```c
/**
 * Return an array of arrays of size *returnSize.
 * The sizes of the arrays are returned as *returnColumnSizes array.
 * Note: Both returned array and *columnSizes array must be malloced, assume caller calls free().
 */
#define ALLOC_LENGTH        (100)

bool checkNQueens(int n, int* pos, int count)
{
    int row_index;
    int diag_count;

    diag_count = 1;
    row_index = count-1;

    while(row_index >= 0)
    {
        if( ( pos[row_index] == pos[count] ) ||
            ( ( (pos[count]-diag_count) >= 0 ) && ( pos[row_index] == (pos[count]-diag_count) ) ) ||
            ( ( (pos[count]+diag_count) <= (n-1) ) && ( pos[row_index] == (pos[count]+diag_count) ) )
          )
        {
            return false;
        }

        diag_count++;
        row_index--;
    }

    return true;
}

char** buildChessboard(int n, int* pos)
{
    int row;
    int col;
    char** board;

    board = (char**)malloc(sizeof(char*)*n);
    memset(board, 0x0, sizeof(char*)*n);
    board[0] = (char*)malloc(sizeof(char)*n*(n+1));

    for(row = 0; row < n; row++)
    {
        for(col = 0; col < n; col++)
        {
            if(board[row] == NULL)
            {
                board[row] = board[0] + row*(n+1);
            }

            board[row][col] = (pos[row] == col) ? 'Q':'.';
        }
        board[row][col] = '\0';
    }

    return board;
}

void _solveNQueens(int n, int* pos, int count, char**** result, int* alloc_length, int* returnSize, int** returnColumnSizes) {
    int index;

    for(index = 0; index < n; index++)
    {
        pos[count] = index;

        if( checkNQueens(n, pos, count) )
        {
            if(count >= (n-1))
            {
                (*result)[*returnSize] = buildChessboard(n, pos);
                (*returnColumnSizes)[*returnSize] = n;
                (*returnSize)++;
                if( *alloc_length == *returnSize)
                {

                    (*alloc_length)*=2;
                    (*result) = (char***)realloc((*result), sizeof(char**)*(*alloc_length));
                    *returnColumnSizes = (int*)realloc(*returnColumnSizes, sizeof(int)*(*alloc_length));
                }

            }else
            {
                _solveNQueens(n, pos, count+1, result, alloc_length, returnSize, returnColumnSizes);
            }

        }

    }
}

char*** solveNQueens(int n, int* returnSize, int** returnColumnSizes) {
    char*** result;
    int* pos;
    int alloc_length;
    int index;
    int row;
    int column;

    alloc_length = ALLOC_LENGTH;
    pos = (int*)malloc(sizeof(int)*n);
    result = (char***)malloc(sizeof(char**)*alloc_length);
    *returnColumnSizes = (int*)malloc(sizeof(int)*alloc_length);
    *returnSize = 0;

    _solveNQueens(n, pos, 0, &result, &alloc_length, returnSize, returnColumnSizes);
```

```
    return result;
}
```