Letter Combinations of a Phone Number

Given a digit string, return all possible letter combinations that the number could represent.

A mapping of digit to letters (just like on the telephone buttons) is given below.


```
Input:Digit string "23"
Output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].
```

Note: Although the above answer is in lexicographical order, your answer could be in any order you want.

【思路】

迭代，思路见如下例子：

Simple and efficient iterative solution.

Explanation with sample input “123”

Initial state:

result = {""}
Stage 1 for number “1”:

result has {""}
candiate is “abc”
generate three strings “” + “a”, “”+“b”, “”+“c” and put into tmp,
tmp = {“a”, “b”,“c”}
swap result and tmp (swap does not take memory copy)
Now result has {“a”, “b”, “c”}
Stage 2 for number “2”:

result has {“a”, “b”, “c”}
candidate is “def”
generate nine strings and put into tmp,
“a” + “d”, “a”+“e”, “a”+“f”,
“b” + “d”, “b”+“e”, “b”+“f”,
“c” + “d”, “c”+“e”, “c”+“f”
so tmp has {“ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf” }
swap result and tmp
Now result has {“ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf” }
Stage 3 for number “3”:

result has {“ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf” }
candidate is “ghi”
generate 27 strings and put into tmp,
add “g” for each of “ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf”
add “h” for each of “ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf”
add “h” for each of “ad”, “ae”, “af”, “bd”, “be”, “bf”, “cd”, “ce”, “cf”
so, tmp has
{“adg”, “aeg”, “afg”, “bdg”, “beg”, “bfg”, “cdg”, “ceg”, “cfg”
“adh”, “aeh”, “afh”, “bdh”, “beh”, “bfh”, “cdh”, “ceh”, “cfh”
“adi”, “aei”, “afi”, “bdi”, “bei”, “bfi”, “cdi”, “cei”, “cfi” }
swap result and tmp
Now result has
{“adg”, “aeg”, “afg”, “bdg”, “beg”, “bfg”, “cdg”, “ceg”, “cfg”
“adh”, “aeh”, “afh”, “bdh”, “beh”, “bfh”, “cdh”, “ceh”, “cfh”
“adi”, “aei”, “afi”, “bdi”, “bei”, “bfi”, “cdi”, “cei”, “cfi” }
Finally, return result.

```cpp
class Solution {
public:
    vector<string> letterCombinations(string digits) {
        string digit[10] = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
        if(digits == "")    return vector<string>();
        vector<string> res;
        res.push_back("");
         for(int i = 0; i < digits.size(); i++){
             int num = digits[i] - '0';
             if(num < 0 || num > 9) break;
            const string& candidate = digit[num];
             if(candidate.empty()) continue;
             vector<string> tem;
             for(int j = 0; j < res.size(); j++){
                 for(int k = 0; k < candidate.size(); k++){
                     tem.push_back(res[j]+candidate[k]);
                 }
             }
             res.swap(tem);
         }
        return res;
    }
};
```