

А. Подарок другу

Решение на Python (100 баллов)

На языке Python задача решается тривиально, так как встроенный тип `int` поддерживает целые числа произвольной длины, включая диапазон до 10^{18} и выше. Достаточно считать два числа и вывести их сумму.

```
A = int(input())
B = int(input())
print(A + B)
```

Далее рассмотрим особенности реализации на языке C++ в зависимости от подгрупп тестов.

Подгруппа 1 ($1 \leq A, B \leq 10^9$)

В этой подгруппе сумма $A + B$ не превышает $2 \cdot 10^9$, что полностью помещается в 32-битный знаковый тип `int` (максимальное значение 2 147 483 647). Достаточно использовать `int` для всех переменных.

Пример кода на C++:

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

int main() {
    int A, B;
    cin >> A >> B;
    cout << A + B << endl;
    return 0;
}
```

Подгруппа 2 ($1 \leq A, B \leq 10^{18}$)

В этой подгруппе значения A и B могут достигать 10^{18} , поэтому их сумма может быть до $2 \cdot 10^{18}$. Тип `int` (32 бита) переполнится, так как его максимум около $2 \cdot 10^9$. Необходимо использовать 64-битный тип `long long`, который может хранить числа до $9 \cdot 10^{18}$.

Пример кода на C++:

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

int main() {
    long long A, B;
    cin >> A >> B;
    cout << A + B << endl;
    return 0;
}
```

В. Гонка

Подгруппа 1 ($T \leq t_1$ и $T \leq t_2$)

В этой подгруппе оба гонщика не успевают изменить скорость, так как момент фиксации T наступает раньше или в момент переключения каждого из них. Следовательно, каждый движется только с начальной скоростью.

Путь Алисы: $S_A = v_1 \cdot T$

Путь Боба: $S_B = v_2 \cdot T$

Сравниваем эти величины и выводим «Alice», «Bob» или «Draw».

Пример кода на Python:

```
v1, t1, f1 = map(int, input().split())
v2, t2, f2 = map(int, input().split())
T = int(input())

s1 = v1 * T
s2 = v2 * T

if s1 > s2:
    print("Alice")
elif s1 < s2:
    print("Bob")
else:
    print("Draw")
```

Подгруппа 2 ($v_1 = v_2$ и $t_1 = t_2$)

Здесь начальные скорости и моменты переключения одинаковы. До момента $t_1 = t_2$ они едут вместе, поэтому их пути равны. После переключения их скорости становятся f_1 и f_2 .

Рассмотрим два случая:

- Случай 1: $T \leq t_1$. Оба не успели переключиться → пути равны → ничья.
- Случай 2: $T > t_1$. До момента переключения они прошли одинаковое расстояние. После переключения Алиса едет со скоростью f_1 , Боб — со скоростью f_2 . Если $f_1 > f_2$, то за оставшееся время Алиса проедет больше и окажется впереди. Если $f_1 < f_2$ — впереди Боб. Если равны — остаются вместе. Таким образом, результат определяется только сравнением f_1 и f_2 .

Пример кода на Python:

```

v1, t1, f1 = map(int, input().split())
v2, t2, f2 = map(int, input().split())
T = int(input())

if T <= t1:
    print("Draw")
else:
    if f1 > f2:
        print("Alice")
    elif f1 < f2:
        print("Bob")
    else:
        print("Draw")

```

Подгруппа 3 ($1 \leq t_1, t_2, T \leq 10$)

Ограничения очень малы. Можно просто смоделировать движение по секундам: на каждой секунде гонщик движется с той скоростью, которая действует в эту секунду. Так как $T \leq 10$, цикл выполнится не более 10 раз.

Правило определения текущей скорости: если текущая секунда не больше t_1 , скорость Алисы равна v_1 , иначе она равна f_1 (аналогично для Боба).

Пример кода на Python:

```

v1, t1, f1 = map(int, input().split())
v2, t2, f2 = map(int, input().split())
T = int(input())

s1 = 0
s2 = 0
for sec in range(1, T + 1):
    if sec <= t1:
        s1 += v1
    else:
        s1 += f1
    if sec <= t2:
        s2 += v2
    else:
        s2 += f2

if s1 > s2:
    print("Alice")
elif s1 < s2:
    print("Bob")
else:
    print("Draw")

```

Подгруппа 4 (полные ограничения: $1 \leq v, t, f, T \leq 10^9$)

Здесь T может быть очень большим, поэтому нельзя использовать цикл. Нужно вычислить пройденный путь по формуле.

