Batch: B1	Roll No.: 16010121045
Experiment / assignment / tutorial No.	

Title: Implementation of Constraint Satisfaction concepts

Objective: Implementation of Local search algorithm

Expected Outcome of Experiment:

Course Outcome	After successful completion of the course students should be able to
CO2	Analyse and solve problems for goal based agent architecture (searching and planning algorithms).

Books/ Journals/ Websites referred:

1. **"Artificial Intelligence: a Modern Approach" by Russel and Norving, Pearson education Publications**
2. **"Artificial Intelligence" By Rich and knight, Tata Mcgraw Hill Publications**
3.

Pre Lab/ Prior Concepts: Informed, uninformed search, Local search

Historical Profile:

New Concepts to be learned: Constraint Satisfaction, CSP with backtracking

Definition:- (Define CSP)

The sequence in which variable-constraint assignments are considered by CSP algorithms to improve the backtracking efficiency:-

Problem chosen: Cryptarithmetic

DAYS + TOO = SHORT

Step by step solution to the problem:

Explanation:

```
   F R E E
+ B I R D
 ————————
  C A G E
```

Step 1:
Start with the rightmost column, E + D = E.
This implies D = 0
Step 2:
Moving to the left, E + R + carry (from Step 1) = G.
Since G can't be 0, the minimum value for E + R + carry is 10. So, E + R + carry = 10 + G.
Step 3:
Now, we have R + I + carry (from Step 2) = A.
This implies R + I + carry – A = 0
Step 4:
Moving further left, F + B + carry (from Step 3) = C.
This implies F + B + carry – C = 0.
Step 5:
Finally, we have a carry in the leftmost column.
Now, we want to find the values of F, R, B, I, C, A such that A + B + C + D + E + F + G equals 22.
Let's solve through an example:
Suppose we set F = 1 and R = 9. Then, from Step 2:
E + 9 + carry = 10 + G
E + G + carry = 1
Since no two letters can have the same value, we can't have E = G = 0. So, the carry must be 1.
Now, from Step 3:
9 + I + 1 – A = 0
I + 10 – A = 0
I – A = -10
This implies A must be 0 (to avoid negative values).
Now, from Step 4:
1 + B + 1 – C = 0
B + 2 – C = 0
B – C = -2
This implies C must be greater than B, and since B can't be 0 (as D is 0), C must be at least 3. Let's choose C = 3 and B = 1.

Now, we've found values for F, R, B, I, C, A such that A + B + C + D + E + F + G = 22:
F = 1, R = 9, B = 1, I = 0, C = 3, A = 0
Now, let's calculate E and G:
E + 9 + 1 = 10 + G
E + 10 = 10 + G
This implies E = G.
So, E = G = 5.
Now, we add up A + B + C + D + E + F + G:
0 + 1 + 3 + 0 + 5 + 1 + 5 = 15
Therefore, A + B + C + D + E + F + G = 22

Code:

```
import itertools

def get_value(word, substitution):
    s = 0
    factor = 1
    for letter in reversed(word):
        s += factor * substitution[letter]
        factor *= 10
    return s


def solve(equation):
    left, right = equation.lower().replace(' ',
'').split('=')
    left = left.split('+')
    letters = set(right)
    for word in left:
        for letter in word:
            letters.add(letter)
    letters = list(letters)


    digits = range(10)
    for perm in itertools.permutations(digits, len(letters)):
        sol = dict(zip(letters, perm))

        if sum(get_value(word, sol) for word in left) ==
get_value(right, sol):
```

```
            values_left = [str(get_value(word, sol)) for word
in left]
            value_right = get_value(right, sol)
            result = ' + '.join(values_left) + " = {}
(mapping: {})".format(value_right, sol)
            print(result)

if __name__ == '__main__':
    solve('D A Y S + T O O=S H O R T ')
```

```
❯ python3 -u "/Users/pargatsinghdhanjal/Documents/College/Sem6/AI/Programs/exp7.py"
9871 + 655 = 10526 (mapping: {'a': 8, 'r': 2, 'y': 7, 'o': 5, 't': 6, 's': 1, 'd': 9, 'h': 0})
```

Post Lab objective Questions:

To overcome the need to backtrack in constraint satisfaction problem can be eliminated by _____

a) Forward Searching

b) Constraint Propagation

c) Backtrack after a forward search

d) Omitting the constraints and focusing only on goals

Consider a problem of preparing a schedule for a class of student. What type of problem is this?

a) Search Problem

b) Backtrack Problem

c) CSP

d) Planning Problem

Q1. How do you solve a CSP Problem?

To solve a Constraint Satisfaction Problem (CSP), you typically use techniques such as:

- **Backtracking:** Enumerate possible assignments of values to variables and backtrack when a constraint violation is encountered.
- **Constraint Propagation:** Use local constraints to reduce the domain of variables, thus making it easier to find a solution.

- **Forward Checking:** After making an assignment, check the remaining domains of variables to see if any are reduced to empty, which would indicate a dead-end.
- **Arc Consistency:** Ensure that for every variable, there is a consistent assignment for every other variable with which it shares a constraint.
- **Heuristic Methods:** Use heuristics to guide the search process, such as variable ordering or value ordering heuristics.

Q2. Explain CSP with an example.

Let's consider a classic example of a CSP: the map coloring problem.

Problem Statement: Given a map of regions (such as countries) where adjacent regions cannot have the same color, assign a color to each region such that no adjacent regions share the same color.

Variables: Each region on the map is a variable.

Domains: The domain of each variable is the set of available colors.

Constraints: The constraint is that no two adjacent regions can have the same color.

Example:

In this example, suppose we have a map with four regions: A, B, C, and D. The constraints are that adjacent regions cannot have the same color. Let's say we have three colors available: red, blue, and green.

We could start by assigning colors to variables and applying constraint propagation and backtracking to find a valid solution. For instance, we might start by assigning red to region A, then check the adjacent regions to see which colors are available for them, and so on until a valid assignment for all regions is found. If at any point a region cannot be assigned a color without violating the constraints, we backtrack and try a different color for a previous region.