128 Software CRC

You work for a company which uses lots of personal computers. Your boss, Dr Penny Pincher, has wanted to link the computers together for some time but has been unwilling to spend any money on the Ethernet boards you have recommended. You, unwittingly, have pointed out that each of the PCs has come from the vendor with an asynchronous serial port at no extra cost. Dr Pincher, of course, recognizes her opportunity and assigns you the task of writing the software necessary to allow communication between PCs.

You've read a bit about communications and know that every transmission is subject to error and that the typical solution to this problem is to append some error checking information to the end of each message. This information allows the receiving program to detect when a transmission error has occurred (in most cases). So, off you go to the library, borrow the biggest book on communications you can find and spend your weekend (unpaid overtime) reading about error checking.

Finally you decide that CRC (cyclic redundancy check) is the best error checking for your situation and write a note to Dr Pincher detailing the proposed error checking mechanism noted below.

> **CRC Generation**
> The message to be transmitted is viewed as a long positive binary number. The first byte of the message is treated as the most significant byte of the binary number. The second byte is the next most significant, etc. This binary number will be called "m" (for message). Instead of transmitting "m" you will transmit a message, "m2", consisting of "m" followed by a two-byte CRC value.
>
> The CRC value is chosen so that "m2" when divided by a certain 16-bit value "g" leaves a remainder of 0. This makes it easy for the receiving program to determine whether the message has been corrupted by transmission errors. It simply divides any message received by "g". If the remainder of the division is zero, it is assumed that no error has occurred.
>
> You notice that most of the suggested values of "g" in the book are odd, but don't see any other similarities, so you select the value 34943 for "g" (the generator value).

You are to devise an algorithm for calculating the CRC value corresponding to any message that might be sent. To test this algorithm you will write a program which reads lines from standard input and writes to standard output.

Input

Each input line will contain no more than 1024 ASCII characters (each line being all characters up to, but not including the end of line character) as input,

The input is terminated by a line that contains a '#' in column 1.

Output

For each input line calculates the CRC value for the message contained in the line, and writes the numeric value of the CRC bytes (in hexadecimal notation) on an output line.

Note that each CRC printed should be in the range 0 to 34942 (decimal).

Sample Input

```
this is a test

A
#
```

Sample Output

```
77 FD
00 00
0C 86
```