425 Enigmatic Encryption

One Saturday morning, Valentine McKee attempted to log into `kevin.bit.edu`, the new BIT supercomputer, to work on her new MPI implementation. After a few failed login attempts, Valentine put her head in her hands. "I can't believe it! I've forgotten my password!"

Valentine thought for a few minutes. She remembered choosing her password as a combination of two words from the body of her thesis. The password program on `kevin` had required her to make the password between 5 and 8 characters long with at least one "special" character (non-alpha), so Valentine had also incorporated a numeral into the password. "Either 0, 2, 4, or 8, between the two words," she recalled. Also, to save typing effort, she had used only lowercase for the letters. She could not recall, however, whether or not the original words had been all lowercase. She did recall that both words were longer than just a single character.

Valentine started to read through the most recent printout of her thesis, trying to jog her memory of the password, but to no avail. Finally, she said, "My thesis is on line in my lab and I have a record there of my encrypted password. I never thought I would have to use it, though."

She sighed as she invoked man `crypt` to determine how `crypt()` processes passwords.

crypt (3C) C Library Functions crypt (3C)

NAME

crypt – generate encryption

SYNOPSIS

char *crypt(const char *_key_, const char *_salt_);

DESCRIPTION

crypt( ) is the password encryption function. It is based on a one-way encryption algorithm with variations intended (among other things) to frustrate use of hardware implementations of a key search.

key is the input string to encrypt, for instance, a user's typed password. Only the first eight characters are used; the rest are ignored. *salt* is a two-character string chosen from the set [**a-zA-Z0-9./**]; this string is used to perturb the hashing algorithm in one of 4096 different ways, after which the input string is used as the key to repeatedly encrypt a constant string. The returned value points to the encrypted input string. The first two characters of the return value are the *salt* itself.

NOTES

The return value for **crypt( )** points to static data that are overwritten by each call.

The **crypt( )** function is provided for you. You do not have to write it yourself.

Note for Pascal Programmers

Include the following in your program so that you can properly call the `pcrypt()` procedure (a wrapper to the C `crypt()` function). The `pcrypt()` procedure is a library procedure that we will supply — you do not have to write it.

```
type
   key_type = array[1..8] of char;
   salt_type = array[1..2] of char;
   encrypt_type = array[1..32] of char;

procedure pcrypt( key : key_type;
                  salt : salt_type;
              var result : encrypt_type); external c;
```

Input

The first line of input is the encrypted password. The next lines of input are the contents of Valentine's thesis. Each input line will consist of 80 or fewer characters of printable ASCII text and will be terminated by a newline. No lines will be empty. Words are assumed to be any contiguous set of alphabetic characters (upper or lower case), separated by whitespace or punctuation.

Output

The output of the program is Valentine's original password.

Sample Input

```
h8E6dqt5lkL9o
The parallel algorithms were executed on the Connection Machine model
CM-2 --- a single-instruction multiple data (SIMD) parallel computer
which, in its largest configuration, contains 65,536 bit-serial
processors and 2048 Weitek floating-point units (FPU's).
The bit-serial processors are clustered together into groups of 16
within a single integrated circuit, and these IC's are connected
together in a 12-dimensional hypercube. Two IC's, or 32 processors,
share a single Weitek FPU. Note that a fully configured CM-2 contains
2048 times as much floating point hardware as a conventional computer
containing a single Weitek FPU (e.g., a SUN-4).
```

Sample Output

```
bit0note
```