题目描述：

疫情期间，小明隔离在家，百无聊赖，在纸上写数字玩。他发明了一种写法：

给出数字**个数 n** 和**行数 m**（$0 < n \leq 999$，$0 < m \leq 999$），从左上角的 1 开始，按照**顺时针螺旋向内写**方式，依次写出 2,3...n，最终形成一个 **m 行矩阵**。

小明对这个矩阵**有些要求**：

1.每行数字的个数一样多

2.列的数量尽可能少

3.填充数字时优先填充外部

4.数字不够时，使用单个*号占位

输入描述：

两个整数，空格隔开，依次表示 n、m

输出描述：

符合要求的唯一矩阵

补充说明：

示例 1

输入：

```
9 4
```

输出：

```
1 2 3

* * 4

9 * 5

8 7 6
```

说明：

9 个数字写成 4 行，最少需要 3 列

示例 2

输入：

```
3 5
```

输出：

```
1

2

3

*

*
```

说明：

3 个数字写 5 行，只有一列，数字不够用*号填充

示例 3

输入：

```
120 7
```

输出：

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 19
45 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 63 20
44 83 114 115 116 117 118 119 120 * * * * * * 99 64 21
43 82 113 112 111 110 109 108 107 106 105 104 103 102 101 100 65 22
42 81 80 79 78 77 76 75 74 73 72 71 70 69 68 67 66 23
41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24
```

说明：

```
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
```

```
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNextInt()) { // 注意 while 处理多个 case
            int a = in.nextInt();
            int b = in.nextInt();

            int[][] matrix = new int[b][a % b == 0 ? a / b : a / b + 1];

            int loop = 1;
            int index = 1;
            while ((loop - 1) * 2 < matrix.length && (loop - 1) * 2 < matrix[0].length) {
                index = func(index, a, loop, matrix);
                loop++;
            }

            for (int i = 0; i < matrix.length; i++) {
                for (int j = 0; j < matrix[0].length; j++) {
                    if (matrix[i][j] == 0) {
                        System.out.print("*" + " ");
                    } else {
                        System.out.print(matrix[i][j] + " ");
                    }
                }
                System.out.println();
            }

        }
    }

    private static int func(int index, int a, int loop, int[][] matrix) {

        if (loop * 2 == matrix.length + 1) {
            for (int i = loop - 1; i <= matrix[0].length - loop; i++) {
                if (index <= a) {
                    matrix[loop - 1][i] = index;
                    index++;
                } else {
                    matrix[loop - 1][i] = 0;
                }
            }
            return index;
        }

        if (loop * 2 == matrix[0].length + 1) {
            for (int i = loop - 1; i <= matrix.length - loop ; i++) {
```

```
            if (index <= a) {
                matrix[i][loop - 1] = index;
                index++;
            } else {
                matrix[i][loop - 1] = 0;
            }
        }
        return index;
    }


    for (int i = loop - 1; i < matrix[0].length - loop; i++) {
        if (index <= a) {
            matrix[loop - 1][i] = index;
            index++;
        } else {
            matrix[loop][i] = 0;
        }
    }

    for (int i = loop - 1; i < matrix.length - loop; i++) {
        if (index <= a) {
            matrix[i][matrix[0].length - loop] =    index;
            index++;
        } else {
            matrix[i][matrix[0].length - loop] = 0;
        }
    }

    for (int i = matrix[0].length - loop; i > loop - 1; i--) {
        if (index <= a) {
            matrix[matrix.length - loop][i] = index;
            index++;
        } else {
            matrix[matrix.length - loop][i] = 0;
        }
    }

    for (int i = matrix.length - loop; i > loop - 1; i--) {
        if (index <= a) {
            matrix[i][loop - 1] = index;
            index++;
        } else {
            matrix[i][loop - 1] = 0;
```

```
                }
            }


            return index;
        }
    }
```