ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 3

CRYPTARITHMETIC PROBLEM

Harsh Goel
RA1811003010185
CSE-C1

Aim:

To solve Cryptarithmetic Problem and verify result and test cases

Problem Description:

Cryptarithmetic Problem is a type of constraint satisfaction problem where the game is about digits and its unique replacement either with alphabets or other symbols. In cryptarithmetic problem, the digits (0-9) get substituted by some possible alphabets or symbols. The task in cryptarithmetic problem is to substitute each digit with an alphabet to get the result arithmetically correct.

In the crypt-arithmetic problem, some letters are used to assign digits to it. Like ten different letters are holding digit values from 0 to 9 to perform arithmetic operations correctly. There are two words are given and another word is given an answer of addition for those two words.

Problem Formulation:

The goal here is to assign each letter a digit from 0 to 9 so that the arithmetic works out correctly. The rules are that all occurrences of a letter must be assigned the same digit, and no digit can be assigned to more than one letter.

As an example, we can say that two words 'BASE' and 'BALL', and the result is 'GAMES'. Now if we try to add BASE and BALL by their symbolic digits, we will get the answer GAMES.

The rules or constraints on a cryptarithmetic problem are as follows:

- There should be a unique digit to be replaced with a unique alphabet.
- The result should satisfy the predefined arithmetic rules, i.e., 2+2 =4, nothing else.
- Digits should be from 0-9 only.
- There should be only one carry forward, while performing the addition operation on a problem.
- The problem can be solved from both sides, i.e., lefthand side (L.H.S), or righthand side (R.H.S)

```
Input:
This algorithm will take three words.
            B A S E
            B A L L
          ----------
          G A M E S

Output:
It will show which letter holds which number from 0 – 9.
For this case it is like this.
```

Letter	A	B	E	G	L	M	S
Values	4	2	1	0	5	9	6

```
            B A S E                     2 4 6 1
            B A L L                     2 4 5 5
          ---------                   ---------
          G A M E S                   0 4 9 1 6
```

Algorithm:

1. First, create a list of all the characters that need assigning to pass to the function
2. If all characters are assigned, return true if puzzle is solved, false otherwise
3. Otherwise, consider the first unassigned character
4. for (every possible choice among the digits not in use)
5. make that choice and then recursively try to assign the rest of the characters
6. if recursion successful, return true
7. if !successful, unmake assignment and try another digit

Source Code:

Language-**PYTHON**

```
from itertools import chain, permutations
from string import digits


def solve(arr, result):
    letters = ''.join(set(chain(result, *arr))) #collect all letter (uni
que) by iterating
    initial_letters = ''.join(set(chain(result[0], (a[0] for a in arr)))
)
```

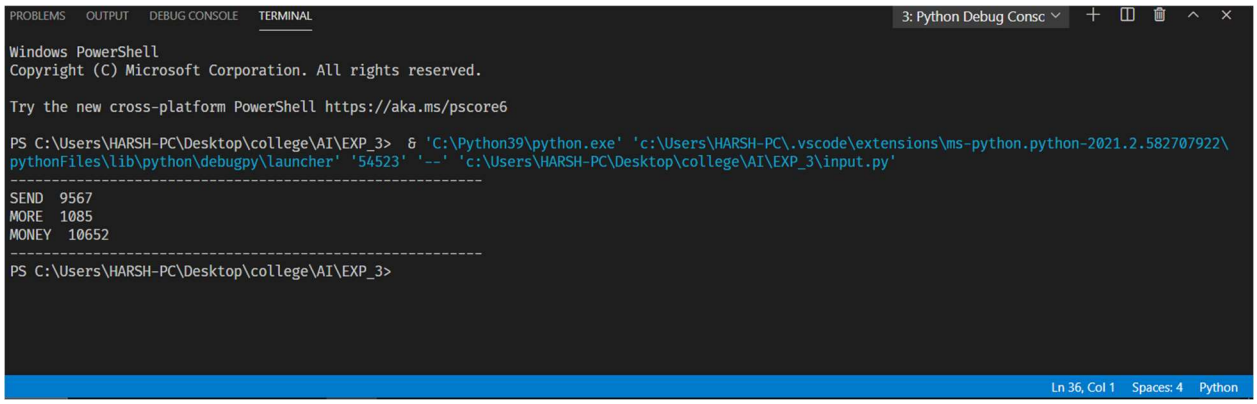
```
    for perm in permutations(digits, len(letters)): # all permutations l
oop
        decipher_table = str.maketrans(letters, ''.join(perm))  #perm =>
 numbers perm
        def decipher(s):
            return s.translate(decipher_table)
        if '0' in decipher(initial_letters):
            continue # as leading zeros not allowed so just skip all
        deciphered_sum = sum(int(decipher(code)) for code in arr) # add
both codes
        if deciphered_sum == int(decipher(result)): #check if sum is equ
al to sum result
            print("-----------------------------------------------------
----")
            print(str(arr[0]) + "  " + str(decipher(arr[0])))
            print(str(arr[1]) + "  " + str(decipher(arr[1])))
            print(str(result) + "  " + str(decipher(result)))
            print("-----------------------------------------------------
----")
            break
    else:
        print(" + ".join(arr), "=", result, " : no solution")

solve(['SEND', 'MORE'], 'MONEY')
# solve(['TWO', 'TWO'], 'FOUR')
# solve(['BASE', 'BALL'], 'GAMES')
```

