Name – Pranav Salunkhe

TY CS-D

Batch-1

Roll no – 09

Assignment 4

Code –

import java.util.Arrays;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Map;

import java.util.Set;

class CSP {

    private String term1, term2, term3;

    private Set<Character> allUniqueLetters;

    private Set<Character> uniqueLetters;

    private Map<Character, Integer> assignment;

    CSP(String term1, String term2, String term3) {

        this.term1 = term1;

        this.term2 = term2;

        this.term3 = term3;

        allUniqueLetters = new HashSet<>();

        assignment = new HashMap<>();

        for (char c : (term1 + term2 + term3).toCharArray()) {

            if (Character.isAlphabetic(c)) {

                allUniqueLetters.add(c);

            }

        }

        for (char a : allUniqueLetters) {

            System.out.println(a);

        }

        uniqueLetters = new HashSet<>(allUniqueLetters);

    }

    public void addConstraint(Character c, Integer val) {

        assignment.put(c, val);

        uniqueLetters.remove(c);

    }

    public void solveCryptarithmetic() {

        int[] domain = new int[10];

        Arrays.fill(domain, -1);

        if (backtrack(uniqueLetters, assignment, domain, term1, term2, term3, 0)) {

            System.out.println("Solution found:");

            for (char letter : allUniqueLetters) {

                System.out.println(letter + " = " + assignment.get(letter));

            }

        } else {

            System.out.println("No solution found.");

        }

    }

    private boolean backtrack(Set<Character> uniqueLetters, Map<Character, Integer> assignment, int[] domain,

            String term1, String term2, String term3, int currentIndex) {

        if (currentIndex == uniqueLetters.size()) {

            int value1 = evaluateTerm(term1, assignment);

            int value2 = evaluateTerm(term2, assignment);

            int value3 = evaluateTerm(term3, assignment);

            return value1 + value2 == value3;

        }

        char currentLetter = (char) uniqueLetters.toArray()[currentIndex];

        for (int digit = 0; digit <= 9; digit++) {

            if (!assignment.containsValue(digit)) {

                assignment.put(currentLetter, digit);

                if (backtrack(uniqueLetters, assignment, domain, term1, term2, term3, currentIndex + 1)) {

                    return true;

                }

                assignment.remove(currentLetter);

            }

        }

        return false;

    }

    private static int evaluateTerm(String term, Map<Character, Integer> assignment) {

        int value = 1;

        char[] terms = term.toCharArray();

        int product = assignment.get(terms[terms.length - 1]);

        for (int i = terms.length - 2; i >= 0; i--) {

            product = product + (int) (Math.pow(10, value++) \* assignment.get(terms[i]));

        }

        return product;

    }

}

public class CryptarithmeticCSP {

    public static void main(String[] args) {

        // String term1 = "TWO";

        // String term2 = "TWO";

        // String term3 = "FOUR";

        // String term1 = "THIS";

        // String term2 = "IS";

        // String term3 = "HERE";

        // String term1 = "HERE";

        // String term2 = "SHE";

        // String term3 = "COMES";

        // String term1 = "SEND";

        // String term2 = "MORE";

        // String term3 = "MONEY";

        // String term1 = "CROSS";

        // String term2 = "ROADS";

        // String term3 = "DANGER";

        // String term1 = "SEND";

        // String term2 = "MORE";

        // String term3 = "MONEY";

        // String term1 = "POINT";

        // String term2 = "ZERO";

        // String term3 = "ENERGY";

        String term1 = "YOUR";

        String term2 = "YOU";

        String term3 = "HEART";

        String[] str = {"reflower","flow","flight"};

        Arrays.sort(str);

        for(String s: str){

            System.out.println(s);

        }

        CSP problem1 = new CSP(term1, term2, term3);

        problem1.addConstraint('Y', 9);

        problem1.solveCryptarithmetic();

    }

}

Output