Как устроен путь одного гонщика? До момента t он едет со скоростью v . Если T не больше t , то он всё время ехал со скоростью v , и его путь равен $v \cdot T$.

Если же T больше t , то первые t секунд он ехал со скоростью v и проехал $v \cdot t$ метров. После этого у него остаётся $(T - t)$ секунд, и он едет со скоростью f . За это время он проезжает $f \cdot (T - t)$ метров. Общий путь получается сложением: $v \cdot t + f \cdot (T - t)$.

Таким образом, для каждого гонщика вычисляем путь по правилу:

- Если $T \leq t$, то путь равен $v \cdot T$.
- Если $T > t$, то путь равен $v \cdot t + f \cdot (T - t)$.

После этого сравниваем два числа и выводим результат.

Пример кода на Python:

```
v1, t1, f1 = map(int, input().split())
v2, t2, f2 = map(int, input().split())
T = int(input())

# Путь Алисы
if T <= t1:
    s1 = v1 * T
else:
    s1 = v1 * t1 + f1 * (T - t1)

# Путь Боба
if T <= t2:
    s2 = v2 * T
else:
    s2 = v2 * t2 + f2 * (T - t2)

if s1 > s2:
    print("Alice")
elif s1 < s2:
    print("Bob")
else:
    print("Draw")
```

С. Битые пиксели

Подгруппа 1 ($N = 1, H_1 = 1, Q = 1$)

В этой подгруппе есть только одно изображение, и оно состоит из одной строки (высота 1). Также есть ровно один битый пиксель. Нужно вывести одну строку из W_1 символов. В столбце, который указан для битого пикселя, выводим звёздочку, во всех остальных столбцах — точку.

Пример кода на Python:

```
N = int(input())
H1, W1 = map(int, input().split())
Q = int(input())
k, r, c = map(int, input().split())
X = int(input())

for j in range(1, W1 + 1):
    if j == c:
        print('*', end='')
    else:
        print('.', end='')
```

Подгруппа 2 ($N = 1, H_1 = 1$)

Изображение всё ещё одно и состоит из одной строки, но теперь битых пикселей может быть несколько. Нужно вывести одну строку, в которой в столбцах, перечисленных в битых пикселях, стоят звёздочки, а в остальных — точки. Создадим список длины W_1 , в котором будем отмечать, какие пиксели стали битыми.

Пример кода на Python:

```
N = int(input())
H1, W1 = map(int, input().split())
Q = int(input())

broken_pixels = [False] * (W1 + 1)
for i in range(Q):
    k, r, c = map(int, input().split())
    broken_pixels[c] = True
X = int(input())

for j in range(1, W1 + 1):
    if broken[j] == True:
        print('*', end='')
    else:
        print('.', end='')
```

Подгруппа 3 ($N_i = 1$ для всех изображений)

Здесь каждое изображение состоит из одной строки. Разные изображения могут иметь разную ширину. Сохраним ширину для каждого изображения. Также сохраним список всех битых пикселей. Нам нужно вывести только изображение с номером X . Для этого, как и в предыдущем решении, создадим список длины W_x , в котором будем хранить, какие пиксели в изображении с номером X стали битыми. Пройдёмся по всем битым пикселям: если $k = X$, то пиксель в столбце c стал битым.

Пример кода на Python:

```
N = int(input())
W = [0] * (N + 1)
for i in range(1, N + 1):
    H, W[i] = map(int, input().split())
Q = int(input())

broken_pixels_list = []
for i in range(Q):
    k, r, c = map(int, input().split())
    broken_pixels_list.append((k, r, c))
X = int(input())

broken_pixels_in_X = [False] * (W[X] + 1)
for i in range(Q):
    k, r, c = broken_pixels_list[i]
    if k == X:
        broken_pixels_in_X[c] = True

for i in range(1, W[X] + 1):
    if broken_pixels_in_X[i] == True:
        print('*', end='')
    else:
        print('.', end='')
```

Подгруппа 4 ($Q = 1$)

Всего один битый пиксель. Нам нужно вывести изображение X . Необходимо знать размеры изображения X , поэтому сохраним высоту и ширину для каждого изображения. Если этот единственный битый пиксель принадлежит изображению X , то в соответствующей строке r и столбце c выводим звёздочку, иначе всё поле состоит из точек.