TEST CASE:

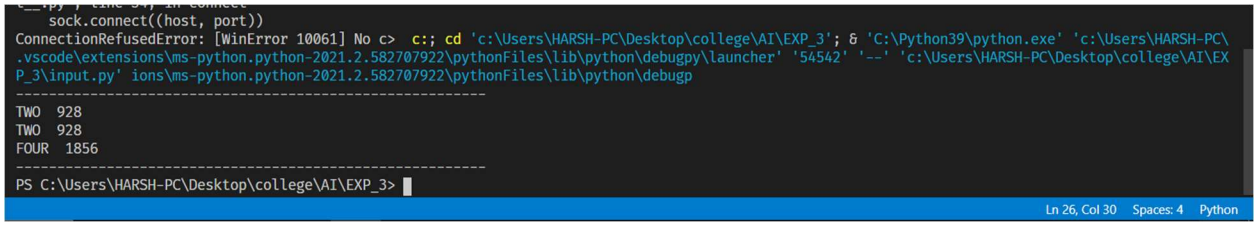
Case 1:

Input : SEND MORE MONEY

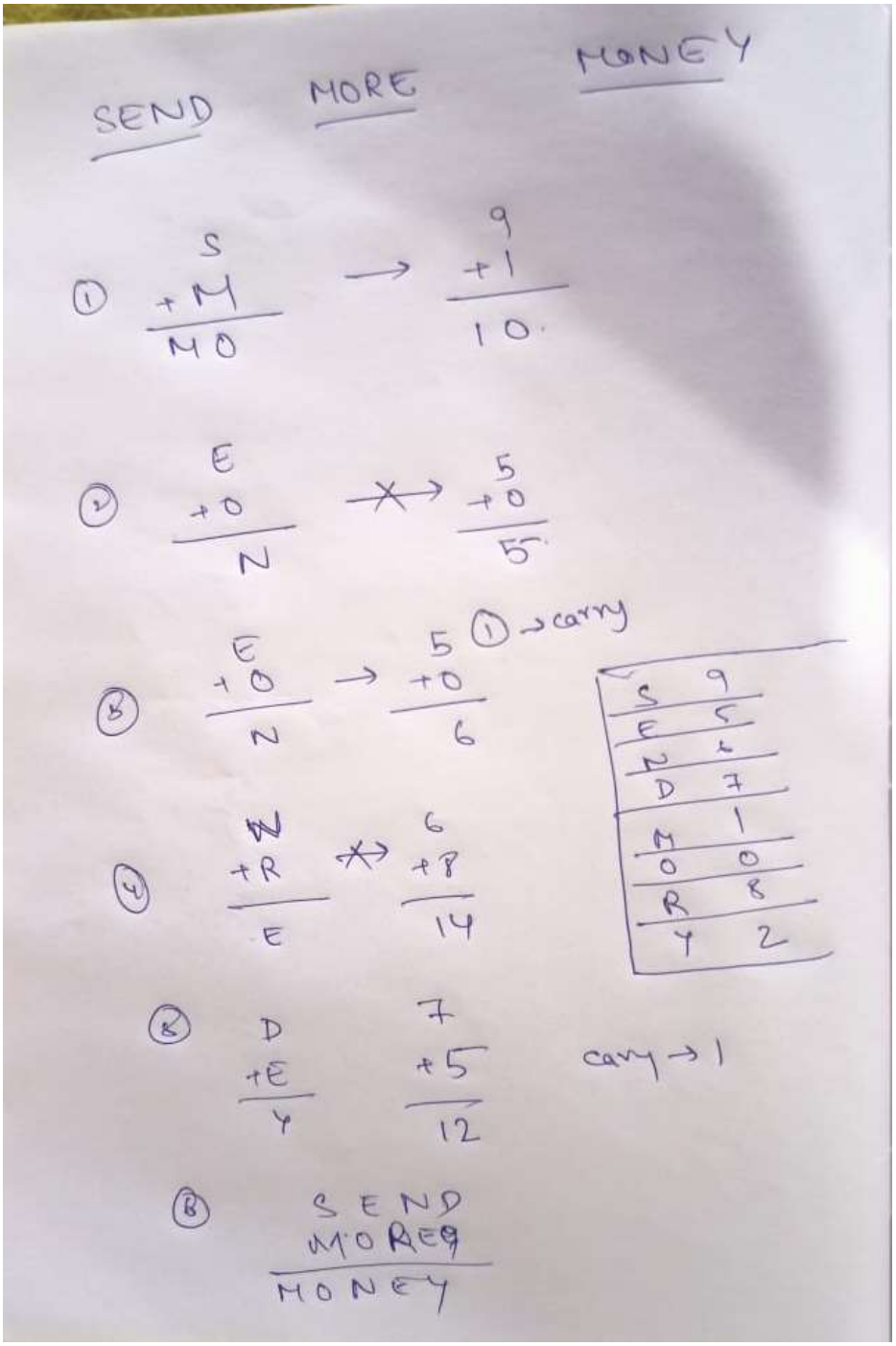


Case 2: TWO TWO FOUR

Input :



Verification:



Result: We have successfully solved the Cryptarithmetic Puzzle Problem in Python and verified the output and test cases.