Задача 1(Железные лапы, живая душа)

Мальчик Вова воровал яблоки, а киберпес преследовал его.

MUKHTAR-2-02 на службе уже больше тридцати лет, и с его точки зрения погоня проходит по ходам, а пространство разбито на клетки. За один ход Вова может пробежать Vv клеток, а киберпес - Vm клеток. Если в конце хода они оказываются на одной клетке или киберпес оказался впереди, то он вцепляется в Вову мертвой хваткой и вызывает полицию через новейшую сеть 20G.

Также, Вова несет N мешков с яблоками. Каждый мешок снижает скорость Вовы на 1 (вплоть до нуля). Каждый ход Вова вначале скидывает один мешок (пока они есть), а потом бежит, сколько может.

В начальный момент времени MUKHTAR-2-02 находится на расстоянии L клеток от Вовы, а Вова на расстоянии K клеток от забора. Если Вова добирается до забора раньше киберпса, то он выигрывает и уносит столько мешков, сколько у него осталось после скидывания каждый ход. Если пес догоняет Вову, Вова проигрывает и остается без яблок.

Определите, сколько мешков с яблоками удалось украсть мальчику Вове.

Формат ввода

Пять целых неотрицательных чисел в диапазоне от 0 до 1000: Vv, Vm, L, K, N.

- Vv - максимальная скорость Вовы.
- Vm - скорость киберпса.
- L - начальное расстояние от киберпса до Вовы.
- K - начальное расстояние от Вовы до забора.
- N - начальное количество мешков у Вовы.

Формат вывода

Одно целое число - сколько мешков унес Вова.

Например:

Ввод	Результат
5 3 3 11 5	0
6 3 5 9 6	2
10 2 10 30 100	0

Задача 2(Соболь)

Соболь прыгал по заметенному снегом бревну и оставил на нем следы. При этом он аккуратно ставил лапки, и после каждого прыжка передние оказывались на одном целом числе, а задние - на другом (см. рис. 1).

Ведущий орнитолог уже определил, в какой последовательности эти следы были оставлены. Нужно определить минимальный размер соболя, который мог бы это сделать.

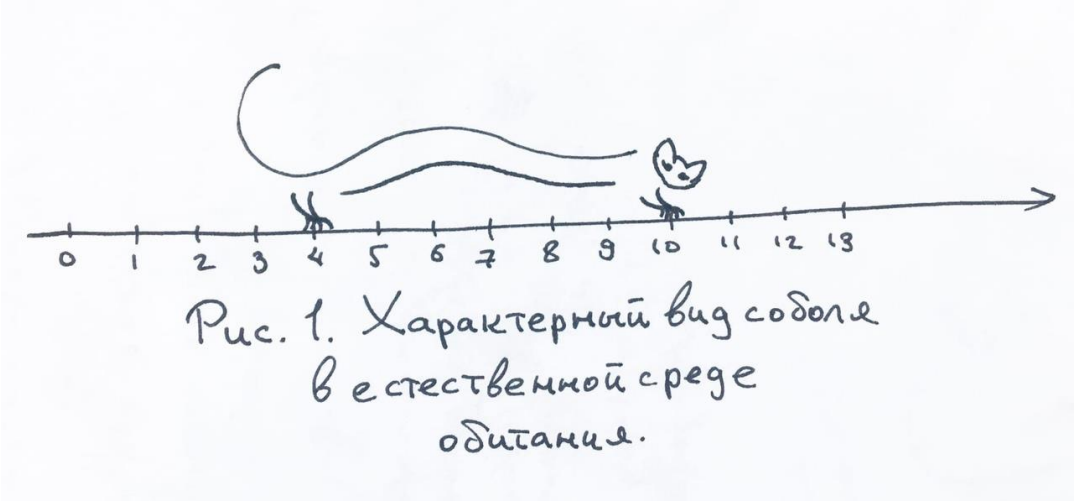

Рис. 1. Характерный вид соболя в естественной среде обитания.

Вам дано начальное положение соболя. Далее задано количество прыжков и изменение положений лапок при них. Определите максимальное расстояние, на котором находились лапки в процессе прыжков.

Начальное положение задаётся координатами задних и передних лапок. Каждый прыжок задаётся изменением координат задних и передних лапок.

На рис. 2 приведен пример.

Начальное положение соболя на нем 4 10.

После этого он совершает 3 прыжка: -6 -9 (оказывается в координатах -2 1, длина 3), 13 8 (после этого он на 11 9, длина 2) и -8 -14 (на 3 -5, длина 8). Наибольшее расстояние между следами передних и задних лапок было обнаружено после последнего прыжка, то есть минимальный размер соболя 8.

