Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



**Example 1:**

**Input:** digits = "23"

**Output:** ["ad","ae","af","bd","be","bf","cd","ce","cf"]

**Example 2:**

**Input:** digits = ""

**Output:** []

**Example 3:**

**Input:** digits = "2"

**Output:** ["a","b","c"]

**Solution:**

class Solution {

public List<String> letterCombinations(String digits) {

List<String> result = new ArrayList<>();

if(digits == null || digits.length() == 0)

return result;

String[] mapping = {"0","1","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"};

letterCombinationsRecursive(result, digits, "", 0, mapping);

return result;

}

public void letterCombinationsRecursive(List<String> result, String digits, String current, int index, String[] mapping){

if(index == digits.length()){

result.add(current);

return;

}

String letters = mapping[digits.charAt(index) - '0'];

for(int i=0;i<letters.length();i++){

letterCombinationsRecursive(result, digits, current+ letters.charAt(i), index+1, mapping);

}

}

}