

## §3. Заключительный этап: индивидуальная часть

### 3.1 Задачи по математике (9 класс)

#### Задача 3.1.1

*Условие:*

Найти все значения  $x$  и  $y$ , удовлетворяющие равенству  $x - xy + y = 1$

*Решение:*

Данное равенство эквивалентно следующему:  $(x - 1)(y - 1) = 0$ .

*Ответ:*

Одно из чисел равно 1, а другое произвольно.

*Критерии оценки:*

0 баллов - нет ответа, не получено равенство, неправильная логика рассуждений;

1-7 баллов - неправильный ответ, получено неверное эквивалентное равенство, верный ход рассуждений;

8-14 баллов - неправильный ответ, верный ход рассуждений, получено верное эквивалентное равенство; правильный ответ, верный ход рассуждений, не получено равенство

15 баллов - правильный ответ, получено эквивалентное равенство.

#### Задача 3.1.2

*Условие:*

Разработчики переходят из проекта в проект, причем работа в каждом следующем проекте либо интереснее, либо лучше оплачивается.

Одному разработчику повезло - в его новом проекте одновременно более интересные задачи и более высокая зарплата.

Могло ли такое быть, если на одно место в проекте претендует один кандидат? Ответ объясните.

*Решение:*

Пусть есть три проекта — А, Б и В. Пусть по сложности задач проекты расположены в порядке В, А, Б (В — самый сложный), а по зарплате — в порядке Б, В, А (А — самый дорогой). Предположим, что разработчик, работавший в А, переходит в Б (более сложный проект), работавший в Б — в В (более дорогой), работавший в В — в А (одновременно более сложный и более дорогой). Как видим, описанная в задаче ситуация могла случиться.

*Ответ:*

Такое могло быть.

*Критерии оценки:*

0 баллов - нет ответа, неправильная логика рассуждений;

1-6 баллов - правильный ответ, нет объяснений и/или примера; неправильный ответ, есть верные рассуждения

7-13 баллов - правильный ответ, верный пример;

14-19 баллов - правильный ответ, есть верное объяснение, нет примеров;

20 баллов - правильный ответ, есть верное объяснение и примеры.

### Задача 3.1.3

Условие:

Сравните дроби  $\frac{11110}{11111}$ ,  $\frac{33331}{33334}$ ,  $\frac{44441}{44445}$  и расположите их в порядке возрастания.

Решение:

Рассмотрим числа

$$1-x=1/11111,$$

$$1-y=3/33334,$$

$$1-z=4/44445,$$

а также обратные к ним

$$1/(1-x)=11111,$$

$$1/(1-y)=11111+1/3,$$

$$1/(1-z)=11111+1/4.$$

Мы видим, что  $1/(1-x) < 1/(1-z) < 1/(1-y)$ .

Поскольку все рассматриваемые числа положительны,  $1-x > 1-z > 1-y$ . Следовательно,  $x < z < y$ .

Ответ:

$$\frac{11110}{11111}, \frac{44441}{44445}, \frac{33331}{33334}$$

Критерии оценки:

0 баллов - нет ответа/неправильный ответ, неправильная логика рассуждений;

1-10 балл - правильный ответ, нет объяснения или неправильная логика рассуждений;

11-20 балла - нет ответа/неправильный ответ, правильная логика рассуждений, решение не доведено до конца;

21-29 - правильный ответ, решение не доведено до конца или допущена арифметическая ошибка;

30 баллов - правильный ответ, верная логика рассуждений, решение доведено до конца.

### Задача 3.1.4

Условие:

В городе 15 остановок, некоторые из них соединены маршрутами общественного транспорта: автобусными, трамвайными и троллейбусными.

Маршруты разработаны так, что в выходной для автобусного депо день, попасть с любой остановки на любую другую остановку (возможно, с пересадками) можно на трамвае и/или троллейбусе. Аналогично для выходных в трамвайном и троллейбусном депо.

Какое наименьшее число маршрутов может быть в городе?

Решение:

Найдем минимальное число маршрутов двух видов транспорта.

