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

typedef enum
{
    SPIRAL_DIR_RIGHT,
    SPIRAL_DIR_DOWN,
    SPIRAL_DIR_LEFT,
    SPIRAL_DIR_UP,
    TOTAL_SPIRAL_DIR
}SPIRAL_DIR;

int* spiralOrder(int** matrix, int matrixSize, int* matrixColSize, int* returnSize) {
    SPIRAL_DIR dir;
    int round;
    int x;
    int y;
    int x_limit;
    int y_limit;
    int count;
    bool change_dir;
    int* result;
    int result_index;
    int round_count;

    dir = SPIRAL_DIR_RIGHT;
    x = 0;
    y = 0;
    round = 0;
    round_count = 0;
    change_dir = false;
    count = 0;
    *returnSize = matrixSize*(*matrixColSize);
    result = (int*)malloc( sizeof(int)*(*returnSize) );
    result_index = 0;
    y_limit = matrixColSize[0];
    x_limit = matrixSize;

    while(true)
    {
        result[result_index] = matrix[x][y];
        result_index++;

        switch(dir)
        {
                case SPIRAL_DIR_RIGHT:
                    y++;
                    if( y >= y_limit )
                    {
                        dir = SPIRAL_DIR_DOWN;
                        y--;
                        result_index--;
                        change_dir = true;
                        if(count)
                        {
                            x_limit = matrixSize - round;
                        }
                        round_count+=count;
                        count = 0;
                    }else
                    {
                        count++;
                    }
                    break;
                case SPIRAL_DIR_DOWN:
                    x++;
                    if( x >= x_limit)
                    {
                        x--;
                        result_index--;
                        dir = SPIRAL_DIR_LEFT;
                        change_dir = true;
                        round_count+=count;
                        if(count)
                        {
                            y_limit = round;
                        }
                        count = 0;
                    }else
                    {
                        count++;
                    }
                    break;
                case SPIRAL_DIR_LEFT:
                    y--;
                    if( y < y_limit )
                    {
                        y++;
                        result_index--;
                        dir = SPIRAL_DIR_UP;
                        change_dir = true;
                        round_count+=count;
                        if(count)
                        {
                            x_limit = round;
                        }
```

```
                        count = 0;
                    }else
                    {
                        count++;
                    }
                    break;
                case SPIRAL_DIR_UP:
                    x--;
                    if( x <= x_limit )
                    {
                        x++;
                        result_index--;
                        dir = SPIRAL_DIR_RIGHT;
                        change_dir = true;
                        round++;
                        round_count+=count;
                        if(count)
                        {
                            y_limit =  matrixColSize[0] - round;
                        }
                        count = 0;
                    }else
                    {
                        count++;
                    }
                    break;

        }

        if(change_dir)
        {
            if( (0 == round_count) && (SPIRAL_DIR_UP == dir) )
            {
                break;
            }else if(SPIRAL_DIR_RIGHT == dir)
            {
                round_count = 0;
            }

            change_dir = false;
        }

    }

    return result;

}
```