

ACM International Collegiate Programming Contest 2001–2002

Sponsored by IBM

Southwestern European Regional Contest

<http://swerc.up.pt/>

Practice Set



University of Porto, Portugal

November 17th, 2001

This problem set should contain two (2) problems on six (6) numbered pages.
If something is missing from your problem set, please inform a runner immediately.

ACM International Collegiate Programming Contest 2001–2002

Southwestern European Regional Contest

University of Porto, Portugal
November 17th, 2001

Contents

Problem A: The Six-Letter Cipher	3
Problem B: Cantor Fractions II	5

Problem A

The Six-Letter Cipher

The Six-Letter Cipher is a method of encoding a secret message that involves both substitution and transposition. The encryption starts by randomly filling a 6×6 grid with the alphabet letters from A to Z and the digits from 0 to 9 (36 symbols in total). This grid must be known to both the sender and receiver of the message. The rows and columns of the grid are labeled with the letters A, B, C, D, E, F, as in the following example:

	A	B	C	D	E	F
A	8	P	3	D	1	N
B	L	T	4	O	A	H
C	7	K	B	C	5	Z
D	J	U	6	W	G	M
E	X	S	V	I	R	2
F	9	E	Y	0	F	Q

The first stage of encryption is to take each letter of the message (ignoring spaces and punctuation signs), locate it in the grid and replace it with the row/column letters that label its' position. For the example, '8' is replaced by 'AA' and 'E' is replaced by 'FB'.

Message: M E E T I N G A T 9 P M
 Stage 1 ciphertext: DF FB FB BB ED AF DE BE BB FA AB DF

The second stage involves a secret keyword, also known to both sender and receiver, for example "MARK". The letters of this keyword are written on top of a fresh grid. Next, the text from the first stage is written underneath it, in row-wise order. Then we re-arrange the columns of the grid so that the letters of the keyword are in alphabetical order. The final ciphertext is obtained by reading the sorted grid column-wise from left to right.

<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><th>M</th><th>A</th><th>R</th><th>K</th></tr> <tr><td>D</td><td>F</td><td>F</td><td>B</td></tr> <tr><td>F</td><td>B</td><td>B</td><td>B</td></tr> <tr><td>E</td><td>D</td><td>A</td><td>F</td></tr> <tr><td>D</td><td>E</td><td>B</td><td>E</td></tr> <tr><td>B</td><td>B</td><td>F</td><td>A</td></tr> <tr><td>A</td><td>B</td><td>D</td><td>F</td></tr> </table>	M	A	R	K	D	F	F	B	F	B	B	B	E	D	A	F	D	E	B	E	B	B	F	A	A	B	D	F	sort <u>columns</u> \rightarrow	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr><th>A</th><th>K</th><th>M</th><th>R</th></tr> <tr><td>F</td><td>B</td><td>D</td><td>F</td></tr> <tr><td>B</td><td>B</td><td>F</td><td>B</td></tr> <tr><td>D</td><td>F</td><td>E</td><td>A</td></tr> <tr><td>E</td><td>E</td><td>D</td><td>B</td></tr> <tr><td>B</td><td>A</td><td>B</td><td>F</td></tr> <tr><td>B</td><td>F</td><td>A</td><td>D</td></tr> </table>	A	K	M	R	F	B	D	F	B	B	F	B	D	F	E	A	E	E	D	B	B	A	B	F	B	F	A	D
M	A	R	K																																																							
D	F	F	B																																																							
F	B	B	B																																																							
E	D	A	F																																																							
D	E	B	E																																																							
B	B	F	A																																																							
A	B	D	F																																																							
A	K	M	R																																																							
F	B	D	F																																																							
B	B	F	B																																																							
D	F	E	A																																																							
E	E	D	B																																																							
B	A	B	F																																																							
B	F	A	D																																																							

Final ciphertext: FBDEBBBBFEAFDFEDBAFBABFD

The final ciphertext contains only six different letters A, B, C, D, E, F (the grid labels), hence the name for this cipher.

Notice that in order to avoid ambiguity in the second stage transposition, the keyword should have no letters repeated. Also, the keyword length must divide the double of the message length so that we fill the grid completely.

Problem

Write a program that reads the grid, the keyword and a plaintext message and outputs the ciphertext.

Input specification

The input consists of the grid, keyword and message text, separated by newlines. The grid consists of six lines of six alphanumeric upper-case characters each. The keyword consists of upper-case alphanumeric characters without repetitions. The message text consists of upper-case alphanumeric characters.

You can assume that the keyword contains no repetitions and is of an acceptable length for the encryption process. You can also assume that the message text is at most 1000 characters length and the keyword is at most 80 characters length.

Output specification

The output should be the characters for the ciphertext, ended by a newline.

Sample input

```
8P3D1N
LT40AH
7KBC5Z
JU6WGM
XSVIR2
9EYOFQ
MARK
MEETINGAT9PM
```

Sample output

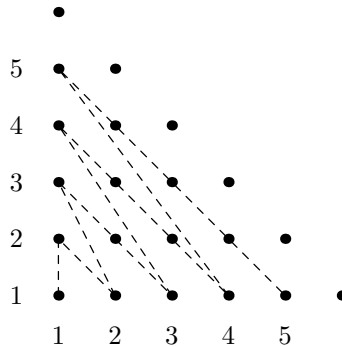
```
FBDEBBBBFEAFDFEDBAFBABFD
```

Problem B

Cantor Fractions II

In the late XIXth century the German mathematician George Cantor argued that the set of positive fractions \mathbb{Q}^+ is equipotent to the set of positive integers \mathbb{N} , meaning that they are both infinite, but of the same class. To justify this, he exhibited a one-to-one mapping from \mathbb{N} to \mathbb{Q}^+ .

To exhibit such a mapping, consider a *traversal* of the $\mathbb{N} \times \mathbb{N}$ plane that covers all the pairs:



The first pairs in this traversal are:

$$(1, 1), (2, 1), (1, 2), (3, 1), (2, 2), (1, 3), \dots$$

It is clear that this traversal will generate all pairs $(p, q) \in \mathbb{N} \times \mathbb{N}$; however, it will also generate several pairs that correspond to the same fraction p/q (for example: the first and fifth pairs in the example above). To obtain a one-to-one mapping it suffices to skip pairs corresponding to fractions that appeared previously. Thus our one-to-one mapping would be:

$$\underbrace{1/1}_{1\text{st}}, \underbrace{2/1}_{2\text{nd}}, \underbrace{1/2}_{3\text{rd}}, \underbrace{3/1}_{4\text{th}}, \underbrace{\cancel{2/2}}_{\text{skipped}}, \underbrace{1/3}_{5\text{th}}, \dots$$

Problem

Write a program that finds the i -th Cantor fraction following the one-to-one mapping outlined above.

Input specification

The inputs consists of a positive integer number i , with $1 \leq i \leq 100\,000$.

Output specification

The output consists of the i -th fraction, with numerator and denominator separated by a slash (/). The fraction should be presented in the simplest form, but always with a denominator (even if it is the unit).

Sample input

5

Sample output

1/3