4983 - Alignment of Code

Europe - Northeastern Europe - 2010/2011

You are working in a team that writes Incredibly Customizable Programming Codewriter (ICPC) which is basically a text editor with bells and whistles. You are working on a module that takes a piece of code containing some definitions or other tabular information and aligns each column on a fixed vertical position, while keeping the resulting code as short as possible, making sure that only whitespaces that are absolutely required stay in the code. So, that the first words on each line are printed at position $p_1 = 1$; the second words on each line are printed at the minimal possible position p_2, such that all first words end at or before position p_2 - 2; the third words on each line are printed at the minimal possible position p_3, such that all second words end at or before position p_3 - 2, etc.

For the purpose of this problem, the code consists of multiple lines. Each line consists of one or more words separated by spaces. Each word can contain uppercase and lowercase Latin letters, all ASCII punctuation marks, separators, and other non-whitespace ASCII characters (ASCII codes 33 to 126 inclusive). Whitespace consists of space characters (ASCII code 32).

Input

The input file contains one or more lines of the code up to the end of file. All lines (including the last one) are terminated by a standard end-of-line sequence in the file. Each line contains at least one word, each word is 1 to 80 characters long (inclusive). Words are separated by one or more spaces. Lines of the code can have both leading and trailing spaces. Each line in the input file is at most 180 characters long. There are at most 1000 lines in the input file.

Output

Write to the output file the reformatted, aligned code that consists of the same number of lines, with the same words in the same order, without trailing and leading spaces, separated by one or more spaces such that i-th word on each line starts at the same position p_i.

Note for the Sample:

The `␣' character in the example below denotes a space character in the actual files (ASCII code 32).

Sample Input

```
  start:  integer;    // begins here
stop: integer; //  ends here
 s:  string;
c:   char; // temp
```

Sample Output

```
start: integer; // begins here
stop:  integer; // ends   here
s:     string;
c:     char;    // temp
```