Задача №1726. Кто ходит в гости...

Программный комитет школьных соревнований по программированию, проходящих в УрГУ — многочисленная, весёлая и дружная команда. Дружная настолько, что общения в университете им явно не хватает, поэтому они часто ходят друг к другу в гости. Все ребята в программном комитете очень спортивные и ходят только пешком.

Однажды хранитель традиций олимпиадного движения УрГУ подумал, что на пешие прогулки от дома к дому члены программного комитета тратят слишком много времени, которое могли бы вместо этого потратить на придумывание и подготовку задач. Чтобы доказать это, он решил посчитать, какое расстояние в среднем преодолевают члены комитета, когда ходят друг к другу в гости. Хранитель традиций достал карту Екатеринбурга, нашёл на ней дома всех членов программного комитета и выписал их координаты. Но координат оказалось так много, что хранитель не смог справиться с этой задачей самостоятельно и попросил вас помочь ему.

Город Екатеринбург представляет собой прямоугольник со сторонами, ориентированными по сторонам света. Все улицы города идут строго с запада на восток или с севера на юг, проходя через весь город от края до края. Дома всех членов программного комитета расположены строго на пересечении каких-то двух перпендикулярных улиц. Известно, что все члены комитета ходят только по улицам, поскольку идти по тротуару гораздо приятнее, чем по дворовым тропинкам. И, конечно, при переходе от дома к дому они всегда выбирают кратчайший путь. Программный комитет очень дружный, и все его члены ходят в гости ко всем одинаково часто.

Исходные данные:

> Первая строка содержит целое число n — количество членов программного комитета ($2 \le n \le 105$). В i-й из следующих n строк через пробел записаны целые числа x_i, y_i — координаты дома i-го члена программного комитета ($1 \le x_i,\ y_i \le 106$).

Результат:

> Выведите среднее расстояние, которое проходит член программного комитета от своего дома до дома своего товарища, округлённое вниз до целых.

Рабочий код

```cpp
#include <iostream>
#include <vector>
using namespace std;
void quicksort(vector<int> &vec, int L, int R) {
    int i = L, j = R, mid = L + (R - L) / 2, piv = vec[mid];
    while (i < R || j > L) {
        while (vec[i] < piv) i++;
        while (vec[j] > piv) j--;
        if (i <= j) {
            int temp = vec[i];
            vec[i] = vec[j];
            vec[j] = temp;
            i++; j--;
        } else {
            if (i < R) quicksort(vec, i, R);
            if (j > L) quicksort(vec, L, j);
            return;
        }
    }
}
int main() {
    long long n, sum = 0;
    cin >> n;
    vector<int> xCoords(n);
    vector<int> yCoords(n);
    for (long long i = 0; i < n; i++) cin >> xCoords[i] >> yCoords[i];
    quicksort(xCoords, 0, n - 1);
    quicksort(yCoords, 0, n - 1);
    for (long long i = 1; i < n; i++) {
        int dX = xCoords[i] - xCoords[i - 1], dY = yCoords[i] - yCoords[i-1];
        sum += (dX + dY) * i * (n - i) * 2;
    }
    sum = sum / (n * (n - 1));
    cout << sum << endl;
    return 0;
}
```

Объяснение алгоритма

Алгоритм сначала сортирует координаты точек по осям x и y с помощью быстрой сортировки (quicksort). Затем он вычисляет сумму расстояний между каждой парой точек по оси x и по оси y, суммируя вклад каждой точки в общую сумму на основе ее позиции в отсортированном списке. В итоге, он выводит среднее значение суммы расстояний между всеми парами точек.

Задача №1401. Игроки

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 $2^n \times 2^n$ 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 ($1 \le n \le 9$). Во второй строке через пробел даны два целых числа x, y: координаты «выколотой» клетки доски ($1 \le x,\ y \le 2^n$), 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 $(2^{2n} - 1) : 3$ — номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, выведите «–1».

