# OldPhone exercise

```
 _____
|       |       |       |           |
|   1   |   2   |   3   |           |
|  &'(  |  ABC  |  DEF  |           |
|       |       |       |           |
|-------|-------|-------|-----------|
|       |       |       |           |
|   4   |   5   |   6   |           |
|  GHI  |  JKL  |  MNO  |           |
|       |       |       |           |
|-------|-------|-------|-----------|
|       |       |       |           |
|   7   |   8   |   9   |           |
| PQRS  |  TUV  | WXYZ  |           |
|       |       |       |           |
|-------|-------|-------|-----------|
|       |       |       |           |
|   *   |   0   |   #   |           |
|   ←   |   ␣   |   →   |           |
|_____|_____|_____|_____|
```
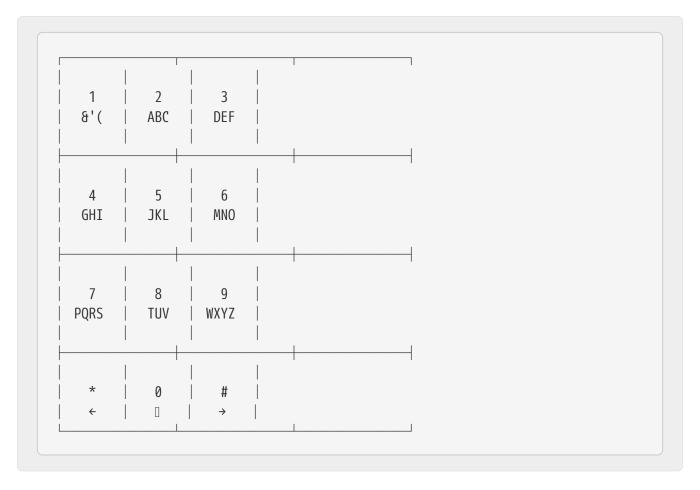
Here is an old phone keypad with alphabetical letters, a backspace key, and a send button.

Each button has a number to identify it and pressing a button multiple times will cycle through the letters on it allowing each button to represent more than one letter.

For example, pressing 2 once will return 'A' but pressing twice in succession will return 'B'.

You must pause for a second in order to type two characters from the same button after each other: "222 2 22" → "CAB".

### The Directions

> Please design and document a class of method that will turn any input to ①
> OldPhonePad into the correct output.
> Assume that a send # will always be included at the end of every input.

① Is the "class of method" supposed to read "class or method"?

This word, I think should be "or" instead, and I have applied this assumption before I have asked for confirmation.

```
public static String OldPhonePad (string input) {
```

```
  // Please write your implementation here!
  }
```

# Input - Output table

Examples:

```
OldPhonePad(33#) => output: E
//add more simple here
OldPhonePad(227*#) => output: B
OldPhonePad(4433555 555666#) => output: HELLO
OldPhonePad(443355 55666#) => output: HEKKO
OldPhonePad(8 88777444666*664#) => output: ?????
```

# Approach

## Our keyboard with these values

| sequence letter | sequence letter | sequence letter |
| --- | --- | --- |
| 1 & | 11 ' | 111 ( |
| 2 a | 22 b | 222 c |
| 3 d | 33 e | 333 f |
| 4 g | 44 h | 444 i |
| 5 j | 55 k | 555 l |
| 6 m | 66 n | 666 o |
| 7 p | 77 q | 777 r 7777 s |
| 8 t | 88 u | 888 v |
| 9 w | 99 x | 999 y 9999 z |

## Pseudo code

- Option 1: . Given a string go through that string and check matches that correspond to an array of hashes holding particular values . constant_number_attributions= [{2:a},{22:b},{222:c}...] .

- Option 2: . constant_number_attributions= [ [a,b,c], [d, e, f], [g, h, i]] . Look at string, split it into an array, then on each element in the array check to see if that matches any of the elements in the constant_number_attributions .

- Option 3: regex matching

- Option 4: counting

# Process Thoughts

*How to get to the numbers*

It's better to have the numbers, represented as strings because we might want to change the value of certain numbers later on, if we need to expand out (i.e have 0 or 000 refer to a symbol or ).

*Ideas for business logic*

```
string.scan(/0+|1+|2+|3+|4+|5+|6+|7+|8+|9+/)
```