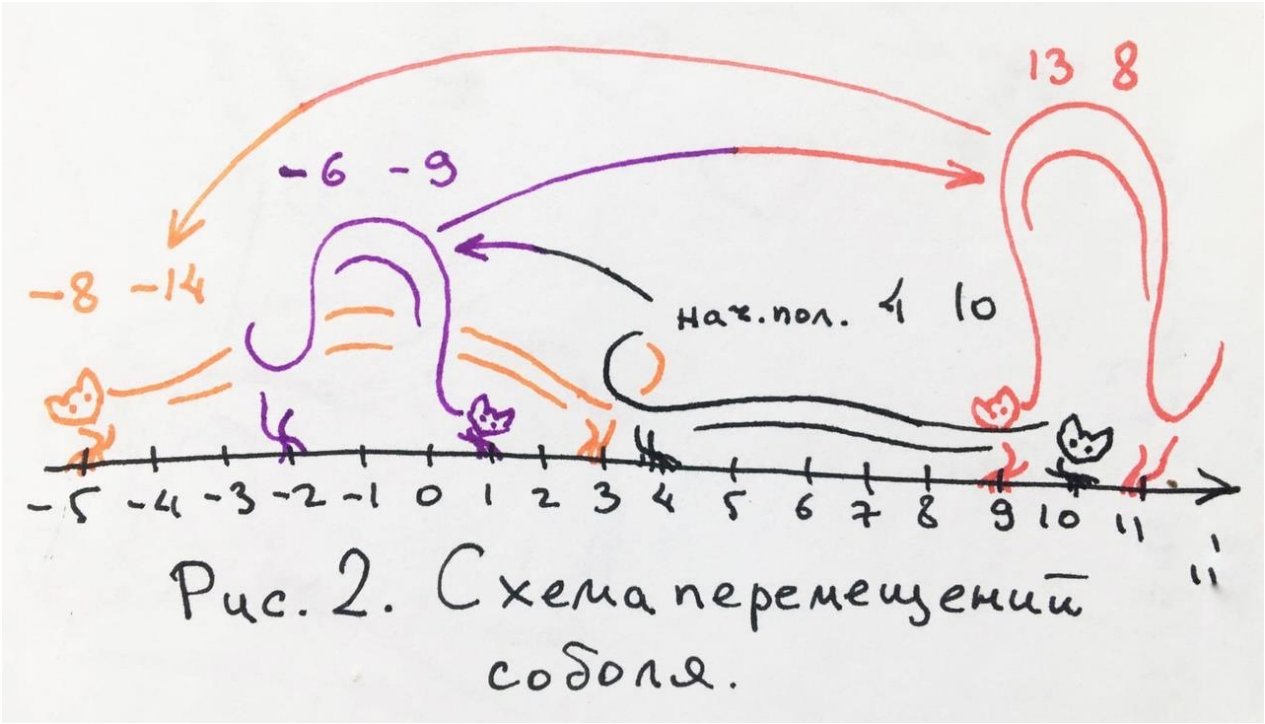

Рис. 2. Схема перемещений соболя.

Формат входных данных

Два целых числа x0 и y0 - начальное положение соболя, x0 - задние лапки, y0 - передние.

N - количество прыжков, последовательность из N пар чисел - изменение координат задних лапок (первое число) и передних (второе число).

Все числа по модулю не превышают 100. N может быть равен нулю (очень ленивый соболь).

Формат выходных данных

Одно целое число - минимальный размер соболя, который мог оставить такие следы.

Например:

Ввод	Результат
4 10 3 -6 -9 13 8 -8 -14	8

Задача 3(Треугольник)

Задан следующий класс точки на плоскости:

```
#include <cmath>

class Point
{
protected:
    int x;
    int y;

public:
    Point(int x, int y): x(x), y(y) {
    }

    Point operator+(const Point& a) const {
        return Point(x + a.x, y + a.y);
    }

    Point operator-(const Point& a) const {
        return Point(x - a.x, y - a.y);
    }

    double dist() const {
        return sqrt(x * x + y * y);
    }
};
```

Логически точка представлена радиус-вектором, поэтому точки можно складывать и вычитать - соответственные операторы для них уже определены. Также у точки есть метод dist() - получить модуль её радиус-вектора.