Рабочий код

```cpp
#include <iostream>
#include <vector>
using namespace std;
void paint(vector<vector<int>> &table, int x, int y, int i, int j, int n) {
    static int c;
    c++;
    int voidPosX = x - j, voidPosY = y - i;
    n = n / 2;
    int ax = i + n, ay = j + n;
    int bx = i + n - 1, by = j + n;
    int cx = i + n, cy = j + n - 1;
    int dx = i + n - 1, dy = j + n - 1;
    if (voidPosX >= n and voidPosY >= n) {
        table[bx][by] = c;
        table[cx][cy] = c;
        table[dx][dy] = c;
        if (n > 1) {
            paint(table, x, y, ax, ay, n);
            paint(table, by, bx, i, by, n);
            paint(table, cy, cx, cx, j, n);
            paint(table, dy, dx, i, j, n);
        }
    } else if (voidPosX >= n and voidPosY < n) {
        table[ax][ay] = c;
        table[cx][cy] = c;
        table[dx][dy] = c;
        if (n > 1) {
            paint(table, ay, ax, ax, ay, n);
            paint(table, x, y, i, by, n);
            paint(table, cy, cx, cx, j, n);
            paint(table, dy, dx, i, j, n);
        }
    } else if (voidPosX < n and voidPosY >= n) {
        table[ax][ay] = c;
        table[bx][by] = c;
        table[dx][dy] = c;
        if (n > 1) {
            paint(table, ay, ax, ax, ay, n);
            paint(table, by, bx, i, by, n);
            paint(table, x, y, cx, j, n);
            paint(table, dy, dx, i, j, n);
        }
    } else if (voidPosX < n and voidPosY < n) {
        table[ax][ay] = c;
        table[bx][by] = c;
        table[cx][cy] = c;
```

```
        if (n > 1) {
            paint(table, ay, ax, ax, ay, n);
            paint(table, by, bx, i, by, n);
            paint(table, cy, cx, cx, j, n);
            paint(table, x, y, i, j, n);
        }
    }
    return;
}
int main() {
    int n, x, y;
    cin >> n >> y >> x;
    n = 1 << n;
    x--;
    y--;
    vector<vector<int>> table(n, (vector<int>(n)));
    paint(table, x, y, 0, 0, n);
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) cout << table[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

Объяснение алгоритма

Алгоритм заполняет поле размером $2^n \times 2^n$ с одной пропущенной клеткой (дыркой) с помощью рекурсивного метода. Он разбивает поле на четыре квадранта, отмечает три клетки в центральных позициях и рекурсивно заполняет каждый из квадрантов, пока не достигнет базового случая. В итоге, каждая клетка поля получает уникальное значение, соответствующее шагу рекурсии, что позволяет полностью покрыть поле плитками L-образной формы.

Задача №1521. Военные учения 2

В соответствии с этой схемой учения делятся на N раундов, в течение которых N солдат, последовательно пронумерованных от 1 до N, маршируют друг за другом по кругу, т.е. первый следует за вторым, второй за третьим, ..., $(N-1)$-й за N-м, а N-й за первым. В каждом раунде очередной солдат выбывает из круга и идёт чистить унитазы, а оставшиеся продолжают маршировать. В очередном раунде выбывает солдат, марширующий на K позиций впереди выбывшего на предыдущем раунде. В первом раунде выбывает солдат с номером K.

Разумеется, г-н Шульман не питал никаких надежд на то, что солдаты в состоянии сами определить очерёдность выбывания из круга. «Эти неучи даже траву не могут ровно покрасить», – фыркнул он и отправился за помощью к прапорщику Шкурко.

Исходные данные:

> Единственная строка содержит целые числа N ($1 \le N \le 10^5$) и K ($1 \le K \le N$).

Результат:

> Вывести через пробел номера солдат в порядке их выбывания из круга.

Рабочий код

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int N, K;
    cin >> N >> K;
    vector<int> soldiers(N); // Вектор для хранения солдат
    for (int i = 0; i < N; ++i) soldiers[i] = i + 1; // Инициализация солдат

    int index = K - 1; // Начальная позиция для удаления
    while (!soldiers.empty()) {
        index = index % soldiers.size(); // Обновление индекса
        cout << soldiers[index] << " "; // Вывод номера солдата
        soldiers.erase(soldiers.begin() + index); // Удаление солдата из круга
        index += K - 1; // Переход к следующему солдату
    }

    return 0;
}
```

Объяснение алгоритма

Этот алгоритм использует «задачу Иосифа Флавия» или «игру в горячо-холодно» для получения порядка выбывания солдат. Он использует вектор для хранения номеров солдат, которые образуют круг. В каждой итерации солдат с определенным номером удаляется из круга, а затем индекс для удаления обновляется на основе заданного шага K. Алгоритм продолжает это, выводя номера солдат в порядке их удаления из круга, до тех пор, пока вектор с номерами солдат не станет пустым.