Return Keypad

Problem Level: Hard

Problem Description:

Given an integer n, using a phone keypad, find out and return all the possible strings that can be made using digits of input n.

Note : The order of permutations is not important.

Sample Input 1:

```
23
```

Sample Output 1:

```
ad
ae
af
bd
be
bf
cd
ce
cf
```

Approach to be followed:

A phone's keypad has either three or four characters mapped to each alphabet. Given an integer, our task is to find all the possible strings that can be generated using the digits of the integer.

Suppose, the integer is 23. Let us find out the characters associated with each digit of this integer. For 2, the characters are **"a"**, **"b"** and **"c"**. We will pair them with each character corresponding to 3 - **"d"**, **"e"** and **"f"**. For example, all the possible strings ending with **"d"** would be **"ad"**, **"bd"**, and **"cd"**.

This can be easily done using recursion. In our function, we will send the integer as a parameter and separate out the last digit. We will let recursion handle the output for all the integers except the last digit. Let us store this in a separate array, **smallOutput**. To find the final strings, we will iterate over all the strings in our smallOutput, and concatenate each character corresponding to

the last digit at their end. To find the last digit, we will modulo our integer with 10 (For example, to get the last digit from 234, we do 234 % 10, which gives us 4). To find the integer left other than the last digit, we will simply divide our integer with 10 (For example, 234 / 10 = 23).

To find all the characters corresponding to a digit, we shall create a mapping. For example, this mapping will return **"abc"** for digit 2.

We will store all these strings in a final output array.

Steps:

1. Initialise base case as if integer **n** becomes 0, return the list of output containing an empty string (because no string can be formed using integer 0).
2. Find out the last digit and left over integer **n** using modulo and division operations respectively.
3. Using recursion, find the output for the left over integers and store it in **smallerOutput**.
4. Obtain the options for the last digit using the mapping created in **optionsForLastDigit** .
5. For each string in the **smallerOutput,** concatenate each character of **optionsForLastDigit** in the end.
6. Append this concatenation into an output array.
7. Return the output array.

Pseudo Code:

```
function keypad(n)

    if n equals 0

        output[] = {""}   // empty string corresponding to 0

        return output


    lastDigit = n % 10

    smallerNumber = n / 10

    smallerOutput = keypad(smallerNumber)

    optionsForLastDigit = mapping(lastDigit)

    output[] = {}


    k = 0

    for each string in smallerOutput
```

```
        for each character in optionsForLastDigit
            finalString = string + character
            output[k] = finalString
            k++
    return output
```

Time Complexity: $O(4^n)$, where **n** is a number of digits in the input number. Every digit has either 3 or 4 alphabets, so for each digit there can be 4 alphabets as options. For n digit number, there can be 4 options for every n^{th} number. Hence, the time complexity is $O(4^n)$.