Обозначим число маршрутов через  $n$ . Если мы уберем один из них, транспортная сеть города может распасться на две разрозненные сети, но может и не распасться. Если убрать еще один маршрут, не более одной транспортной сети может разделиться ещё на две, количество сетей увеличится не более чем на одну. Продолжая аналогичные рассуждения, получим, что после отмены последнего маршрута количество транспортных сетей не может превысить  $n+1$ . Поскольку 15 остановок без маршрутов между ними - это 15 разрозненных сетей, то  $n \geq 14$  (1).

Пусть количество маршрутов автобусов, трамваев и троллейбусов будет  $a$ ,  $b$  и  $c$ . Исходя из найденного минимального числа маршрутов двух видов транспорта  $n$ , получаем  $a+b \geq 14$ ,  $b+c \geq 14$ ,  $c+a \geq 14$ . Складывая эти неравенства, получаем:  $2(a+b+c) \geq 42$ , т. е. у трёх видов транспорта в сумме как минимум 21 маршрут.

*Ответ:*  
21 маршрут.

*Критерии оценки:*

- 0 баллов - нет ответа/неправильный ответ, неправильная логика рассуждений;
- 1-10 баллов - правильный ответ, нет объяснения или неправильная логика рассуждений;
- 11-20 баллов - нет ответа/неправильный ответ, правильная логика рассуждений, решение не доведено до конца;
- 21-30 баллов - верная логика рассуждений, найдено минимальное число маршрутов двух видов транспорта, решение не доведено до конца, то есть не получена система неравенств или получена неправильная;
- 31-34 балла - верная логика рассуждений, найдено минимальное число маршрутов двух видов транспорта, получена система неравенств, допущена арифметическая ошибка в решении системы неравенств;
- 35 баллов - правильный ответ, верная логика рассуждений, найдено минимальное число маршрутов двух видов транспорта, получена система неравенств.

## 3.2 Задачи по математике (10-11 класс)

### Задача 3.2.1

*Условие:*

Решить систему уравнений:

$$\begin{aligned}x + y + z &= -1 \\x^2 + y^2 + z^2 &= 1 \\x^3 + y^3 + z^3 &= -1\end{aligned}$$

*Решение:*

Возводим (1) в квадрат и вычитаем (2), получаем:

$$(x + y + z)^2 - (x^2 + y^2 + z^2) = 2(xy + yz + xz) \Rightarrow xy + xz + yz = 0(\bullet)$$

Возводим (1) в куб и вычитаем (3), получаем:

$$(x + y + z)^3 - (x^3 + y^3 + z^3) = 3(x + y)(x + z)(y + z) \Rightarrow 3xyz = 0(\bullet\bullet)$$

Анализируем ( $\bullet\bullet$ ):

Пусть  $x = 0$ , то выражение ( $\bullet$ ) показывает, что  $yz = 0$ . Поэтому либо  $y = 0$  и  $z = -1$ , либо  $z = 0$  и  $y = -1$ .

Аналогично для  $y=0$  и  $z=0$ . В результате получаем:  $(0, 0, -1)$ ,  $(0, -1, 0)$  и  $(-1, 0, 0)$ .

*Ответ:*

$(-1, 0, 0)$ ,  $(0, -1, 0)$ ,  $(0, 0, -1)$

*Критерии оценки:*

- 0 баллов - нет ответа/неправильный ответ, неправильная логика рассуждений;
- 1-7 баллов - неправильный ответ, получено неверное эквивалентное равенство, верный ход рассуждений; правильный ответ, но нет решения
- 8-14 баллов - правильная логика рассуждений, решение не доведено до конца или допущена арифметическая ошибка
- 15 баллов - правильный ответ, верная логика рассуждений, решение доведено до конца.

### Задача 3.1.1

*Условие:*

Разработчики переходят из проекта в проект, причем работа в каждом следующем проекте либо интереснее, либо лучше оплачивается.

Одному разработчику повезло - в его новом проекте одновременно более интересные задачи и более высокая зарплата.

Могло ли такое быть, если на одно место в проекте претендует один кандидат? Ответ объясните.

*Решение:*

Пусть есть три проекта — А, Б и В. Пусть по сложности задач проекты расположены в порядке В, А, Б (Б — самый сложный), а по зарплате — в порядке Б, В, А (А — самый дорогой). Предположим, что разработчик, работавший в А, переходит в Б (более сложный проект), работавший в Б — в В (более дорогой), работавший в В — в А (одновременно более сложный и более дорогой). Как видим, описанная в задаче ситуация могла случиться.

