Given a string s, perform the following algorithm:

1. Replace every letter with its 1-based position in the alphabet;
2. Divide the obtained thus number by two, and round the result down to the nearest integer;
3. Replace every digit in the obtained number with the letter at the corresponding 0-based position in the alphabet;
4. Return the resulting string.

**Example**

For s = "abcde", the output should be  
stringDivision(s) = "gbhc"

Here're the results obtained at each step of the algorithm:

1. "abcde" is changed to 12345;
2. 12345 becomes 12345 / 2 = 6172;
3. 6172 turns into "gbhc";
4. "gbhc" is returned.

**Input/Output**

* **[time limit] 3000ms (java)**
* **[input] string s**

A string of lowercase Latin letters.

*Constraints:*  
1 ≤ s.length ≤ 35.

* **[output] string**

Result of the above described algorithm.

<https://codefights.com/challenge/ifSxNwEhkPER6hExe/main>

*/\**

*\* To change this template, choose Tools | Templates*

*\* and open the template in the editor.*

*\*/*

package javaapplication150;

import java.math.BigInteger;

*/\*\**

*\**

*\* @author Administrador*

*\*/*

public class JavaApplication150 {

*/\*\**

*\* @param args the command line arguments*

*\*/*

    static String stringDivision(String s) {

        String num = "";

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

        {

            num += (s.charAt(i) - 'a' + 1);

        }

        //int b = Integer.parseInt(num) / 2;

        BigInteger b = new BigInteger(num);

        b = b.divide(BigInteger.valueOf(2));

        String res = "";

        String bString = String.valueOf(b);

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

        {

            res += (char)(bString.charAt(i) + '0' +1);

        }

        return res;

    }

    public static void main(String[] args) {

        // TODO code application logic here

        String s = "friendshipismagic";

        System.out.println(stringDivision(s));

    }

}