**EXERCISE-3**

# Constraint Satisfaction Problem-Crypt arithmetic puzzle

**NAME:** A.HariChandana

**REG NO:** RA1911026010009

**SEC:** K1

**AIM :**

To develop a CSP Problem i.e., CryptArithmetic Problem

**CONSTRAINT SATISFACTION PROBLEM :**

CSP is a class of problems that may be represented in terms of variables (a, b, …), domains (a in [1, 2, 3], …), and constraints (a < b, …).

**PROBLEM STATEMENT:**

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

**ALGORITHM/PROCEDURE:**

**1.BASE + BALL = GAMES**

1. Assuming numbers can't start with 0, G is 1 because two four-digit numbers can't sum to 20000 or more. SE+LL=ES or 1ES.

2. If it is ES, then LL must be a multiple of 9 because SE and ES are always congruent mod 9. But LL is a multiple of 11, so it would have to be 99, which is impossible.

3. So SE+LL=1ES. LL must be congruent to 100 mod 9. The only multiple of 11 that work is 55, so L is 5. SE+55=1ES. This is possible when E+5=S.

4. The possibilities for ES are 27, 38, or 49. BA+BA+1=1AM. B must be at least 5 because B+B (possibly +1 from a carry) is at least 10. If A is less than 5, then A+A+1 does not carry, a and A must be even. Inversely, if A is greater than 5, it must be odd.

5. The possibilities for A are 0, 2, 4, 7, or 9. 0 does not work because M would have to be 1. 2 and 7 don't work because M would have to be 5. 9 doesn't work because M would also have to be 9. So A is 4, M is 9, and B is 7. This leaves 38 as the only possibility for ES.

6.The full equation is: 7483 + 7455 14938

**2.SEND + MORE = MONEY**

5 4 3 2 1

S E N D

+ M O R E

c3 c2 c1

----------------------

M O N E Y

1. From Column 5, M=1, since it is only carry-over possible from sum of 2 single digit number in column 4.

2. To produce a carry from column 4 to column 5 'S + M' is at least 9 so 'S=8or9' so 'S+M=9or10' & so 'O = 0 or 1'. But 'M=1', so 'O = 0'.

3. If there is c+rry from Column 3 to 4 then 'E=9' & so 'N=0'. But 'O = 0' so there is no carry & 'S=9' & 'c3=0'.

4. If there is no carry from column 2 to 3 then 'E=N' which is impossible, therefore there is carry & 'N=E+1' & 'c2=1'.

5. If there is carry from column 1 to 2 then 'N+R=E mod 10' & 'N=E+1' so 'E+1+R=E mod 10', so 'R=9' but 'S=9', so there must be c+rry from column 1 to 2. Therefore 'c1=1' & 'R=8'. 14

6. To produce carry 'c1=1' from column 1 to 2, we must h+ve 'D+E=10+Y' as Y cannot be 0/1 so D+E is at least 12. As D is at most 7 & E is At least 5 (D cannot be 8 or 9 as it is already assigned). N is at most 7 & 'N=E+1' so 'E=5or6'.

7. If E were 6 & D+E at least 12 then D would be 7, but 'N=E+1' & N would also be 7 which is impossible. Therefore 'E=5' & 'N=6'.

8. D+E is at least 12 for that we get 'D=7' & 'Y=2'.

**SOLUTION:**

9 5 6 7

+ 1 0 8 5

-----------------

1 0 6 5 2

**VALUES:**

S=9

E=5

N=6

D=7

M=1

O=0

R=8

Y=2

**CODE:**

import string

import itertools

inListNumsAsStringArray = [ ['BASE', 'BALL'],

['SEND', 'MORE'] ]

inResultsArray = [ 'GAMES',

'MONEY' ]

inPossibleNumsAsStr = '0123456789'

