Print Keypad

Problem Level: Hard

Problem Description:

Given an integer n, using a phone keypad, find out and print 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
```

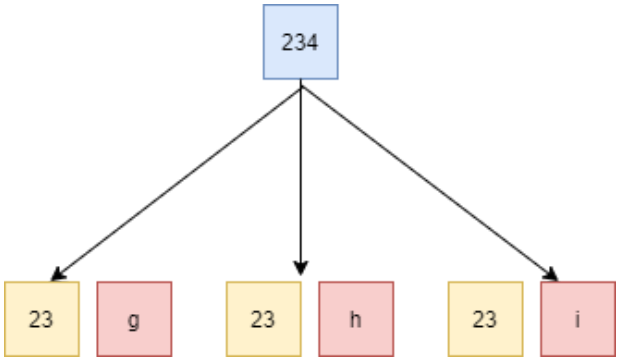
Approach to be followed:

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. This time, instead of storing the possible strings in an array and returning it, we want to print them out. To do this, we will take another parameter in our

function, which will store the output uptil a point. Let us understand this using a diagram. Consider the following tree for n = 23.



For each branch, once we reach the base case, that is, n becomes 0, we will print the output we have until now. This will happen for each recursive call, thereby printing all the possible strings.

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.

Steps:

1. Initialise base case as if integer **n** becomes 0, print the output until now.
2. Find out the last digit and left over integer **n** using modulo and division operations respectively.
3. Obtain the options for the last digit using the mapping created in **optionsForLastDigit** .
4. Using a loop for each character in **optionsForLastDigit ,** new output will be the output uptil now concatenated with each character.
5. Recursively, print all the possible strings for the smallerNumber and new output.

Pseudo Code:

```
function printKeypad(n, outputSoFar)

    if n equals 0

        print(outputSoFar)

        return


    lastDigit = n % 10
```

```
    smallerNumber = n / 10
    smallerOutput = keypad(smallerNumber)
    optionsForLastDigit = mapping(lastDigit)

    for each character in optionsForLastDigit
        newOutput = character + outputSoFar
        printKeypad(smallerNumber, newOutput)
```

Time Complexity: $O(4^n)$, where **n** is a number of digits in the input number.
Each digit of a number has 3 or 4 alphabets, thus it can be said that each digit has 4 alphabets as options. If there are **n** digits then there are 4 options for the first digit and for each alphabet of the first digit there are 4 options in the second digit, i.e for every recursion 4 more recursions are called (if it does not match the base case). Therefore the time complexity is **$O(4^n)$**.