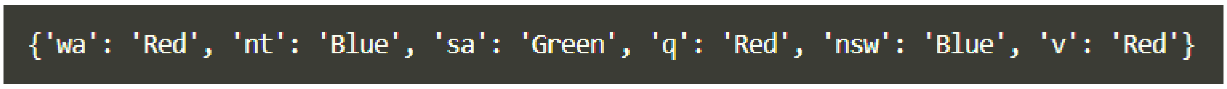
Name: Alekhya Tejomurtula
Class: TY A
Roll No: 58
Srn: 201900449

AI Assignment-5

1) Map Coloring:

```
colors = ['Red', 'Blue', 'Green']
states = ['wa', 'nt', 'sa', 'q', 'nsw', 'v']
neighbors = {}
neighbors['wa'] = ['nt', 'sa']
neighbors['nt'] = ['wa', 'sa', 'q']
neighbors['sa'] = ['wa', 'nt', 'q', 'nsw', 'v']
neighbors['q'] = ['nt', 'sa', 'snw']
neighbors['nsw'] = ['q', 'sa', 'v']
neighbors['v'] = ['sa', 'nsw']

colors_of_states = {}

def promising(state, color):
    for neighbor in neighbors.get(state):
        color_of_neighbor = colors_of_states.get(neighbor)
        if color_of_neighbor == color:
            return False
    return True

def get_color_for_state(state):
    for color in colors:
        if promising(state, color):
            return color

def main():
    for state in states:
        colors_of_states[state] = get_color_for_state(state)
    print(colors_of_states)

if _name_ == "_main_":
    main()
```

Output:

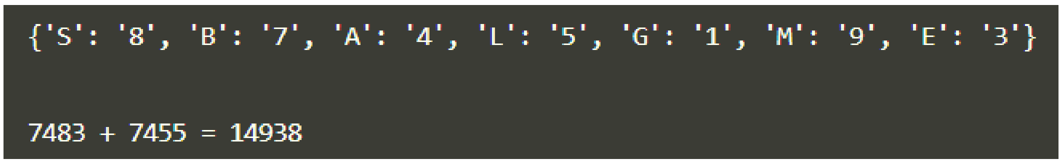


```
{'wa': 'Red', 'nt': 'Blue', 'sa': 'Green', 'q': 'Red', 'nsw': 'Blue', 'v': 'Red'}
```

2) Crypto:

```
import itertools
# import string as str
from tokenize import String

def correct_vals(p, puzzle):
    op1, op, op2, e, r = break_puzzle(puzzle.translate(p))
    return eval(op1 + op + op2 + "==" + r)

def break_puzzle(puzzle):
    return tuple(puzzle.upper().split())

def get_unique_letters(puzzle):
    return [i for i in set(''.join(break_puzzle(puzzle))) if i.isalpha()]

def get_starting_letters(puzzle, letters):
    return [i for i in range(len(letters)) if letters[i] == break_puzzle(puzzle)[0][0] or letters[i] ==
break_puzzle(puzzle)[2][0] or letters[i] == break_puzzle(puzzle)[4][0]]

def get_valid_permutations(puzzle):
    digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    letters = get_unique_letters(puzzle)
    critical_indices = get_starting_letters(puzzle, letters)
    poss_perms = []
    for perm in itertools.permutations(digits, len(letters)):
        flag = 0
        for i in critical_indices:
            if perm[i] == '0':
                flag = 1
                break
        if flag == 0:
            poss_perms.append(perm)
    return poss_perms

def solve(puzzle):
    letters = get_unique_letters(puzzle)
    if len(letters) > 10:
        print("INVALID EQUATION : more than one letter maps to same digit")
        return
    for poss in get_valid_permutations(puzzle):
        p = String.maketrans(''.join(letters), ''.join(poss))
        if correct_vals(p, puzzle):
            answer = dict(zip(letters, poss))
            print("\n",answer,"\n")
            solved_puzzle = puzzle
            for c in answer:
                x = solved_puzzle.replace(c, answer[c])
                solved_puzzle = x
            print("\n",solved_puzzle,"\n")

solve("BASE + BALL = GAMES")
```

Output:



```
{'S': '8', 'B': '7', 'A': '4', 'L': '5', 'G': '1', 'M': '9', 'E': '3'}


7483 + 7455 = 14938
```