*Ответ:*

Такое могло быть.

*Критерии оценки:*

0 баллов - нет ответа, неправильная логика рассуждений;

1-6 баллов - правильный ответ, нет объяснений и/или примера;

7-13 баллов - правильный ответ, верный пример;

14-19 баллов - правильный ответ, есть верное объяснение, нет примеров;

20 баллов - правильный ответ, есть верное объяснение и примеры.

### Задача 3.1.1

*Условие:*

Три стартапера выбирают место для работы в своем городе. Один из них планирует добираться до работы пешком, второй - ехать на велосипеде, третий - на автомобиле. Их скорости соотносятся как 1:2:3. Какое место выбрать стартаперам, чтобы суммарное время, затрачиваемое на дорогу всеми участниками, было наименьшим?

Все трое живут не на одной улице. Каждый выбирает кратчайший путь до места встречи.

*Решение:*

Этим местом для работы является дом первого стартапера, планирующего ходить на работу пешком. Для доказательства этого обозначим искомое место для работы буквой А, а дома стартаперов - цифрами 1, 2, 3 в соответствии с соотношением скоростей. Расстояния между домом 1 и домами 2 и 3 обозначим через а и b, а расстояния от точки А до домов 1, 2, 3 — через x, y и z соответственно. Тогда

$$\frac{a}{2} \leq \frac{x}{2} + \frac{y}{2}, \frac{b}{3} \leq \frac{x}{3} + \frac{z}{3},$$

откуда

$\frac{a}{2} + \frac{b}{3} \leq \frac{x}{2} + \frac{y}{2} + \frac{x}{3} + \frac{z}{3} \leq \frac{y}{2} + \frac{z}{3}$ , причём равенство достигается при  $x = 0$ , т. е. когда точка А совпадает с домом 1.

*Ответ:*

Дома у того, кто планирует добираться до работы пешком.

*Критерии оценки:*

0 баллов - нет ответа/неправильный ответ, неправильная логика рассуждений;

1-10 балл - правильный ответ, нет объяснения или неправильная логика рассуждений;  
11-20 балла - нет ответа/неправильный ответ, правильная логика рассуждений, решение не доведено до конца;  
21-29 - правильный ответ, решение не доведено до конца или допущена арифметическая ошибка;  
30 баллов - правильный ответ, верная логика рассуждений, решение доведено до конца.

### **Задача 3.1.1**

*Условие:*

Компьютеры компании объединены в несколько сетей, при этом: 1) любой компьютер может напрямую обратиться к любому другому компьютеру; 2) для любой пары сетей найдется, и притом единственное, общее устройство; 3) в каждой сети ровно три компьютера. Сколько всего сетей?

*Решение:*

Докажем, что если выполняются указанные условия, то число устройств  $n$  и число сетей  $N$  связаны соотношением  $N = n(n - 1) + 1$ . Пусть  $a$  — одна из сетей,  $B$  — компьютер, не подключенный к сети  $a$ . Каждая сеть, к которой подключен  $B$ , имеет общее устройство с сетью  $a$ . Поэтому  $B$  подключен ровно к  $n$  сетям. Аналогично доказывается, что через каждый компьютер сети  $a$  подключен к  $n - 1$  сети, отличной от  $a$ . Всего получаем  $n(n - 1)$  разных сетей и ещё сама  $a$ . Итого  $n^2$  сетей

*Ответ:*

7

*Критерии оценки:*

0 баллов - нет ответа/неправильный ответ, неправильная логика рассуждений;  
1-10 баллов - правильный ответ, нет объяснения или неправильная логика рассуждений;  
11-20 баллов - нет ответа/неправильный ответ, правильная логика рассуждений, решение не доведено до конца;  
21-30 баллов - верная логика рассуждений, решение не доведено до конца, то есть не получено соотношение между числом устройств и числом сетей или получено неправильное;  
31-34 балла - верная логика рассуждений, получено соотношение между числом устройств и числом сетей, допущена арифметическая ошибка;  
35 баллов - правильный ответ, верная логика рассуждений, получено соотношение между числом устройств, получена система неравенств.

