实验环境

```
IDE:CLion
编译器:g++,C++11
运行环境:window7 64位
```

实验原理

在一个 2^k * 2^k 个方格组成的棋盘中，若恰有一个方格与其它方格不同，则称该方格为一特殊方格，且称该棋盘为一特殊棋盘。显然特殊方格在棋盘上出现的位置有 4^k 种情形。因而对任何 k>=0 ，有 4^k 种不同的特殊棋盘。图1-1中的特殊棋盘为 k=2 时 16 个特殊棋盘中的一个。

在棋盘覆盖问题中，要用图1-2所示的4种不同形态的 L 型骨牌覆盖一个给定的特殊棋牌上除特殊方格以外的所有方格，且任何 2 个 L 型骨牌不得重叠覆盖。易知，在任何一个 2^k * 2^k 的棋盘中，用到的 L 型骨牌个数恰为 (4^k-1)/3 。

特殊方格必位于 4 个较小子棋盘之一中，其余 3 个子棋盘中无特殊方格。为了将这 3 个无特殊方格的子棋盘转化为特殊盘，我们可以用一个 L 型骨牌覆盖这 3 个较小的棋盘的汇合处，如下图所示，这 3 个子棋盘上被 L 型骨牌覆盖的方格就成为该棋盘上的特殊方格，从而将原问题化为 4 个较小规模的棋盘覆盖问题。递归的使用这种分割，直至棋盘简化为 2^1x2^1 棋盘。

========

代码如下

```
//ChessBoard.h
//create by mly
//
#ifndef CHESSBOARD
#define CHESSBOARD
const int maxn = 100 + 10;
int Board[maxn][maxn];
int tile = 0;

void ChessBoard(int tr, int tc, int dr, int dc, int size) {
    if (size == 1)
        return;
    int t = ++tile;
    int s = size / 2;
    if (dr < tr + s && dc < tc + s)
        ChessBoard(tr, tc, dr, dc, s);
    else {
        Board[tr + s - 1][tc + s - 1] = t;
        ChessBoard(tr, tc + s, tr + s - 1, tc + s - 1, s);
    }
    if (dr < tr + s && dc >= tc + s)
        ChessBoard(tr, tc + s, dr, dc, s);
    else {
        Board[tr + s - 1][tc + s] = t;
        ChessBoard(tr, tc + s, tr + s - 1, tc + s, s);
    }
    if (dr >= tr + s && dc < tc + s)
        ChessBoard(tr + s, tc, dr, dc, s);
    else {
        Board[tr + s][tc + s - 1] = t;
        ChessBoard(tr + s, tc, tr + s, tc + s - 1, s);
    }
    if (dr >= tr + s && dc >= tc + s)
        ChessBoard(tr + s, tc + s, dr, dc, s);
    else {
        Board[tr + s][tc + s] = t;
        ChessBoard(tr + s, tc + s, tr + s, tc + s, s);
    }
}

#endif //CHESSBOARD
```

运行结果

main.cpp函数

```
#include <bits/stdc++.h>
#include "ChessBoard.h"

using namespace std;

int main() {
    memset(Board, 0, sizeof(Board));
    int n, dr, dc;
    cin >> n >> dr >> dc;
    ChessBoard(0, 0, dr, dc, (int) pow(2, n));
    for (int i = 0; i < (int) pow(2, n); ++i) {
        for (int j = 0; j < (int) pow(2, n); ++j) {
            cout << Board[i][j] << " ";
        }
        cout << endl;
    }
}
```

输入

样例1

```
2 1 0
```

输出

样例1

```
2 2 3 3
0 2 1 3
4 1 1 5
4 4 5 5
```