigned\_region, color):

assignment[unassigned\_region] = color # Assign color

result = backtrack(assignment) # Recur

if result:

return result

del assignment[unassigned\_region] # Remove assignment (backtrack)

return None # No valid assignment found

# Solve the map coloring problem

solution = backtrack({})

# Print the result

if solution:

print("Coloring solution found:")

for region, color in solution.items():

print(f"{region}: {color}")

else:

print("No solution found.")

**Output:**