## **3.2 Задачи по информатике (9-11 классы)**

### **Задача 3.2.1**

*Условие:*

Ограничение по времени: 1 секунда

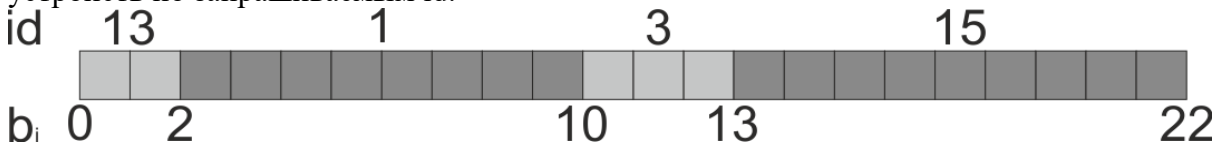
Ограничение по памяти: 32 мегабайта

При управлении сложными системами центральному серверу постоянно приходится отслеживать состояние множества компонент — будь то датчики, исполнительные устройства или другое оборудование. Рассмотрим систему, состоящую из  $N$  различных устройств. Каждое из этих устройств имеет свой уникальный целочисленный идентификатор  $id$ . Для хранения информации об  $i$ -ом устройстве в памяти центрального сервера выделяется  $w_i$  байт. Память под все эти устройства выделяется одним целым блоком, размера  $W = \sum w_i$ .

В любой момент времени серверу известен адрес в памяти на начало этого блока. Отступом  $i$ -го устройства будем называть количество байт  $b_i$ , которое нужно прибавить к адресу начала блока, чтобы получить адрес  $i$ -го устройства. Отступ первого устройства  $b_1$  всегда равен 0. Отступ каждого последующего устройства рассчитывается по формуле:

$$b_i = b_{i-1} + w_{i-1}$$

Каждый раз, когда серверу необходимо обратиться к устройству с определенным  $id$ , ему необходимо определить место в памяти, где храниться информация об этом устройстве. Ваша задача написать программу, которая будет рассчитывать отступ для устройств по запрашиваемым  $id$ .



**Формат входных данных:**

В первой строке даны два целых числа:  $N$  – количество различных устройств, и  $K$  – количество запросов ( $2 \leq N \leq 103$ ,  $1 \leq K \leq 103$ ).

В следующих  $N$  строках заданы по два целых числа:  $id_i$  и  $w_i$  – идентификатор и объём выделяемой памяти для  $i$ -го устройства соответственно ( $1 \leq id_i \leq 105$ ,  $1 \leq w_i \leq 105$ ). Устройства даны в том порядке, в котором они хранятся в выделенном блоке памяти.

Следующие  $K$  строк содержат по единственному целому числу  $id_k$  – идентификатор устройства, для которого необходимо посчитать отступ  $b_k$ .

**Формат выходных данных:**

Для каждого из  $K$  запросов выведите число  $b_k$  в отдельной строке.

Sample Input:

```
4 6
13 2
1 8
3 3
15 9
1
13
3
13
1
15
```

Sample Output:

```
2
0
10
0
2
13
```

```
import operator

def generate():
    tests = []
    return tests

def solve(dataset):
```

```

input = dataset.split();

cnt = 0;
n = int(input[cnt]);
cnt += 1;
k = int(input[cnt]);
cnt += 1;

devs = [];
w_cum = 0;
for i in range(n):
    dev = {'id':0, 'w_cum':w_cum};
    if i > 0:
        dev['w_cum'] += devs[i - 1]['w_cum'];
    dev['id'] = int(input[cnt]);
    cnt += 1;
    w_cum = int(input[cnt]);
    cnt += 1;
    devs.append(dev);

devs.sort(key=operator.itemgetter('id'));

res = "";
for i in range(k):
    id = int(input[cnt]);
    cnt += 1;
    idx = bin_search(devs, id);
    res += str(devs[idx]['w_cum']) + '\n';

return res

def bin_search(ar, x):
    l = 0;
    r = len(ar) - 1;
    while (l <= r):
        m = int((l + r) / 2);
        if (ar[m]['id'] == x):
            return m;
        elif (ar[m]['id'] < x):
            l = m + 1;
        else:
            r = m - 1;

    return -1;

def check(reply, clue):
    reply = [str(i) for i in reply.strip().split()]
    clue = [str(i) for i in clue.strip().split()]
    return reply == clue

—

import operator

def generate():
    tests = []
    return tests

