# Pop_corn

## Описание игры

Игра проводится на квадратном поле размером 40x40, на котором размещено несколько орешков разной стоимости и несколько белочек. Каждый игрок управляет одной белочкой. Игроки делают ходы по очереди. Ход состоит в перемещении белочки на одну из четырех соседних клеток. Ход можно пропустить, то есть остаться на месте. Если белочка попадает на клетку с орешком, то она съедает этот орешек и число набранных этой белочкой очков увеличивается на стоимость орешка. Игра заканчивается, когда на поле съедены все орешки. Побеждает белочка, набравшая наибольшее число очков.

## Формат входных данных

Программа читает данные со стандартного ввода.
Первая строка входных данных содержит целое число M - количество орешков на поле. Далее идет M строк, каждая из которых содержит три числа: координаты x и y орешка и его стоимость. Координаты орешка - это целые числа от 1 до 40, а стоимость - целое число от 1 до 9.
В следующей строке записано два числа: количество белочек на поле N и номер белочки, которой вы управляете (число от 1 до N). Далее идет N строк, содержащих описания белочек. Каждая строка содержит три числа: координаты белочки и текущий счет данной белочки (то есть сумма стоимостей съеденных ею орешков).
Ни в одной клетке не может находиться два ореха или две белочки одновременно.

## Формат выходных данных

Программа должна вывести на стандартный вывод два числа: новые координаты белочки, которой управляет игрок. Белочка должна переместиться в одну из четырех соседних по стороне клеток или остаться в той же клетке. Клетка, в которую перемещается белочка, должна находиться в границах поля и не должна быть занята другой белочкой. Все некорректные ходы игнорируются, то есть белочка остается на месте. После вывода координат белочки необходимо вывести символ перехода на новую строку.

## Как принять участие в соревновании?

Каждому участнику соревнований выдается следующий комплект:
-визуализатор соревнований Visualizer.py и библиотека pygame, используемая им;
-скрипт для проведения соревнований;

-пример решения, в котором белочка стоит на месте.
Участник должен предоставить жюри свое решение в виде исходного кода написанной им программы. Решение считается принятым, если оно проходит ряд тестовых ситуаций в контесте ejudge.

# Challenge

Жюри оставляет за собой право менять условия игры, продолжительность раунда и правила возникновения орешков на поле. Правила отбора для заключительного раунда также могут меняться в зависимости от количества участников. Каждый участник может совершить любое число посылок в ejudge, однако, для соревнований будет принято последнее, прошедшее все начальные тесты.
Начальные тесты предназначены для отсева некорректно написанных решений. Они состоят из набора ситуаций, в каждой из которых белочка должна совершить корретный ход, то есть она не должна ходить в клетки, занятые другими белками или находящиеся вне поля. Хоть такие ходы и будут игнорироваться во время соревнований, за их совершение в ejudge решение участника не будет принято к участию.

# Пример начального теста

Стандартный ввод:
1
2 1 1
2 1
1 1 0
1 2 0
Допустимые ходы:
2 1
1 1
Пояснение:
первый ход соответствует поеданию орешка, второй - пропуску хода. Ход 1 2 недопустим, так как клетка (1, 2) занята другой белочкой.

# Пример программы, которая стоит на месте

```
#include <iostream>
#include <vector>

using namespace std;

struct inch {
    int x, y, cost;
};
```

```
struct squirrel {
    int x, y, score;
};

int main()
{
    int m;
    cin >> m;
    vector <inch> a(m);
    for (int i = 0; i < m; ++i)
        cin >> a[i].x >> a[i].y >> a[i].cost;
    int n, k;
    cin >> n >> k;
    vector <squirrel> b(n);
    for (int i = 0; i < n; ++ i)
        cin >> b[i].x >> b[i].y >> b[i].score;
    cout << b[k - 1].x << " " << b[k - 1].y;
    return 0;
}
```

# Инструкция по работе с визуализатором

Визуализатор работает с уже скомпилированными решениями. Для того, чтобы просмотреть раунд игры с участием ваших решений, вам необходимо поместить в одной папке файлы compete, Visualizer, bots и исполняемые файлы ваших решений.
В файле bots.txt в первой строке необходимо написать число тестируемых решений, а в последующих - названия исполняемых файлов, по одному в строке.
Для запуска раунда необходимо запустить файл compete.

Не забудьте установить библиотеку pygame, без нее визуализатор работать не будет!

**По всем вопросам, связанным с соревнованием, обращайтесь к его организаторам:**
Денис Кириенко
Алексей Золотов
Артем Таболин
Павел Мельничук