Date 02/03/2024

10.Aim Write the python program for Map Coloring to implement CSP..

Program:

```
class CSP:
    def __init__(self, variables, domains, constraints):
        self.variables = variables
        self.domains = domains
        self.constraints = constraints

    def is_consistent(self, variable, value, assignment):
        for constraint in self.constraints.get(variable, []):
            if constraint[0] in assignment and not constraint[1](value, assignment[constraint[0]]):
                return False
        return True

    def backtracking_search(self, assignment={}):
        if len(assignment) == len(self.variables):
            return assignment

        unassigned = [v for v in self.variables if v not in assignment]
        var = unassigned[0]

        for value in self.domains[var]:
            if self.is_consistent(var, value, assignment):
                assignment[var] = value
                result = self.backtracking_search(assignment)
                if result:
                    return result
                assignment.pop(var)
```

```
        return None


# Example usage:
if __name__ == "__main__":
    # Define variables, domains, and constraints
    variables = ['WA', 'NT', 'SA', 'Q', 'NSW', 'V', 'T']
    domains = {v: ['R', 'G', 'B'] for v in variables}
    constraints = {
        'WA': [('NT', lambda x, y: x != y)],
        'NT': [('WA', lambda x, y: x != y), ('SA', lambda x, y: x != y)],
        'SA': [('WA', lambda x, y: x != y), ('NT', lambda x, y: x != y), ('Q', lambda x, y: x != y), ('NSW', lambda x,
y: x != y), ('V', lambda x, y: x != y)],
        'Q': [('SA', lambda x, y: x != y), ('NSW', lambda x, y: x != y)],
        'NSW': [('SA', lambda x, y: x != y), ('Q', lambda x, y: x != y), ('V', lambda x, y: x != y)],
        'V': [('SA', lambda x, y: x != y), ('NSW', lambda x, y: x != y)],
        'T': []
    }

    csp = CSP(variables, domains, constraints)
    solution = csp.backtracking_search()
    if solution:
        print("Solution found:")
        for var, val in solution.items():
            print(f"{var}: {val}")
    else:
        print("No solution found.")
```

Output:

```
= RESTART: C:/Users/
oloring to implement
Solution found:
WA: R
NT: G
SA: B
Q: R
NSW: G
V: R
T: R
```

Result: The given program has been executed successfully