NAME:- DIVYANG BAGLA
PANEL:- C PC-33
C-3

PAGE NO.:

AI LAB ASSIGNMENT - 3

TITLE:- Implementation of soln of constraint satisfaction problem like SEND + MORE = MONEY.

AIM:- Solve constraint satisfaction problem like SEND + MORE = MONEY

THEORY:-

1.) Constraint Satisfaction Method →
It is the process of finding a solution to set of constraints that impose some conditions that value must satisfy. The objective for constraint satisfaction problem is to assign value for each variable such that all constraints are satisfied.

2) Backtracking Search →
Algorithmic approach which used recursive approach to solve problem. It's a systematic way of trying out different sequences and until optimal is found. All constraints among variables are satisfied.

3) Constraint propagation:
Use constraint to reduce no. of legal value for variable which in turn can reduce legal values for another variable. It is an essential process of solving a constraint problem for constraint reasoning.

INPUT:- Initial values for some letters in given problem

OUTPUT:- Unique values for letters S, E, N, D, M, O, RE.

PAGE NO.:

ALGORITHM:- Constraint Satisfaction Method.

PLATFORM:- Windows

FAQ's

Q1) What are other constraint satisfaction problem?
Ans N-Queen, Map colouring, Crossword, Sudoku etc.

Q2) What do you mean by constraint propagation?
Ans It is the process that uses constraints to reduce no. of legal values for another variable.

Q3) Why backtracking can be used to solve constraint satisfaction problem?
In DFS nodes values for 1 variable at a time i.e. backtracks when variable has no values left to assign. Backtracking search keeps only single represents of a state & enters the same value rather than creating a new one.

AI - LAB3 CODE

```
import time
import itertools

def solution1():
    letters = ('s', 'e', 'n', 'd', 'm', 'o', 'r', 'y')
    digits = range(10)
    for perm in itertools.permutations(digits, len(letters)):
        sol = dict(zip(letters, perm))
        if sol['s'] == 0 or sol['m'] == 0:
            continue
        send = 1000 * sol['s'] + 100 * sol['e'] + 10 * sol['n'] + sol['d']
        more = 1000 * sol['m'] + 100 * sol['o'] + 10 * sol['r'] + sol['e']
        money = 10000 * sol['m'] + 1000 * sol['o'] + 100 * sol['n'] + 10 * sol['e']
+ sol['y']
        if send + more == money:
            print("SEND + MORE = MONEY")
            return send, more, money

def solution2():
    letters = ('c', 'r', 'o', 's', 'a', 'd', 'n', 'g', 'e')
    digits = range(10)
    for perm in itertools.permutations(digits, len(letters)):
        sol = dict(zip(letters, perm))
        if sol['c'] == 0 or sol['r'] == 0:
            continue
        cross = 10000 * sol['c'] + 1000 * sol['r'] + 100 * sol['o'] + 10 * sol['s']
+ sol['s']
        roads = 10000 * sol['r'] + 1000 * sol['o'] + 100 * sol['a'] + 10 * sol['d']
+ sol['s']
        danger = 100000 * sol['d'] + 10000 * sol['a'] + 1000 * sol['n'] + 100 *
sol['g'] + 10 * sol['e'] + sol['r']
        if cross + roads == danger:
            print("CROSS + ROADS = DANGER")
            return cross, roads, danger

print(solution1())
print(solution2())


'''
OUTPUT:-

SEND + MORE = MONEY
(9567, 1085, 10652)
CROSS + ROADS = DANGER
(96233, 62513, 158746)

'''
```