CMPUT 275 - Tangible Computing **Winter 2018**
Morning Problem: Punchcards

Description

Our protagonist, let's call him J, found some punchcards in his attic during spring cleaning. Before throwing them away during the cleanup, he wanted to digitize them for posterity. He found a punchcard scanner at the university's museum that is producing scans of the punchcard, in text-file format. A scanned punchcard looks like this in the file:

```
X....XX
X..X...
X.....X
X.X....
X.X.X..
X...X.X
X.X..X.
 .X.....
X..X..X
 .X.XXX.
 .X.....
X...X..
XX.XXXX
XXX.XXX
```

A file can contain images of multiple of these punchcards. From ancient documentation it became clear that all rows represent the bitcode of an ASCII character, where 'X' represents bits of one and '.' represents a bit of zero. The ordering of bits is as shown above the image of the card. Your job is to read in the punchcard data (from the standard input) and write the text that it represents to the standard output.

To be clear, each line has exactly 7 characters, each being 'X' or '.' which correspond to bits '1' and '0' respectively. Convert each line to the corresponding ASCII character and print it out. End each punchcard by printing a newline (see the second example and input/output specification). Some helpful functions are described just after the output specification.

Input

The input will contain the image of one or more punchcards. The length (number of rows) in the punchcards can vary, but it will be at least one. There is a single blank line at the end of each punch card.

Output

Write the text computed to the standard output. After every punchcard, print a newline character `\n`.

Helpful Functions

The function `chr(x)` will return the single-letter string corresponding to the ASCII value `x`. For example, `chr(67)` is `"C"`.

You can also specify the base of the string when casting to an integer. For example, `int("1000011", 2)` is `67` because specifying 2 in the second argument causes the function to treat the string as a binary number.

Sample Input 1

```
X....XX
X..X...
X.....X
X.X....
X.X.X..
X...X.X
X.X..X.
.X.....
X..X..X
.X.XXX.
.X.....
X...X..
XX.XXXX
XXX.XXX
```

Sample Output 1

```
CHAPTER I. Dow
```

Explanation: There is a single punchcard in this example. For example, just working out the first character in the output, the first line in the first punchcard, together with the bit-position indicators is:

```
 6 5 4 3 2 1 0
 X . . . . X X
```

Replace the 'X' characters with 1 and the relevant '.' with 0 to get:

```
 6 5 4 3 2 1 0
 1 0 0 0 0 1 1
```

which gives the binary number 1000011, which is decimal 67. This is the ASCII code of letter C, which is what chr(65) would return in Python.

Sample Input 2

```
XXX.X..
.X.....
XXX.XX.

XX..XX.
.X.....
XXX..XX
```

Sample Output 1

```
t v
f s
```

Explanation: There are two punchcards in this example, each with three characters. The first is `t v` (the space is from binary 0100000) and the second is `f s`. Each punchcard is followed by a single blank line indicating a newline should be printed at the end of the punchcard.