Напишите класс треугольника со следующим прототипом:

```
class Triangle
{
public:
    // Создать треугольник из трёх точек
    Triangle(const Point& a, const Point& b, const Point& c);

    // Посчитать и вернуть периметр треугольника
    double perimeter() const;
};
```

Для базового тестирования можете использовать следующий пример:

```
int main()
{
    Point a(0, 0), b(0, 5), c(5, 0);
    Triangle t(a, b, c);
    cout << t.perimeter() << endl;
    return 0;
}
```

Например:

Ввод	Результат
0 0 5 0 0 5	Point #1: (0; 0) Point #2: (5; 0) Point #3: (0; 5) Creating triangle Triangle perimeter: 17.0711

Задача 4(Ломаная)

Реализуйте класс, описывающий ломаную линию на плоскости. Прототип класса:

```
class Polyline {
public:
    // Конструктур и деструктор, если необходимы

    // Добавить очередную точку в ломаную
    void addPoint(double x, double y);

    // Получить количесто точек в ломаной в данный момент
    unsigned int getNumberOfPoints() const;

    // Получить длину ломаной в данный момент
    double getLength() const;

    // Получить x-координату i-ой точки ломаной, точки нумеруются в порядке добавления
    // (Гарантируется, что номер i строго меньше, чем то, вернула ваша getNumberOfPoints())
    double getX(unsigned int i) const;

    // Получить y-координату i-ой точки ломаной, точки нумеруются в порядке добавления
    // (Гарантируется, что номер i строго меньше, чем то, вернула ваша getNumberOfPoints())
    double getY(unsigned int i) const;
};
```

Для тестирования можете использовать следующий пример:

```
int main()
{
    Polyline p;
    p.addPoint(0.0, 0.0);
    p.addPoint(1.0, 0.0);
    p.addPoint(1.0, 1.0);
    cout << "Length: " << p.getLength() << endl;
    cout << "Points:" << endl;
    for(unsigned int i = 0; i < p.getNumberOfPoints(); i++) {
        cout << p.getX(i) << " " << p.getY(i) << endl;
    }
    return 0;
}
```

Например:

Ввод	Результат
5 0 0 1 0 1 1 1 4.5 1.5 4.5	Adding point #0: (0; 0) Adding point #1: (1; 0) Adding point #2: (1; 1) Adding point #3: (1; 4.5) Adding point #4: (1.5; 4.5) Polyline reports these points: 0 0 1 0 1 1 1 4.5 1.5 4.5 Polyline length: 6

Задача 5(Дроби)

Реализуйте простой класс для хранения рациональных чисел в виде дроби с целыми числителем и знаменателем. Реализуйте для данного класса базовые операторы для действий с рациональными числами.

Прототип класса:

```
class Rational {
public:
    // Конструктор дроби, здесь a - числитель, b - знаменатель
    Rational(int a, int b);

    // Реализуйте операторы:
    // - сложения двух дробей
    // - вычитания двух дробей
    // - умножения двух дробей
    // - деления двух дробей
    // - умножения дроби на целое число (должно работать при любом порядке операндов)
};
```

Также реализуйте оператор вывода, который должен выводить дробь в формате: a/b.

Замечания:

- нулевой знаменатель отдельно обрабатывать не требуется, такого случая не будет в тестах
- сокращать дроби не требуется (например, если получилось 4/6, то пусть так и остаётся, преобразовывать в 2/3 не нужно - эти нюансы будут обработаны проверяющим кодом)
- данная реализация, разумеется, получается хрупкая и незавершённая, реальный класс для рациональных чисел будет значительно сложнее

Для тестирования можете использовать следующий пример:

```
int main()
{
    Rational a(1, 2);
    Rational b(1, 3);

    cout << a << endl;
    cout << b << endl;
    cout << a + b << endl;
    cout << a - b << endl;
    cout << a * b << endl;
    cout << a / b << endl;
    cout << 3 * a << endl;
    cout << b * 4 << endl;

    return 0;
}
```

Например:

Ввод	Результат
1 2 1 3 3 4	Creating the first rational r1 from 1 and 2 Creating the second rational r2 from 1 and 3 r1: 1/2 r2: 1/3 r1 + r2: 5/6 r1 - r2: 1/6 r1 * r2: 1/6 r1 / r2: 3/2 r1 * 3: 3/2 4 * r2: 4/3

Задача 6(Транспорт)

Создайте иерархию классов для описания различных транспортных средств. Вам дан следующий базовый интерфейс, который должны реализовать все классы:

```
// Абстрактное транспортное средство
class Vehicle {
public:
    // Может ли ездить по суще
    virtual bool canDrive() const {
        return false;
    }

    // Может ли плавать
    virtual bool canSail() const {
        return false;
    }

    // Может ли летать
    virtual bool canFly() const {
        return false;
    }

    virtual ~Vehicle() {};
};
```

Реализуйте следующие классы:

- Car - автомобиль, является Vehicle, может только ездить по суше
- Ship - корабль, является Vehicle, может только плавать
- Plane - самолёт, является Vehicle, может только летать
- Truck - грузовик, является Car, полностью наследует его поведение

Классы должны соответствующим образом перегрузить методы canDrive, canSail, canFly.