```

```

def solve(dataset):
    input = dataset.split();

    cnt = 0;
    n = int(input[cnt]);
    cnt += 1;
    k = int(input[cnt]);
    cnt += 1;

    devs = [];
    w_cum = 0;
    for i in range(n):
        dev = {'id':0, 'w_cum':w_cum};
        if i > 0:
            dev['w_cum'] += devs[i - 1]['w_cum'];
        dev['id'] = int(input[cnt]);
        cnt += 1;
        w_cum = int(input[cnt]);
        cnt += 1;
        devs.append(dev);

    devs.sort(key=operator.itemgetter('id'));

    res = "";
    for i in range(k):
        id = int(input[cnt]);
        cnt += 1;
        idx = bin_search(devs, id);
        res += str(devs[idx]['w_cum']) + '\n';

    return res

def bin_search(ar, x):
    l = 0;
    r = len(ar) - 1;
    while (l <= r):
        m = int((l + r) / 2);
        if (ar[m]['id'] == x):
            return m;
        elif (ar[m]['id'] < x):
            l = m + 1;
        else:
            r = m - 1;

    return -1;

def check(reply, clue):
    reply = [str(i) for i in reply.strip().split()]
    clue = [str(i) for i in clue.strip().split()]
    return reply == clue

```

### **Задача 3.2.2**

*Условие:*

Ограничение по времени: 1 секунда

Ограничение по памяти: 32 мегабайта

В системе управления иногда возникают простои, когда никаких важных задач выполнять не надо. Это свободное время используется для выполнения некоторых



второстепенных задач. Время, которое система гарантировано может потратить на второстепенные задачи, всегда известно и равно  $TT$ . Всего возможных второстепенных задач  $NN$ . Для каждой из них известно время выполнения  $tt$  и некоторая целочисленная величина  $bb$ , характеризующая пользу, которую несет с собой выполнение этой задачи.

Вам необходимо по известному списку задач определить, какую максимальную суммарную пользу может принести система, оптимальным образом выбрав задачи, которые можно выполнить за суммарное время, не превосходящее  $TT$ .

Польза от выполнения задачи засчитывается только для полностью завершенных задач. Задачи выполняются последовательно, после завершения одной – сразу начинается выполнение второй. Каждая задача может быть выполнена только один раз.

**Формат входных данных:**

В первой строке даны два целых числа:  $NN$  и  $TT$  – количество доступных задач и время на выполнение второстепенных задач ( $1 \leq N \leq 20$ ,  $1 \leq T \leq 103$ ).

В следующих  $NN$  строках заданы по два целых числа:  $t_i$  и  $b_i$  – время выполнения и характеристика пользы  $i$ -ой задачи ( $1 \leq t_i \leq 103$ ,  $1 \leq b_i \leq 105$ ).

**Формат выходных данных:**

В ответ выведите единственное число  $BB$  – максимальную возможную суммарную пользу, которую можно получить, выполнив оптимальный набор задач из списка за суммарное время, не превосходящее  $TT$ .

Sample Input:

```
5 7
2 8
3 10
3 8
5 5
3 7
```

Sample Output:

```
18
```

```
def generate():
    tests = []
    return tests

def solve(dataset):
    input = dataset.split()

    cnt = 0;
    n = int(input[cnt]);
    cnt += 1;
    T = int(input[cnt]);
    cnt += 1;

    t = [];
    b = [];
    for i in range(n):
        t.append(int(input[cnt]));
        cnt += 1;
        b.append(int(input[cnt]));
        cnt += 1;
    dp = [[0 for i in range(T + 1)] for i in range(n + 1)];
    for i in range(1, n + 1):
        for j in range(T + 1):
            if (t[i - 1] > j):
                dp[i][j] = dp[i - 1][j];
            else:
```

```

        dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - t[i -
1]] + b[i - 1]);

    res = str(dp[n][T]) + '\n';

    return res

