Assignment No: 9

**Title:**

Constraint Satisfaction Problem: Implement crypt-arithmetic problem.

**Theory:**

Crypt arithmetic Problem is a type of [constraint satisfaction problem](https://www.tutorialandexample.com/constraint-satisfaction-problems-in-artificial-intelligence/) where the game is about digits and its unique replacement either with alphabets or other symbols. In **crypt arithmetic problem,** the digits (0-9) get substituted by some possible alphabets or symbols. The task in crypt arithmetic problem is to substitute each digit with an alphabet to get the result arithmetically correct.

**The rules or constraints on a crypt arithmetic 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)**

**Program:**

import java.sql.Array;

import java.sql.Time;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

/\*\*

\* AI Experiment 4

\* Cryptarithmetic.java

\* Purpose: Solve Cryptarithmetic Problems Using Permutations

\*

\* @author Javed Khatri

\* @version 1.0 09/08/2014

\*/

public class CryptArithmetic

{

static Scanner sc = new Scanner(System.in);

static ArrayList<Character> uniquechar = new ArrayList<Character>();

static int nos[] = {0,1,2,3,4,5,6,7,8,9};

static HashMap<Character, Integer> hm = new HashMap<Character, Integer>();

static int no1,no2,no3,count=0;

static boolean solutionfound=false;

static ArrayList<ArrayList<Integer>> permuts = new ArrayList<ArrayList<Integer>>();

static String s1,s2,s3;

public static void main(String[] args) throws InterruptedException

{

getInput();

System.out.println("Calculating. Please wait...");

long start = System.currentTimeMillis();

calculate();

long end = System.currentTimeMillis();

double time = (end-start)/1000.0;

System.out.println("Time required for execution: "+time+" seconds");

}

public static void getInput()

{

System.out.println("Enter string 1:");

s1 = sc.nextLine();

System.out.println("Enter string 2:");

s2 = sc.nextLine();

System.out.println("Enter output string :");

s3 = sc.nextLine();

addToArrayList(s1);

addToArrayList(s2);

addToArrayList(s3);

}

public static void calculate()

{

Collections.sort(uniquechar);

permute(nos, 0);

for(int i=0;i<permuts.size();i++)

{

for(int j=0;j<uniquechar.size();j++)

{

hm.put(uniquechar.get(j),permuts.get(i).get(j));

}

no1 = getNumber(s1);

no2 = getNumber(s2);

no3 = getNumber(s3);

if(no3==no1+no2 && getLengthOfInt(no1)==s1.length() && getLengthOfInt(no2)==s2.length() && getLengthOfInt(no3)==s3.length() && count<1)

{

solutionfound=true;

System.out.println(s1+":"+no1+" \n" +s2+":"+no2+" \n"+s3+":"+no3);

count++;

}

}

if(!solutionfound)

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

}

public static void permute(int []a,int k )

{

if(k==a.length)

{

ArrayList<Integer> perm = new ArrayList<Integer>();

for(int i=0;i<a.length;i++)

{

perm.add(a[i]);

}

permuts.add(perm);

}

else

{

for (int i = k; i < a.length; i++)

{

int temp=a[k];

a[k]=a[i];

a[i]=temp;

permute(a,k+1);

temp=a[k];

a[k]=a[i];

a[i]=temp;

}

}

}

public static boolean found(char c)

{

boolean flag=false;

for(int i=0;i<uniquechar.size();i++)

{

if(uniquechar.get(i)==c)

flag=true;

}

if(flag)

return true;

else

return false;

}

public static void addToArrayList(String s)

{

for(int i=0;i<s.length();i++)

{

if(!found(s.charAt(i)))

{

uniquechar.add(s.charAt(i));

}

}

}

public static void iterateHashMap()

{

for (Map.Entry<Character, Integer> entry : hm.entrySet())

{

char key = entry.getKey();

int value = entry.getValue();

System.out.println("\n Key:"+key+" Value:"+value);

}

}

public static int getNumber(String s)

{

String temp="";

for(int i=0;i<s.length();i++)

{

temp=temp+hm.get(s.charAt(i));

}

return Integer.parseInt(temp);

}

public static int getLengthOfInt(int n)

{

return String.valueOf(n).length();

}

}

**Output:**

