1401. Игроки

Ограничение времени: 2.0 секунды
Ограничение памяти: 64 МБ

Известно, что господин Чичиков зарабатывал свой капитал и таким способом: он спорил со всякими недотёпами, что сможет доказать, что квадратную доску размера 512×512 нельзя замостить следующими фигурами:

```
X      XX      X      XX
XX     X      XX       X
```

и всегда выигрывал. Однако один из недотёп оказался не так уж глуп, и сказал, что сможет замостить такими фигурами доску размера 512 × 512 без правой верхней клетки. Чичиков, не подумав, ляпнул, что он вообще может любую доску размера 2n × 2n без одной произвольной клетки замостить такими фигурами. Слово за слово, они поспорили. Чичиков чувствует, что сам он не докажет свою правоту. Помогите же ему!

Исходные данные

В первой строке записано целое число n $(1 \le n \le 9)$. Во второй строке через пробел даны два целых числа x, y: координаты «выколотой» клетки доски $(1 \le x, y \le 2n)$, x — номер строки, y — номер столбца. Левый верхний угол доски имеет координаты $(1, 1)$.

Результат

Ваша программа должна выдать $2n$ строчек по $2n$ чисел в каждой строке. На месте выбитой клетки должно стоять число 0. На месте остальных клеток должны стоять числа от 1 до $(22n - 1)/3$ — номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, выведите «−1».

Примеры

исходные данные	результат
2 1 1	0 1 3 3 1 1 4 3 2 4 4 5 2 2 5 5

Успешная попытка

Автор: Aleksei Smirnov • Задача: Игроки

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10895126	22:54:43 28 фев 2025	Aleksei Smirnov	1401. Игроки	Clang++ 17 x64	Accepted		0.031	1 420 КБ

Решение

```
#include <iostream>

long m = 3;
int N;
int grid[512][512] = {};
int HX, HY = 0;

inline void input(int* n, int* hx, int *hy) {
std::cin >> *n >> *hx >> *hy;
```

```
}

inline void place(int x, int y) {
grid[y][x] = m++/3;
}

inline void place0(int x, int y) {
if (x == HX && y == HY)
grid[y][x] = 0;
}


void generate(int n, int x, int y, int hx, int hy);
void generate(int n, int x, int y, int hx, int hy) {


std::string prefix;
for (int i = 0; i < N-n; i++) {
prefix += "  ";
}

// hx and hy ALWAYS are inside current square
// they are relative to x, y
// (x, y) is the top-left cell of given square.

// x x
// x x
if (n == 1) {
// std::cout << '\n';
if (hx == 0) {
if (hy == 0) {
// 0 x
// x x
place0(x, y);
place(x+1, y);
place(x, y+1);
place(x+1, y+1);

} else {
// x x
// 0 x
place(x, y);
place(x+1, y);
place0(x, y+1);
place(x+1, y+1);
}
} else {
if (hy == 0) {
```

```
// x 0
// x x
place(x, y);
place0(x+1, y);
place(x, y+1);
place(x+1, y+1);

} else {
// x x
// x 0
place(x, y);
place(x+1, y);
place(x, y+1);
place0(x+1, y+1);
}
}
return;
}

int side = (1 << n);

if (hx < side/2) {
if (hy < side/2) {
// x x x x
// x 0 # x
// x # # x
// x x x x

place(x+side/2,  y+side/2);
place(x+side/2,  y+side/2-1);
place(x+side/2-1,y+side/2);

generate(n-1,        x,        y,       hx,       hy);
generate(n-1, x+side/2,        y,        0, side/2-1);
generate(n-1,        x, y+side/2, side/2-1,        0);
generate(n-1, x+side/2, y+side/2,        0,        0);

} else {
// x x x x
// x # # x
// x 0 # x
// x x x x

place(x+side/2,  y+side/2);
place(x+side/2,  y+side/2-1);
place(x+side/2-1,y+side/2-1);

generate(n-1, x       , y       , side/2-1, side/2-1);
```

```
generate(n-1, x+side/2, y       ,        0, side/2-1);
generate(n-1, x       , y+side/2,       hx,hy-side/2);
generate(n-1, x+side/2, y+side/2,        0,        0);

}
} else {
if (hy < side/2) {
// x x x x
// x # 0 x
// x # # x
// x x x x

place(x+side/2,  y+side/2);
place(x+side/2-1,y+side/2);
place(x+side/2-1,y+side/2-1);

generate(n-1, x       , y       ,   side/2-1, side/2-1);
generate(n-1, x+side/2, y       ,  hx-side/2,       hy);
generate(n-1, x       , y+side/2,   side/2-1,        0);
generate(n-1, x+side/2, y+side/2,          0,        0);

} else {
// x x x x
// x # # x
// x # 0 x
// x x x x

place(x+side/2-1,y+side/2-1);
place(x+side/2-1,y+side/2);
place(x+side/2,  y+side/2-1);

generate(n-1, x       , y       ,   side/2-1,   side/2-1);
generate(n-1, x+side/2, y       ,          0,   side/2-1);
generate(n-1, x       , y+side/2,   side/2-1,          0);
generate(n-1, x+side/2, y+side/2,  hx-side/2,  hy-side/2);
}
}
return;
}

int main() {
int n, hx, hy;
input(&n, &hx, &hy);
N = n;

HX = --hx, HY = --hy;

generate(n, 0, 0, hx, hy);
```

```
for (int x = 0; x < (1 << N); x++) {
for (int y = 0; y < (1 << N); y++) {
std::cout << grid[y][x] << " ";
}
std::cout << "\n";
}
}
```

Про решение

Решение по индукции.

Базовый случай: очевидно, что можно замостить любой квадрат при $n = 1$.

Индукционный переход: Предположим, что возможно замостить квадрат любого размера, если известно, что в нем ровно одна «дырка». Тогда для квадрата размера 2^{n+1} задача решается следующим образом:

Где серый квадрат со стороной 2^n — квадрат с «дыркой». Таким образов, поместив жёлтую фигуру в углу серого квадрата, создаём «дырку» для каждого из трех квадратов со сторой 2^n, которые можем замостить.