# Input: (string, dictionary)

# string 'AB'

# key-value pairs for characters and numbers. Ex: dictCharAndDigit = {'A': '1', 'B': '2'}

# Output:(Number)

# 12

def getNumberFromStringAndMappingInfo(inStr, inDictMapping):

numAsStr = ''

for ch in inStr:

numAsStr = numAsStr + inDictMapping[ch]

return int(numAsStr)

def solveCryptarithmeticBruteForce(inListNumsAsString, inResultStr, inPossibleNumsAsStr):

nonZeroLetters = []

strFromStrList = ''

for numStr in inListNumsAsString:

nonZeroLetters.append(numStr[0])

strFromStrList = strFromStrList + numStr

nonZeroLetters.append(inResultStr[0])

strFromStrList = strFromStrList + inResultStr

uniqueStrs = ''.join(set(strFromStrList))

for tup in itertools.permutations(inPossibleNumsAsStr, len(uniqueStrs)):

dictCharAndDigit = {}

for i in range(len(uniqueStrs)):

dictCharAndDigit[uniqueStrs[i]] = tup[i]

nonZeroLetterIsZero = False

for letter in nonZeroLetters:

if(dictCharAndDigit[letter] == '0'):

nonZeroLetterIsZero = True

break

if(nonZeroLetterIsZero == True):

continue

result = getNumberFromStringAndMappingInfo(inResultStr, dictCharAndDigit)

testResult = 0

for numStr in inListNumsAsString:

testResult = testResult + getNumberFromStringAndMappingInfo(numStr, dictCharAndDigit)

if(testResult == result):

strToPrint = ''

for numStr in inListNumsAsString:

strToPrint = strToPrint + numStr + '(' + str(getNumberFromStringAndMappingInfo(numStr, dictCharAndDigit)) + ')' + ' + '

strToPrint = strToPrint[:-3]

strToPrint = strToPrint + ' = ' + inResultStr + '(' + str(result) + ')'

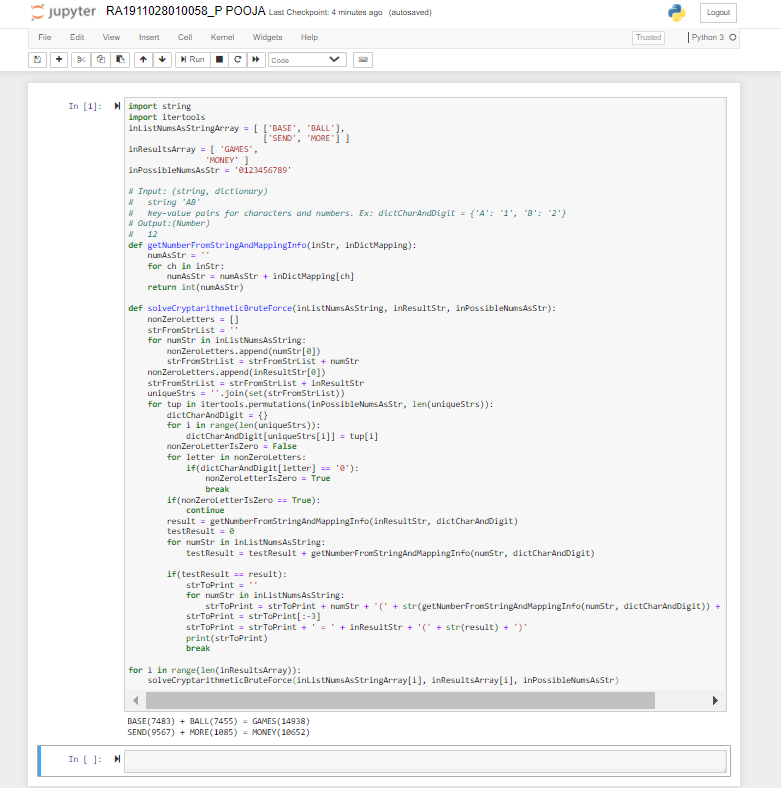
print(strToPrint)

break

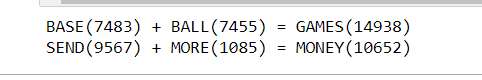
for i in range(len(inResultsArray)):

solveCryptarithmeticBruteForce(inListNumsAsStringArray[i], inResultsArray[i], inPossibleNumsAsStr)

