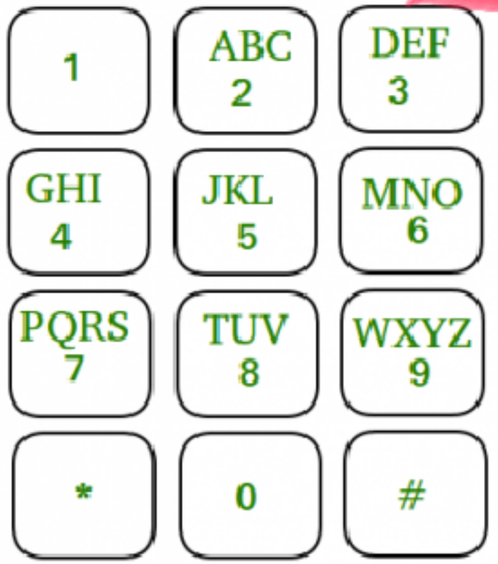
12. Possible Words From Phone Digits

Medium Accuracy: 51.06% Submissions: 8164 Points: 4

Given a keypad as shown in the diagram, and an **N** digit number which is represented by array **a[ ]**, the task is to list all words which are possible by pressing these numbers.



Example 1:

```
Input: N = 3, a[] = {2, 3, 4}
Output:
adg adh adi aeg aeh aei afg afh afi
bdg bdh bdi beg beh bei bfg bfh bfi
cdg cdh cdi ceg ceh cei cfg cfh cfi
Explanation: When we press 2,3,4 then
adg, adh, adi, ... cfi are the list of
possible words.
```

Here the question is asking us to print all the possible strings that could be made using given set of keys.

As it seems

{2, 3, 4}

↓

Now question assumes

2 → a

22 → b

222 → c

```
void printWordsUtil(int number[], int curr_digit,
string output, int n, vector <string> &res)
{
    // Base case, if current output word is prepared
    int i;
    if (curr_digit == n)
    {
        res.push_back(output);
        return ;
    }

    // Try all 3 possible characters for current digir in number[]
    // and recur for remaining digits
    for (i=0; i<strlen(hashTable[number[curr_digit]]); i++)
    {
        output.push_back(hashTable[number[curr_digit]][i]);

        printWordsUtil(number, curr_digit+1, output, n, res);

        if (number[curr_digit] == 0 || number[curr_digit] == 1)
            return;
        output.pop_back();
    }
}
```

→ 2nd of array that contains the given keys.

→ used to backtrack if key, present in the array {1,2,3}

↓

currDigit = 0

Also base case ::

① when all the options have been explored with any set present then simply push the resulting string.

→ next page

```
void printWordsUtil(int number[], int curr_digit,
string output, int n, vector <string> &res)
{
    // Base case, if current output word is prepared
    int i;
    if (curr_digit == n)
    {
        res.push_back(output);
        return ;
    }

    // Try all 3 possible characters for current digir in number[]
    // and recur for remaining digits
    for (i=0; i<strlen(hashTable[number[curr_digit]]); i++)
    {
        output.push_back(hashTable[number[curr_digit]][i]);

        printWordsUtil(number, curr_digit+1, output, n, res);

        if (number[curr_digit] == 0 || number[curr_digit] == 1)
            return;
        output.pop_back();
    }
}
```

(1) Looking for all the keys that a numeric key could hold.

(2) output string we start with first choosing '0' th index as was in the case of backtracking.

(3) Next step sequentially adds to the output string in all the manners.

Redundant code.

When the fn- call returns after a successful run at digits the last chr- to consider the next no. that could have been used.

Solution:

1. 2D array containing diff char of keys at corresponding index.
2. Backtracking based solution,
3. Maintain a digit count for, this will correspond to level of rec call.

for 1,2,3 → currDigit will take values 0, 1, 2

Recursion will exhaust on 0/p.