```

```

def check(reply, clue):
    reply = [str(i) for i in reply.strip().split()]
    clue = [str(i) for i in clue.strip().split()]
    return reply == clue

```

### **Задача 3.2.3**

*Условие:*

Ограничение по времени: 1 секунда

Ограничение по памяти: 32 мегабайта

В данной задаче рассматривается история выполнения различных операций некой системы управления за некоторый промежуток времени. Известно, что за рассматриваемое время система успела  $NN$  раз выполнить определенные операции. Большинство из них были выполнены четное число раз, но  $KK$  операций были выполнены нечетное количество раз.

Каждая операция имеет свой целочисленный идентификационный номер  $idid$ .

Ваша задача по истории выполненных операций определить  $KK$  идентификационных номеров  $idkidk$  тех операций, которые выполнились нечетное количество раз.

#### **Формат входных данных:**

В первой строке даны два целых числа  $NN$  и  $KK$  – общее количество выполненных за день операций и количество операций, выполненных нечетное число раз соответственно ( $2 \leq N \leq 107$ ,  $2 \leq K \leq 107$ ,  $K=1$ ).

В следующих  $NN$  строках содержится по одному целому числу  $idid$  – идентификатор выполненной операции ( $1 \leq idid \leq 109$ ).

#### **Формат выходных данных:**

В единственной строке через пробел выведите  $KK$  чисел  $idkidk$  в порядке возрастания – идентификаторы операций, выполненных нечетное число раз.

Sample Input:

```

9 1
7 3 5 5 3 7 2 2 7

```

Sample Output:

```

7

```

```

def generate():
    tests = []
    return tests

def get_next(dataset, ptr, slen):
    while (ptr < slen and not((dataset[ptr] >= '0' and dataset[ptr]
<= '9') or dataset[ptr] == '-')):
        ptr += 1;
    if ptr == slen:
        return 0;
    mult = 1;
    if (dataset[ptr] == '-'):

```

```

        mult = -1;
        ptr += 1;

    n = 0;
    while (dataset[ptr] >= '0' and dataset[ptr] <= '9'):
        n = n * 10 + int(dataset[ptr]);
        ptr += 1;

    return n, ptr + 1

def solve(dataset):
    ptr = 0;
    slen = len(dataset);

    n, ptr = get_next(dataset, ptr, slen)
    k, ptr = get_next(dataset, ptr, slen)

    res = "";
    if (k == 1):
        res = sol_3_1(n, dataset, ptr, slen);
    elif (k == 2):
        res = sol_3_2(n, dataset, ptr, slen);

    res += '\n';

    return res

def sol_3_1(n, dataset, ptr, slen):
    x = 0;
    for i in range(n):
        tmp, ptr = get_next(dataset, ptr, slen);
        x ^= tmp;

    return str(x);

def sol_3_2(n, dataset, ptr, slen):
    x = 0;
    ar = [0] * 31;
    for i in range(n):
        tmp, ptr = get_next(dataset, ptr, slen);
        x ^= tmp;
        for shift in range(31):
            if (tmp & (1 << shift)):
                ar[shift] ^= tmp;

    a = -1;
    b = -1;
    for i in range(31):
        if (ar[i] != 0):
            if (a == -1):
                a = ar[i];
            elif (a != ar[i]):
                b = ar[i];
                break;

    if (a == x):
        a ^= b;

```

```

elif (b == x):
    b ^= a;
if (b < a):
    a, b = b, a;

return str(a) + ' ' + str(b);

def check(reply, clue):
    reply = [str(i) for i in reply.strip().split()]
    clue = [str(i) for i in clue.strip().split()]
    return reply == clue