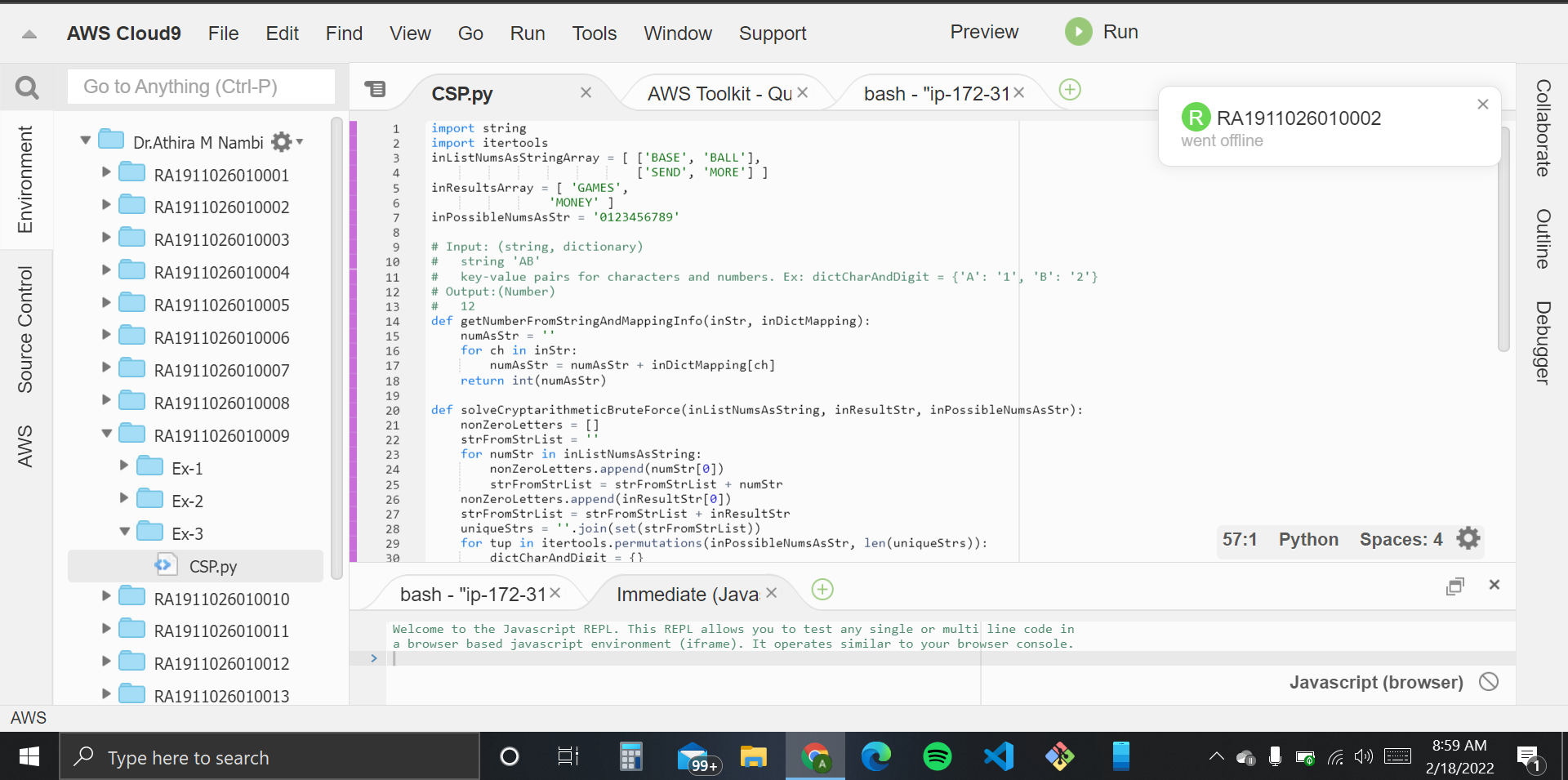
**SCREENSHOT:**

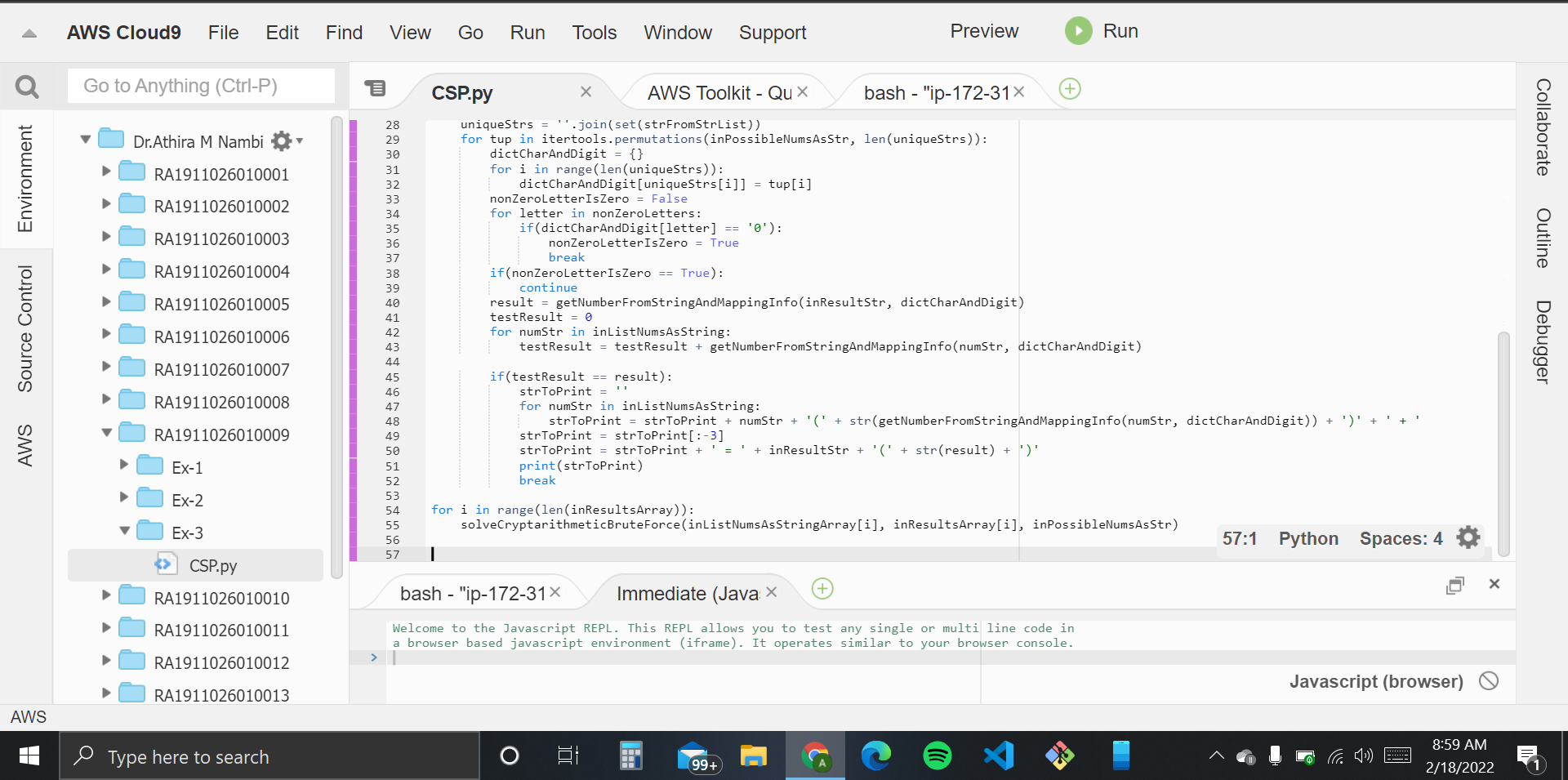


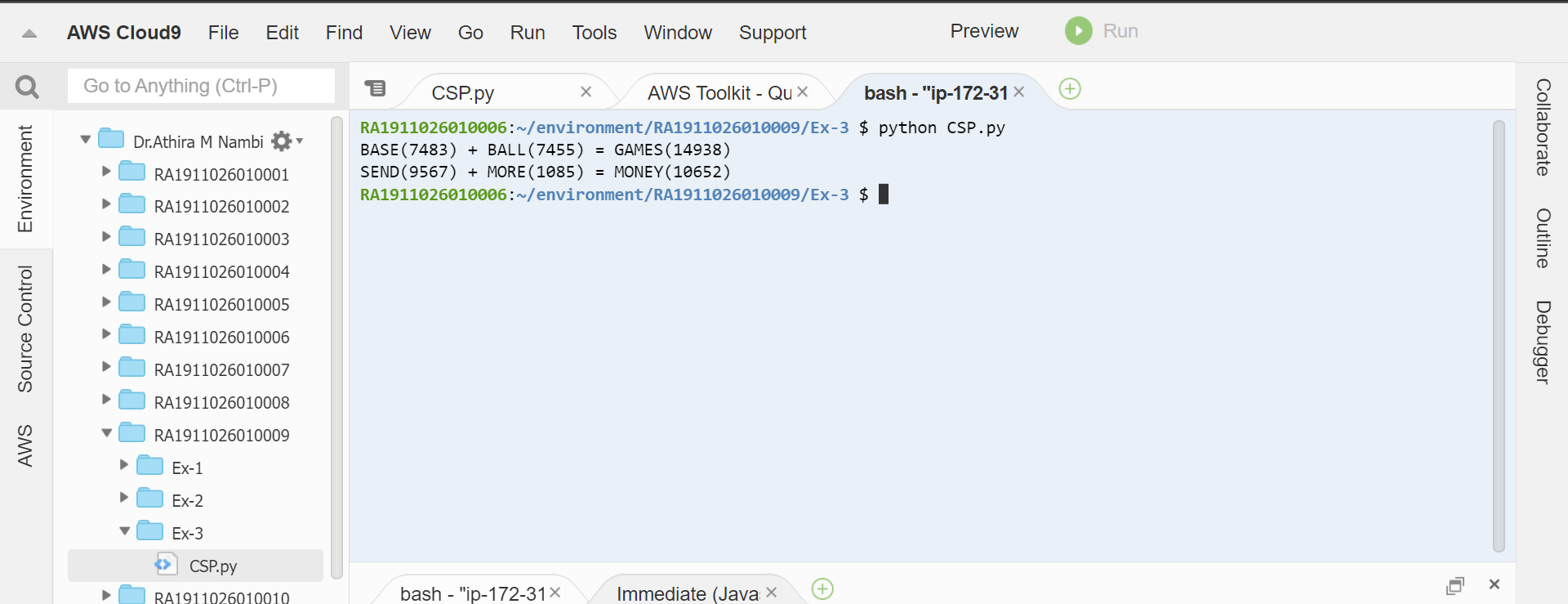
**OUTPUT:**

****

**SCREENSHOT OF CODE AND OUTPUT IN AWS ENVIRONMENT:**

****

****

****

**RESULT:** The constraint satisfying problem BASE + BALL = GAMES and SEND + MORE = MONEY solved using the carry over technique and values for the alphabets obtained successfully.