INDEX

NAME: Sulkit Shringi STD.: ____ SEC.: I I ROLL NO.: 596 SUB.: Artificial Intelligence

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	22/1/24	Implementation of toy problem.	10	
2	31/1/24	Implementation of real world problem	10	31/01/24
3		Implementation of constraint satisfaction problem		

7/2/24

Exc-03
Implementation of Constraint Satisfaction problem

- **Aim**: To solve Crypt Arithmetic Puzzle

- **Constraints**:
  - Assign an unique digit to each variable.
  - Carry is always 1.
  - Digits range [0-9]
  - Starting character cannot be 0.

Ques: HERE + SHE = COMES

```
  HERE
+ SHE
-------
 COMES
```

Assume C=1

with Carry	Without Carry
1+H = 0+10	H = 0+10
H = 0+9	H = 0+10
Take H = 9	Take H = 9
Then O=0	O = -1 not possible.

H - 9
E - 4
R - 5
S - 8
H - 9
E - 1
O - 0
M - 2

```
  HERE
+ SHE
-------
 COMES
```

- **Code**:

```
import itertools
import string
```

$E + S = M + 10$

$R + 9 = E$

$E + E = S$

$2E = S$ possible value of $E = 2, 3, 4$

Take $E = 2$

$S = 4$

$R + 9 = 2 + 10$ (with Carry)

$R = 2$

$E = R$. not possible

Take $E = 3$

$S = 6$

$R + 9 = 3 + 10$

$R = 4$

$E + S = M + 10$

$3 + 6 = M + 10$ Not Possible

Take $E = 4$

then, $S = 8$

$R + 9 = E + 10$

$R + 9 = 4 + 10$

$R = 5$

$E + S = M + 10$

$4 + 8 = M + 10$

$M = 2$

```
def correct_sol(p, puzzle):
    op1, op, op2, e, r = break_puzzle(puzzle.
                                      translate())
    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 .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_void_points(puzzle):
    digits = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    letters = get_unique_letters(puzzle)
    critical_indices = get_starting_letters
    (puzzle, letters)
    pass_perm []
    for perm in itertools.permutation(digits,
    len(letters)):
    flag = 0
    for i in critical_indices:
        if perm[i] == '0'
            flag = 1
            break;
```

Output :

{'E': '4', 'c': '1', 'S': '8', 'H': '9', 'D': 'M': '3', 'R': S

```
if flag == 0:
    pass_perms.append(perm)
return pass_perms

def solve(puzzle):
    letters = get_unique_letters(puzzle)

for pass in get_valid_permutations(puzzle):
    p = str.maketrans(''.join(letters), ''.join(pass))
if correct_vds(p, puzzle):
    answer = dict(zip(letters, pass))
    print(answer)
    solved_puzzle = puzzle
    for c in answer:
        x = solved_puzzle.replace(c, answer[c]).
    solved_puzzle = x
    print(solved_puzzle)

solve("HERE + SHE = COMES")
```

- Result : Crypt Arithmetic problem was successfully implemented.