595 A Major Problem

In western music the twelve notes used in musical notation are identified with the capital letters A through G, possibly followed by a sharp '#' or flat 'b' character, and are arranged as shown below. A slash is used to identify alternate notations of the same note.

```
C/B#   C#/Db   D   D#/Eb   E/Fb   F/E#   F#/Gb   G   G#/Ab   A   A#/Bb   B/Cb
```

Any two adjacent notes in the above list are separated by a *semitone*. Any two notes that have exactly one note separating them in the above list are separated by a *tone*. A *major scale* is composed of eight notes; it begins on one of the above notes and follows the progression tone-tone-semitone-tone-tone- tone-semitone from left to right in the list above, wrapping from B/Cb to C/B# when necessary. For example, the major scales starting on C and Db, respectively, are made up of the following notes:

```
C   D   E  F   G   A   B  C
Db  Eb  F  Gb  Ab  Bb  C  Db
```

The following rules also apply to major scales:

1. The scale will contain each letter from A to G once and only once, with the exception of first letter of the scale which repeated as the last letter of the scale.
2. The scale may not contain a combination of both flat and sharp notes.

The note which begins a major scale is referred to as the key of the scale. For example, the scales above are the scales for the major keys of C and Db, respectively. Transposing notes from one scale to another is a simple matter of replacing a note in one scale with the note in the corresponding position of another scale. For example, the note F in the major key of C would transpose to the note Gb in the major key of Db since both notes occupy the same position (fourth) in their respective scales.

The object of this problem is to write a program that will transpose notes from one major scale to another. An input file will contain several lines, each of which contains a source key, target key, and list of one or more notes to be transposed. An output file should be generated which indicates the transposed notes, as well as indicating major scales that are not valid and notes that are not valid in a given major scale.

Input

Each line of the input, except for the last, will contain two musical keys followed by a list of notes to be transposed from the major scale of the first key to the major scale of the second key. Each list is terminated by a single asterisk character. The final line of the input contains only a single asterisk. All notes on a line and the terminating asterisk are delimited by a single space

Output

For each input line which defines a transposition problem, several lines will be produced in the output file. If the source and target keys are valid, then the first output line for each input line should read "`Transposing from` X `to` Y`:`" where X is the source key and Y is the target key. **If either the source or target key is not valid**, output one line which reads "`Key of` X(or Y) `is not a valid major key`", where X(or Y) is the key that is not valid. If both the source and target key are not valid, report only the source key. The remainder of the input for that line is to be skipped. For input lines which contain valid source and target keys, the first output line will be followed by one output line for each note to be transposed. If the note is a valid note in the major scale of the source key then the output line should read " M `transposes to` N" where M is the note in the source key and N is the corresponding note in the target key. If the input note is not a valid note in the major scale of the source key then the output line should read " M `is not a valid note in the` X `major scale`" where M is the input note and X is the source key. Observe that for either valid or non-valid notes the output line begins with precisely two spaces.

The output data for each input line should be delimited by a single blank line. All output formatting rules described above must be observed precisely. The only exception is that the number of blank lines *at the end of the output* **is to be considered not significant**.

Sample Input

```
C Db F *
Db C Gb *
C B# A B *
C D A A# B Bb C *
A# Bb C *
*
```

Sample Output

```
Transposing from C to Db:
  F transposes to Gb

Transposing from Db to C:
  Gb transposes to F

Key of B# is not a valid major key

Transposing from C to D:
  A transposes to B
  A# is not a valid note in the C major scale
  B transposes to C#
  Bb is not a valid note in the C major scale
  C transposes to D

Key of A# is not a valid major key
```