

*Национальный исследовательский университет ИТМО   
(Университет ИТМО)*

*Факультет систем управления и робототехники*

Дисциплина: Алгоритмы и структуры данных

**Отчет по практической работе (1401 задача).**

Студент:

*Евстигнеев Дмитрий*

Группа: *R3242*

Преподаватель:

*Тропченко Андрей Александрович*

Санкт-Петербург

2021

**Цель:** написать программу для решения задачи №1401 на сайте Timus Online

**Задача:**

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 2*n* × 2*n* 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 ≤ *n* ≤ 9). Во второй строке через пробел даны два целых числа *x*, *y*: координаты «выколотой» клетки доски (1 ≤ *x*, *y* ≤ 2*n*), *x* — номер строки, *y* — номер столбца. Левый верхний угол доски имеет координаты (1, 1).

**Результат**

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

**Пример**

|  |  |
| --- | --- |
| **исходные данные** | **результат** |
| 2  1 1 | 0 1 3 3  1 1 4 3  2 4 4 5  2 2 5 5 |

**Принято системой (JUDGE\_ID: 231802FR):**



**Решение на языке С++:**

#include <iostream>

using namespace std;

class Gamers {

private:

int N;

int size;

int\*\* matrix;

void add\_figure(int i, int j) {

int tmp\_i = i % 2;

int tmp\_j = j % 2;

if (tmp\_i == 1 && tmp\_j == 1) {

matrix[i - 1][j - 1] = matrix[i - 1][j] = matrix[i][j - 1] = N++;

}

else if (tmp\_i == 0 && tmp\_j == 0) {

matrix[i + 1][j + 1] = matrix[i + 1][j] = matrix[i][j + 1] = N++;

}

else if (tmp\_i == 0 && tmp\_j == 1) {

matrix[i + 1][j] = matrix[i][j - 1] = matrix[i + 1][j - 1] = N++;

}

else if (tmp\_i == 1 && tmp\_j == 0) {

matrix[i - 1][j] = matrix[i][j + 1] = matrix[i - 1][j + 1] = N++;

}

}

void main\_solve(int rows\_start, int cols\_start, int i, int j, int size) {

if (size == 2) {

add\_figure(rows\_start + i % 2, cols\_start + j % 2);

return;

}

int new\_size = size / 2;

if (i >= new\_size && j >= new\_size) {

main\_solve(rows\_start + new\_size, cols\_start + new\_size, i - new\_size, j - new\_size, new\_size);

matrix[rows\_start + new\_size - 1][cols\_start + new\_size - 1] =

matrix[rows\_start + new\_size][cols\_start + new\_size - 1] =

matrix[rows\_start + new\_size - 1][cols\_start + new\_size] = N++;

main\_solve(rows\_start, cols\_start + new\_size, new\_size - 1, 0, new\_size);

main\_solve(rows\_start, cols\_start, new\_size - 1, new\_size - 1, new\_size);

main\_solve(rows\_start + new\_size, cols\_start, 0, new\_size - 1, new\_size);

}

else if (i < new\_size && j < new\_size) {

main\_solve(rows\_start, cols\_start, i, j, new\_size);

matrix[rows\_start + new\_size][cols\_start + new\_size] =

matrix[rows\_start + new\_size][cols\_start + new\_size - 1] =

matrix[rows\_start + new\_size - 1][cols\_start + new\_size] = N++;

main\_solve(rows\_start + new\_size, cols\_start + new\_size, 0, 0, new\_size);

main\_solve(rows\_start, cols\_start + new\_size, new\_size - 1, 0, new\_size);

main\_solve(rows\_start + new\_size, cols\_start, 0, new\_size - 1, new\_size);

}

else if (i < new\_size && j >= new\_size) {

main\_solve(rows\_start, cols\_start + new\_size, i, j - new\_size, new\_size);

matrix[rows\_start + new\_size][cols\_start + new\_size] =

matrix[rows\_start + new\_size][cols\_start + new\_size - 1] =

matrix[rows\_start + new\_size - 1][cols\_start + new\_size - 1] = N++;

main\_solve(rows\_start, cols\_start, new\_size - 1, new\_size - 1, new\_size);

main\_solve(rows\_start + new\_size, cols\_start + new\_size, 0, 0, new\_size);

main\_solve(rows\_start + new\_size, cols\_start, 0, new\_size - 1, new\_size);

}

else {

main\_solve(rows\_start + new\_size, cols\_start, i - new\_size, j, new\_size);

matrix[rows\_start + new\_size][cols\_start + new\_size] =

matrix[rows\_start + new\_size - 1][cols\_start + new\_size] =

matrix[rows\_start + new\_size - 1][cols\_start + new\_size - 1] = N++;

main\_solve(rows\_start, cols\_start, new\_size - 1, new\_size - 1, new\_size);

main\_solve(rows\_start + new\_size, cols\_start + new\_size, 0, 0, new\_size);

main\_solve(rows\_start, cols\_start + new\_size, new\_size - 1, 0, new\_size);

}

}

public:

Gamers(int\*\* matrix, int size) {

this->matrix = matrix;

this->size = size;

N = 1;

}

void solve(int i, int j) {

main\_solve(0, 0, i, j, size);

}

};

int main() {

int degree = 0, i = 0, j = 0;

cin >> degree >> i >> j;

int count = 1 << degree;

int\*\* matrix = new int\*[count];

for (int i = 0; i < count; i++) {

matrix[i] = new int[count];

for (int j = 0; j < count; j++)

matrix[i][j] = 0;

}

Gamers gamers(matrix, count);

gamers.solve(i - 1, j - 1);

for (int i = 0; i < count; i++) {

for (int j = 0; j < count; j++)

cout << matrix[i][j] << " ";

cout << endl;

}

return 0;

}

**Суть алгоритма:**

Основная суть в том, что мы делим каждый раз нашу задачу на 4. То есть делим поле на 4 части, пока размер одной такой части не будет 2 х 2.

Сначала доходим до части, где стоит ноль, и зарисовываем остальные три точки. После (когда возвращаеся из квадрата 2 х 2 в исходный квадрат 4 х 4), зарисовываем три точки, в зависимости от того, где находился наш квадрат 2 х 2. Далее зарисовываем все остальные.   
  
**Примеры работы программы:**

  
  
