```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */

#define ALLOC_LENGTH        (100)

char* rep[] = {"", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
char** result;
char tmp_str[256];
int result_index;
int str_len;
int alloc_length;

void recursive(char* str, char* next_str)
{

    char* tmp;
    int tmp_index;

    if(*next_str == '\0')
    {
        *str = '\0';
        tmp = tmp_str;
        result[result_index] = (char*)malloc((str_len+1)*sizeof(char));
        tmp_index = 0;

        while(*tmp != '\0')
        {
            result[result_index][tmp_index] = *tmp;
            tmp++;
            tmp_index++;
        }

        result_index++;

        if(result_index % ALLOC_LENGTH == 0)
        {
            alloc_length+= ALLOC_LENGTH;
            result = (char**)realloc(result, alloc_length*sizeof(char*));
        }
    }else
    {
        tmp = rep[*next_str - '0'];

        while(*tmp != '\0')
        {
            *str = *tmp;
            recursive(str+1, next_str+1);
        }
    }
}

char ** letterCombinations(char * digits, int* returnSize){

    char** result;
    int result_index;
    int asign_index;


    str_len = strlen(digits);
    alloc_length = ALLOC_LENGTH;
    result = (char**)malloc(alloc_length*sizeof(char*));
    result_index = 0;

    recursive(tmp_str, digits);
    *returnSize = result_index;
    return result;
}
```