```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().
 */

typedef enum
{
    DIRECTION_RIGHT = 0,
    DIRECTION_DOWN,
    DIRECTION_LEFT,
    DIRECTION_UP,
    TOTAL_DIRECTION
}DIRECTION;



int** generateMatrix(int n, int* returnSize, int** returnColumnSizes){
    DIRECTION direction;
    int** result;
    int index;
    int x;
    int y;
    int x_limit;
    int y_limit;
    int total_count;
    int count;
    int round;

    direction = DIRECTION_RIGHT;
    result = (int**)malloc(sizeof(int*)*n);
    *returnColumnSizes = (int*)malloc(sizeof(int)*n);
    memset(result, 0x0, sizeof(int*)*n);
    x = 0;
    y = 0;
    round = 0;
    total_count = n*n;
    count = 1;
    x_limit = n-1;
    y_limit = n-1;
    *returnSize = n;

    while(total_count >= count)
    {
        if( (x < n) && (result[x] == NULL) )
        {
            result[x] = (int*)malloc(sizeof(int)*n);
            (*returnColumnSizes)[x] = n;

        }

        switch(direction)
        {
                case DIRECTION_RIGHT:
                    if(y > y_limit)
                    {
                        direction = DIRECTION_DOWN;
                        x_limit = n-1-round;
                        x++;
                        y--;
                        //printf("DOWN:%d, %d, %d, %d\n", x, y, x_limit, count);
                    }else
                    {
                        //printf("%d, %d, %d, %d\n", x, y, direction, count);
                        result[x][y] = count;
                        count++;
                        y++;
                    }
                    break;
                case DIRECTION_DOWN:
                    if(x > x_limit)
                    {
                        direction = DIRECTION_LEFT;
                        y_limit = round;
                        x--;
                        y--;
                    }else
                    {
                        //printf("%d, %d, %d, %d\n", x, y, direction, count);
                        result[x][y] = count;
                        count++;
                        x++;
                        //printf("xxx, %d\n", x_limit);
                    }
                    break;
                case DIRECTION_LEFT:
                    if(y < y_limit)
                    {
                        direction = DIRECTION_UP;
                        x_limit = round+1;
                        y++;
                        x--;
                    }else
                    {
                        //printf("%d, %d, %d, %d\n", x, y, direction, count);
                        result[x][y] = count;
                        count++;
                        y--;
                    }
                    break;
                case DIRECTION_UP:
                    if(x < x_limit)
                    {
                        direction = DIRECTION_RIGHT;
                        round++;
                        y_limit = n-1-round;
                        x++;
                        y++;
                    }else
                    {
                        //printf("%d, %d, %d, %d\n", x, y, direction, count);
                        result[x][y] = count;
                        count++;
                        x--;
                    }
                    break;
                default:
                    assert(false);
        }


    }
    return result;
}
```