```

### Задача 3.2.4

*Условие:*

Ограничение по времени: 1 секунда

Ограничение по памяти: 32 мегабайта

Умный беспилотный робот пылесос, убирающий грязь с пола автономного космического транспорта, всегда передвигается по заданной последовательности из  $NN$  векторов. Сначала пылесос находится в точке  $(0;0)$ , соответствующей положению его док-станции. Первым шагом он перемещается по вектору  $v_1$ . Его координаты становятся равны  $(v_{1,x}; v_{1,y})$ . Отсюда он следует по вектору  $v_2$  в точку  $(v_{1,x}+v_{2,x}; v_{1,y}+v_{2,y})$ . Далее по вектору  $v_3$  и т.д. до вектора  $v_N$ . После этого робот возвращается на док-станцию. «Почему же этот пылесос называется умным, если ездит он исключительно по заданной последовательности векторов?» – удивитесь вы. Оказывается, когда робот собирается начать движение по очередному вектору, он может выбрать двигаться ему по вектору  $v_i$  или по вектору  $-v_i$  (в противоположном направлении). Известно, что длина всех векторов  $v_i$  не превосходит  $D$ .

Переместившись по всем доступным векторам умный пылесос должен иметь связь (конечно же, беспроводную) со своей док-станцией, чтобы по окончании обхода знать, куда вернуться на зарядку. Для обеспечения связи между обоими устройствами, пылесос должен находиться на расстоянии не большем чем  $2\sqrt{D^2 \cdot D}$  от док-станции.

Ваша задача, написать программу, которая должна определить направление движения пылесоса вдоль каждого вектора таким образом, чтобы в конце пути он не отделился от док-станции с координатами  $(0;0)$ , на расстояние большее чем  $2\sqrt{D^2 \cdot D}$ . Обратите внимание, что это условие должно выполняться только после прохода по всем  $NN$  векторам. Промежуточные координаты пылесоса могут превосходить эту величину.

#### Формат входных данных:

В первой строке даны два целых числа: количество векторов  $NN$ , и расстояние  $DD$  ( $2 \leq N \leq 104$ ,  $1 \leq D \leq 100$ ).

В следующих  $NN$  строках содержится по два целых числа  $xx$  и  $yy$  – координаты очередного вектора ( $0 \leq x^2 + y^2 \leq D^2$ ).

#### Формат выходных данных:

Если выбрать правильную последовательность направлений векторов таким образом, чтобы пылесос всегда был на связи с док-станцией невозможно, выведите единственное слово “Impossible” (без кавычек).

Если найти правильную последовательность направлений возможно, в первой строке выведите слово “Possible” (без кавычек), а во второй строке последовательность символов “F” и “B” (без кавычек) по следующему правилу. Если для очередного вектора выбрано направление, совпадающее с направлением вектора, выводится символ “F” (от англ. – Forward). Если же выбрано направление противоположное направлению вектора, выводится символ “B” (от англ. – Backward).

Sample Input:

```
5 7
0 7
-7 0
1 -1
-1 1
-6 3
```

Sample Output:

```
Possible
FFFFB
```

```
import sys
```

```
sys.setrecursionlimit(20000)
```

```
def generate():
```

```
    tests = []
    return tests
```

```
def get_next(dataset, ptr, slen):
```

```
    while (ptr < slen and not((dataset[ptr] >= '0' and dataset[ptr]
<= '9') or dataset[ptr] == '-')):
```

```
        ptr += 1;
```

```
    if ptr == slen:
```

```
        return 0;
```

```
    mult = 1;
```

```
    if (dataset[ptr] == '-');
```

```
        mult = -1;
```

```
        ptr += 1;
```

```
    n = 0;
```

```
    while (ptr < slen and dataset[ptr] >= '0' and dataset[ptr] <=
'9');
```

```
        n = n * 10 + int(dataset[ptr]);
```

```
        ptr += 1;
```

```
    return n * mult, ptr + 1
```

```
def sum_vec(a, b):
```

```
    return {'x' : a['x'] + b['x'], 'y' : a['y'] + b['y']};
```

```
def dif_vec(a, b):
```

```
    return {'x' : a['x'] - b['x'], 'y' : a['y'] - b['y']};
```

```
def len_vec(a):
```

```
    return a['x'] * a['x'] + a['y'] * a['y'];
```

```
def node_init():
```

```
    return {
```

```
        'r_sign' : True,
```

```
        'left' : None,
```

```
        'right' : None,
```

```
        'vec' : { 'x' : 0, 'y' : 0 },
```

```
        'idx' : 0
```

```
    }
```

```

signs = [];
sign = False;

def dfs(nd):
    global signs;
    global sign;
    if (nd['left'] == None and nd['right'] == None):
        signs[nd['idx']] = sign;
    if (nd['left'] != None):
        dfs(nd['left']);
    if (nd['right'] != None):
        old_sign = sign;
        if (not sign):
            sign = not nd['r_sign'];
        else:
            sign = nd['r_sign'];
        dfs(nd['right']);
        sign = old_sign;

def solve(dataset):
    global sign;
    global signs;

    ptr = 0;
    slen = len(dataset);

    n, ptr = get_next(dataset, ptr, slen);
    l, ptr = get_next(dataset, ptr, slen);

    signs = [0] * n;

    maxl = l * l * 2;
    l *= l;

    tree = [];
    tree_org = [];

    for i in range(n):
        vx, ptr = get_next(dataset, ptr, slen);
        vy, ptr = get_next(dataset, ptr, slen);
        nd = node_init();
        nd['vec']['x'] = vx;
        nd['vec']['y'] = vy;
        nd['idx'] = i;
        tree.append(nd);
        tree_org.append(nd);

    while (len(tree) > 2):
        tree_tmp = [];
        len_tree = len(tree);
        for i in range(len_tree - 1):
            nd_cur = tree[i];
            for j in range(i + 1, len_tree):
                nd_next = tree[j];
                if len_vec(sum_vec(nd_cur['vec'],
nd_next['vec'])) <= 1:
                    nd = node_init();

