INSOMNIA:

1	5
2	10
3	15
4	20
5	25
6	30
7	35

1. **Smart Messenger**: Cellular phone or online chats are very popular these days. While talking to our friends we use different set of spellings to communicate. For example if I want to send “I am fine” we generally write “m fn”. Being a insomniac you found that peple generally remove vowels from the sentences. Your task is to write a program called smart Messenger that take a sentence and return its smart sentence version.

   Example:

   1)

   “I am Insomniac”

   returns:

   “m nsmnc”

   2)

   “How are you ?”

   Returns:

   “hw r yu ?”

   3) “TechVibes”

   Returns:

   “TchVbs”

2. **Smart Finder:** You are already familiar with Windows search option. If you want to search for files of pdf you type "*.pdf" Similarly if you want to search for "a" file starting with "a" and ending with ".doc" You write "a*.doc". Your task here is to write a program having method equivalent(String1, String2). Where String1 would be actual string like "abc.doc, xyz.pdf, notepad etc" while string 2 will be pattern like *pdf, x*yz.

   Example :

   1)

   Equivalent("abc.txt", "a*t")

   Returns : true; // since abc.txt has patter a*t

   2) Equivalent("xyzwxyzwabcwabc","xyz*abc*abc")

   Returns true:

3. **Smart Matcher:** We know in computer the values are stored in the form of 0s and 1s. And there are instructions available to rotate them either left or right in cyclic order. For example left cyclic shift of "000111" will be "001110". Your task here is to write a program that takes two strings (strings contain on 1s and 0s) and you have to check if second string can be obtained by the first.

Example :

1)

("000111", "111000")

Returns : true;

2) ("01010101","10101010")

Returns true

3) ("1001","1010")

Returns false;

4. **Smart Insomaniac:** Lets test your logic skill. You have worked on various pattern printing program like print a hill of *s. Below is the given pattern. Lets extract the logic and write down the program for that.

```
*

**

***

****

*****

********

*******

***************

*********

*******************************

***********
```

5. **Smart Rectange:** A smart rectangle is a rectangle divided into a grid of unit squares. Each square contains a number, and the numbers in neighboring squares always satisfy the following property:
   The number in any square S that is neither in the bottom row nor in the right column can be computed as the sum of the following three numbers:
   - The number in the square directly below S.
   - The number in the square directly to the right of S.
   - The number in the other square adjacent to the previous two squares (the one diagonally down and to the right of S).

An example of a correctly filled smart rectangle:

```
+----+----+----+----+----+
| 88 | 57 | 33 | 10 |  5 |
+----+----+----+----+----+
| 18 | 13 | 11 | 12 | -7 |
+----+----+----+----+----+
|  1 |  4 | -2 |  1 | 18 |
+----+----+----+----+----+
```

For example, in the top left corner we have 88 = 18 + 57 + 13.

We have a secret smart rectangle. You will be given a int[] leftColumn containing the leftmost number in each row of our rectangle, from the top to the bottom. You will also be given an int[] topRow containing the topmost number in each column of our rectangle, from the left to the right. Compute and return the number in the bottom right corner. If the input is such that this number cannot be determined uniquely, return 0 instead.

Examples

1)

{88,18,1}

{88,57,33,10,5}

Returns: 18

This is the rectangle from the problem statement. The lower right corner is uniquely determined by the left column and the top row.

2)

{0,0,0,0}

{0,0,0,0}

Returns: 0

The only correct way to fill this rectangle is to place a zero into each square.

3)

{6,1}

{6,2}

Returns: 3

This is the smallest non-trivial case:

6. **Smart Number:** Let us consider a function super(X) defined as follows. For a given non-full set S of digit Supersum of X is simply the sum of sub-numbers for all valid sets S.

For example, if X is 123, then super(X) = 123 + 12 + 13 + 23 + 1 + 2 + 3 = 177.

Given String X, find its super(X) and return remainder with 9.

Examples

1)

"123"

Returns: 6

Example from the problem statement. super(123) = 177, which gives remainder 6 after division by 9.

2)

"24816"

Returns: 3

Supersum of 24816 is 43986.

3)

"8"

Returns: 8

4)

"11235813213455"

Returns: 7

Supersum is 43950094900477.

7. **Smart DNA:** A DNA molecule is formed by two nucleotide chains of equal length. Each nucleotide in the first chain must form a bond with the nucleotide at the same position in the second chain. There are four types of nucleotides: A, C, G and T. Each type of nucleotide can only form a bond with one other type: A can only bond with T, and G can only bond with C. These pairs, AT and CG, are called complementary pairs. No other bonds are allowed.

You are given a String nucleotides, where each character is an available nucleotide. Return the length of the longest DNA molecule that can be created using only the available nucleotides. The length of a DNA molecule is the number of nucleotides in either one of its chains. If no DNA molecule can be created, return 0.

1)

"AGGCA"

Returns: 1

The set of nucleotides contains only one complementary pair CG.

2)

"GGTACAGTTT"

Returns: 3

The set of nucleotides contains one CG pair and two AT pairs.

3)

"ACCACCAACCA"

Returns: 0

The set of nucleotides contains only A and C nucleotides, which aren't complementary.

4)

"AAAAAAAAAAAAAAAAACCCCCCCCGGGGGGGGTTTTTTTTTTTTTTTT"

Returns: 25