Problem 8

Make a Note of It

5 Points

Program Name: note.cpp **Input File: note.dat**

Sometimes teachers intercept notes that you pass around in class. If you are lucky, they simply give them back to you when class is over. If you are unlucky, they read them aloud in class. For this problem, we will create a simple encryption method that may not deter the CIA, but will at least hinder a teacher.

The first step in defining your encryption is to establish the character set that your note can contain. In the table below, you can see the 56-character set. Note that the set starts numbering at zero and there are four punctuation characters. Messages and masks (described later) use the underscore (position 46) instead of the blank. Position 4 is the period; position 37 is the question mark; and position 53 is the exclamation.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
L	x	D	n	.	v	p	J	S	G	P	Y	j	d	V	l	t	N	B	g	b	U	y	E	A	r	c	Z

28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55
R	e	K	k	o	Q	X	h	w	?	m	F	W	s	f	q	H	u	_	z	a	O	i	I	T	!	C	M

Messages are encrypted by shifting the characters of the message forward according to a "mask". For example, if the character "S" were to be encrypted with a mask of "B", we would count forward 18 (the value of "B") starting with "S" giving the value "c". To encrypt a message, we use a multi-character mask (called a key) that is repeated over and over. Each character of the message being encrypted is "masked" by the character in the key.

In the example encryption job below, you can see how the source message "Meet_at_locker_twelve!" would be encrypted using the key "Amber". The position of each character in the source message is added to the position of the corresponding character in the repeated key. You will note that the encryption of the first "M" uses "A" from the repeated key and that the sum of the two positions is computed as 55+24=79. When the resulting position exceeds 55, the character set is considered to wrap such that the "L" is at position 0 and at position 56. Therefore, the position 79 is interpreted to "E".

M	e	e	t	_	a	t	_	l	o	c	k	e	r	_	t	w	e	l	v	e	!
A	m	b	e	r	A	m	b	e	r	A	m	b	e	r	A	m	b	e	r	A	m
E	Y	O	u	l	t	C	P	H	x	i	d	O	C	l	b	B	O	H	K	!	h

With a little work, you can also figure out how to "decrypt" a message given the encrypted message and the key. In the case of a decryption job, an encrypted string is considered the source string.

Input

Input to your program is will consist of a series of encryption/decryption jobs each on a single line of input. Column 1 will contain either a "E" for encrypt or "D" for decrypt. Column 2 will contain a single space character (an actual space and not the underscore from the encryption character set). The source string will begin in column 3 and will be 1 to 50 characters in length and will be followed by a single space. Finally, the input line contains the "key" which is 1 to 10 characters in length and is followed by the end of line. You can be sure that there will be no invalid input (characters outside the above character set plus the space character) nor will there be any invalid formatting.

Output

For each source string, your program should encrypt or decrypt (depending on the job type) the source string and print the result on a line by itself to the screen.

Example: Input File

```
E Meet_at_locker_twelve! Amber
D SilNHofGv.?sNRnHXST Alfalfa
```

Output to screen

```
QVQgqFYvVlWNOg_R_A!eU
Where_is_Buckwheat?
```