```c
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
#define MAX_IP_LENGTH           (17)
#define ALLOC_LENGTH            (100)
char** result;
int alloc_length;

void _restoreIpAddresses(char* s, int* returnSize, int ip_index, int group) {
    static char ip[MAX_IP_LENGTH];
    int index;
    int tmp_ip_index;
    int num;

    if(*s == '\0' && 4 == group)
    {
        ip[ip_index] = '\0';
        result[*returnSize] = (char*) malloc(sizeof(char)*MAX_IP_LENGTH);
        memcpy(result[*returnSize], ip, sizeof(char) * MAX_IP_LENGTH);
        (*returnSize)++;

        if( 0 == ((*returnSize) % ALLOC_LENGTH) )
        {
            alloc_length += ALLOC_LENGTH;
            result = (char**) realloc(result, sizeof(char*)*alloc_length);
        }
    }else if(group < 4 && ip_index < MAX_IP_LENGTH)
    {
        num = 0;
        tmp_ip_index = ip_index;

        if(ip_index > 0)
        {
            ip[tmp_ip_index] = '.';
            tmp_ip_index++;
        }

        for(index = 0; index < 3; index++)
        {
            num*=10;
            num+=s[index]-'0';

            if( ( s[index] == '\0' ) ||
                ( (index > 0) &&
                  ('0' == s[0])
                )
              )
            {
                break;
            }

            ip[tmp_ip_index+index] = s[index];

            if(num <= 255)
            {
                _restoreIpAddresses(&s[index+1], returnSize, tmp_ip_index+index+1, group+1);
            }
        }
    }
}

char** restoreIpAddresses(char* s, int* returnSize) {

    alloc_length = ALLOC_LENGTH;
    result = (int**) malloc(sizeof(int*)*alloc_length);
    *returnSize = 0;
    _restoreIpAddresses(s, returnSize, 0, 0);

    return result;
}
```