```

```

        nd['vec'] = sum_vec(nd_cur['vec'],
nd_next['vec']));
        nd['left'] = nd_cur;
        nd['right'] = nd_next;
        nd['r_sign'] = True;
        nd_cur = nd;
        elif len_vec(dif_vec(nd_cur['vec'],
nd_next['vec'])) <= 1:
        nd = node_init();
        nd['vec'] = dif_vec(nd_cur['vec'],
nd_next['vec']));
        nd['left'] = nd_cur;
        nd['right'] = nd_next;
        nd['r_sign'] = False;
        nd_cur = nd;
        else:
            tree_tmp.append(nd_next);
            tree_tmp.append(nd_cur);
            if len(tree_tmp) != len(tree):
                break;

        tree, tree_tmp = tree_tmp, tree;

root = {};
if (len(tree) > 1):
    nd_cur = tree[0];
    nd_next = tree[1];

    if len_vec(sum_vec(nd_cur['vec'], nd_next['vec'])) <=
len_vec(dif_vec(nd_cur['vec'], nd_next['vec'])):
        root['vec'] = sum_vec(nd_cur['vec'], nd_next['vec']);
        root['left'] = nd_cur;
        root['right'] = nd_next;
        root['r_sign'] = True;
    else:
        root['vec'] = dif_vec(nd_cur['vec'], nd_next['vec']);
        root['left'] = nd_cur;
        root['right'] = nd_next;
        root['r_sign'] = False;
else:
    root = tree[0];

sign = True;
dfs(root);

res = "Possible\n";
for i in range(n):
    if (signs[i]):
        res += 'F';
    else:
        res += 'B';
res += '\n';

return res

def check(reply, clue):
    reply = [str(i) for i in reply.strip().split()]
    clue_ = [str(i) for i in clue.strip().split()]

```

```

    if (clue_[0] != 'HACK'):
        clue = ''5 7
0 7
-7 0
1 -1
-1 1
-6 3
'''

    if (reply[0] != 'Possible' or len(reply) != 2):
        return False;

    dirs = reply[1];

    ptr = 0;
    slen = len(clue);

    n, ptr = get_next(clue, ptr, slen);
    l, ptr = get_next(clue, ptr, slen);
    maxl = l * l * 2;

    if len(dirs) != n:
        return False;

    x = 0;
    y = 0;

    for i in range(n):
        if (dirs[i] != 'F' and dirs[i] != 'B'):
            return False;
        x_cur, ptr = get_next(clue, ptr, slen);
        y_cur, ptr = get_next(clue, ptr, slen);
        if (dirs[i] == 'F'):
            x += x_cur;
            y += y_cur;
        else:
            x -= x_cur;
            y -= y_cur;

    return x * x + y * y <= maxl;

```