Cell Phone Messaging

Table Of Contents

- Problem
- Input
- Output
- Sample
- Files
- How to Submit
- README
- .zip archive
- Email Subject Line
- Grading

Problem

Write an application that decodes a sequence of cell phone key presses into text messages. The letters are mapped to the digits as shown below.

Specifications:

```
* To sequentially insert 2 characters from the same key, the user will have to pause between the 1st
and 2nd characters. A pause in key presses is denoted by a space (' ') in the input. For example,
pressing '99' will output an ‘x’. Alternatively, pressing '9 9' will output ‘ww’.
* Pressing '0' will output a space.
* Pressing '#' will clear the last character. If there are no more characters to delete, pressing '#'
will have no effect.
* For the keys mapped to letters, continuing to press the same key will cycle through the letters for
that key. So pressing '99999' will output a 'w'.
* Pressing '*' or '1' has the same effect as pausing, but is otherwise ignored. So pressing '9*119'
will output 'ww'.
```

Input

The first line of input gives the number of cases, N. N test cases follow.

- Each test case is a line of text consisting of a string of valid characters.
- Valid characters include 0-9, #, *, and space.

Output

For each test case, output one line containing "Case #x: " followed by the message translated from the sequence of key presses. Each message should consist of **only lowercase characters a-z** and space characters ' '

Sample

Input

```
4
44 444444
99933117777
333666 6660022 2777
4433555 55566609666777555533#3
```

Output

```
Case #1: hi
Case #2: yes
Case #3: foo  bar
Case #4: hello world
```

Files

Input: messaging.in

How to Submit

You are asked to submit a .zip archive as an email attachment to codingchallenge2011@gmail.com. The archive must contain:

- A README file (see below)
- ALL source files in the **src** directory
- Output file (messaging.out) in the **data** directory which should be generated for the supplied messaging.in
- NO executables

- Note: Keep performance (time and space efficiency) and code quality in mind when developing a workable solution

README

Create a README with the following information:

```
Name:
Email:
Phone:
School:

Brief overview of your solution
```

.zip archive

Your archive must be named according to this format: firstname_lastname_school.zip

Upon unzipping the project, the contents should adhere to the following structure (at the least):

//e.g: John_Smith_IIT.zip

```
John_Smith_IIT (folder)
 | README
 | src
          | source files
 | bin, lib, etc.. (optional)
 | data
         | messaging.out
```

Email Subject Line

Your email must have the subject line set according to this format: firstname_lastname_school

Grading

1. The output file (messaging.out) will be checked for correctness
2. Submissions that do not follow the submission guidelines above will not be considered
3. Winners will be picked at random from the bucket that is produced from the above criteria