Для тестирования можете использовать следующий пример:

```
int main()
{
    cout << boolalpha;
    Vehicle* v;

    v = new Car();
    cout << "Car can drive: " << v->canDrive() << endl;
    cout << "Car can sail: " << v->canSail() << endl;
    cout << "Car can fly: " << v->canFly() << endl;
    delete v;

    v = new Ship();
    cout << "Ship can drive: " << v->canDrive() << endl;
    cout << "Ship can sail: " << v->canSail() << endl;
    cout << "Ship can fly: " << v->canFly() << endl;
    delete v;

    v = new Plane();
    cout << "Plane can drive: " << v->canDrive() << endl;
    cout << "Plane can sail: " << v->canSail() << endl;
    cout << "Plane can fly: " << v->canFly() << endl;
    delete v;

    Car* c = new Truck();
    cout << "Truck can drive: " << c->canDrive() << endl;
    cout << "Truck can sail: " << c->canSail() << endl;
    cout << "Truck can fly: " << c->canFly() << endl;
```

```
    delete c;

    return 0;
}
```

Например:

Ввод	Результат
4 Car Ship Plane Truck	Testing Car Car can drive: true Car can sail: false Car can fly: false Testing Ship Ship can drive: false Ship can sail: true Ship can fly: false Testing Plane Plane can drive: false Plane can sail: false Plane can fly: true Testing Truck Truck can drive: true Truck can sail: false Truck can fly: false Testing if Truck is Car: true

Задача 7(Диспетчер)

Напишите класс диспетчера, управляющего набором модулей. Диспетчер должен уметь регистрировать модули и вызывать их по запросу.

Модулей будет много и разных, но все они реализуют следующий интерфейс:

```
class Module {
public:
    // Получить имя модуля
    virtual string getName() const = 0;

    // Запустить модуль
    virtual void run() = 0;
};
```

На этапе тестирования можете использовать следующие модули:

```
class ModuleA : public Module {
public:
    string getName() const {
        return "moduleA";
    }

    void run() {
        cout << "ModuleA runs" << endl;
    }
};

class ModuleB : public Module {
public:
    string getName() const {
        return "moduleB";
    }

    void run() {
        cout << "ModuleB runs" << endl;
    }
};
```

Прототип требуемого диспетчера:

```
class Dispatcher {
public:
    // Зарегистрировать переданный модуль
    void registerModule(Module* m);

    // Вызвать метод run для модуля с именем moduleName
    // Если такого модуля не зарегистрировано, ничего не делать (но не падать)
    void runModule(const string moduleName) const;
};
```

Для тестирования своей реализации можете использовать следующий пример:

```
int main()
{
    Module* m1 = new ModuleA();
    Module* m2 = new ModuleB();

    Dispatcher d;
    d.registerModule(m1);
    d.registerModule(m2);
    d.runModule("moduleA");
    d.runModule("moduleB");
    d.runModule("moduleC");

    delete m1;
    delete m2;
```

```
    return 0;
}
```

Например:

Ввод	Результат
2 moduleOne moduleTwo 3 moduleTwo moduleOne moduleX	Registering module: moduleOne Registering module: moduleTwo Running module: moduleTwo ModuleTwo runs Running module: moduleOne ModuleOne runs Running module: moduleX

Задача 8(Медианные значения)

На вход программы приходит множество отсчётов значений, снятых с разных датчиков. Для каждого отсчёта доступны строка (имя датчика) и целое число (собственно значение).

После окончания потока значений на вход программы приходят запросы. Каждый запрос содержит имя датчика (могут встречаться имена, которых не было в множестве значений).

Для каждого запроса нужно вывести медианное значение для данного датчика. Если количество отсчётов на датчике чётное - вывести меньшее из двух чисел в середине диапазона. Если такой датчик не встречался совсем - вывести строку "no data".

Справочно: медианное значение - такое значение, которое стоит в середине массива, если массив упорядочен. Не путать со средним значением. Например, для массива [-10; 1; 2; 100; 100000] медианное значение 2. По условию данной задачи для массива чётной длины берётся меньшее из чисел в середине. То есть, например, для массива [0; 2; 10; 200] нужно брать значение 2.

Формат ввода

В первой строке целое число N - количество отсчётов.

Далее N строк, каждая содержит строку (имя датчика), далее пробел, далее целое число (значение). Имя датчика может содержать малые и большие буквы латинского алфавита, а также цифры.

Далее целое число M - количество запросов.

Далее M строк с именами датчиков, для которых запрашиваются медианные значения.

Формат вывода

M строк, каждая содержит или медианное значение для запрошенного датчика, или строку "no data", если такого датчика не было.

Например:

Ввод	Результат
7 sensor1 80 sensor2 35 sensor1 -4 sensor1 8 sensor2 -1 sensor2 15 sensor2 80 3 sensor1 sensor2 sensor3	8 15 no data