Free2Card

ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST
2008/2009 SOUTHERN CALIFORNIA REGIONAL
Problem #1

Background

Big Fen Industries has a large web presence. From the outside all looks modern, but the internal interfaces are still built on files consisting of 80-character lines that are a legacy of the punch card days.

Normally no one need be aware that data is moving around on 80-character lines in fixed fields, but from time to time it is necessary to add or modify some data. Last week a 30 minute outage, during the busy part of the day, was caused by a line added manually that had some characters in the wrong position. The job of your team is to write a utility that will make it easy to put data into specific locations on 80-character card-image lines.

Input/Output

An output card image has 80 characters, numbered 1-80 from the left.

The input consists of three types of items, separated by one or more spaces:

- Position command: a decimal integer in the range 1-80.
- String: a sequence of characters, starting with a non-whitespace and non-numeric character that is not "!" and ending when that same character occurs again. (Example: Athis is a stringA.) The value of the string is the characters without the initial and final delimiting characters. An end-of-line is not allowed in a string, hence it must fit on one line.
- Punch command: "!".

The input is a series of lines of at most 100 characters, consisting of pairs of positions and strings interspersed with punch commands. There is an 80-character output "card" buffer, initially all blank. The characters making up the value of each string are placed in the buffer starting at the location specified by the preceding position command. Characters added beyond the 80th character in the output buffer are ignored. The punch command causes the buffer to be output (with a final new-line) and reset to all blanks.

For example:

```
12 /this is a test/ 20 /1/ !
```

results in:

```
           this is 1 test
```

Note that there are 11 spaces at the beginning and 55 spaces at the end of this output line.

Sample Input

```
1 #@# 3 @duplicate the deck@ ! 1 @F@ 2 @0180@ ! 1 @APBP@ 14 @LF00180@
! 1 @#@ !
1 @this is the first card@ 77 @0010@ !
1 @this is the second card@ 77 @0020@ !
1 @this is the third card@ 77 @0030@ !
```

Expected Output

```
@ duplicate the deck
F0180
APBP         LF00180
#
this is the first card                                                      0010
this is the second card                                                     0020
this is the third card                                                      0030
```