Пример кода на Python:

```

N = int(input())
H = [0] * (N + 1)
W = [0] * (N + 1)
for i in range(1, N + 1):
    H[i], W[i] = map(int, input().split())
Q = int(input())
k, r, c = map(int, input().split())
X = int(input())

for i in range(1, H[X] + 1):
    for j in range(1, W[X] + 1):
        if k == X and i == r and j == c:
            print('*', end='')
        else:
            print('.', end='')
    print()

```

Подгруппа 5 ($Q \leq 5$)

Битых пикселей не больше пяти. Заведём отдельные переменные для каждого из них ($k_1, r_1, c_1, k_2, r_2, c_2, \dots$). Далее при выводе изображения X проверяем каждую клетку на совпадение с любым из этих пяти пикселей. Так как Q может быть меньше пяти, лишние переменные можно заполнить нулями (или заведомо несовпадающими значениями).

Пример кода на Python:

```

N = int(input())
H = [0] * (N + 1)
W = [0] * (N + 1)
for i in range(1, N + 1):
    H[i], W[i] = map(int, input().split())

Q = int(input())
k1 = r1 = c1 = 0
k2 = r2 = c2 = 0
k3 = r3 = c3 = 0
k4 = r4 = c4 = 0
k5 = r5 = c5 = 0
if Q >= 1:
    k1, r1, c1 = map(int, input().split())
if Q >= 2:
    k2, r2, c2 = map(int, input().split())
if Q >= 3:
    k3, r3, c3 = map(int, input().split())
if Q >= 4:
    k4, r4, c4 = map(int, input().split())
if Q >= 5:
    k5, r5, c5 = map(int, input().split())

X = int(input())

for i in range(1, H[X] + 1):
    for j in range(1, W[X] + 1):
        if (k1 == X and i == r1 and j == c1) or \
            (k2 == X and i == r2 and j == c2) or \
            (k3 == X and i == r3 and j == c3) or \
            (k4 == X and i == r4 and j == c4) or \
            (k5 == X and i == r5 and j == c5):
            print('*', end='')
        else:
            print('.', end='')
    print()

```

Подгруппа 6 (полные ограничения)

Храним размеры всех изображений и все битые пиксели. При выводе изображения X для каждой клетки проверяем, есть ли среди битых пикселей такой, у которого $k = X, r = i, c = j$. Так как Q и размеры небольшие (до 100), можно просто перебирать все битые пиксели для каждой клетки, это быстро.

Пример кода на Python:

```

N = int(input())
H = [0] * (N + 1)
W = [0] * (N + 1)
for i in range(1, N + 1):
    H[i], W[i] = map(int, input().split())

Q = int(input())
broken_pixels_list = []
for i in range(Q):
    k, r, c = map(int, input().split())
    broken_pixels_list.append((k, r, c))

X = int(input())

for i in range(1, H[X] + 1):
    for j in range(1, W[X] + 1):
        is_broken = False
        for (k, r, c) in broken_pixels_list:
            if k == X and r == i and c == j:
                is_broken = True
                break
        if is_broken:
            print('*', end='')
        else:
            print('.', end='')
    print()

```

D. Произведение чисел

Подгруппа 1 ($N \leq 5$)

Чисел не больше пяти. Заведём отдельные переменные для каждого числа. Если N равно 1, то произведение равно этому числу. Если N равно 2, перемножаем два числа. И так далее до пяти. После умножения смотрим на знак: если больше нуля, выводим 1; если меньше нуля, выводим -1; если равно нулю, выводим 0.

Пример кода на Python:

```
N = int(input())
result = 0

if N == 1:
    a = int(input())
    result = a
elif N == 2:
    a, b = map(int, input().split())
    result = a * b
elif N == 3:
    a, b, c = map(int, input().split())
    result = a * b * c
elif N == 4:
    a, b, c, d = map(int, input().split())
    result = a * b * c * d
else:
    a, b, c, d, e = map(int, input().split())
    result = a * b * c * d * e

if result > 0:
    print(1)
elif result < 0:
    print(-1)
else:
    print(0)
```

Подгруппа 2 (каждое число от -1 до 1)

Здесь все числа могут быть равны -1, 0 или 1. Можно просто перемножить все числа в цикле. Это быстро, потому что числа маленькие и их произведение никогда не будет меньше -1 или больше 1.

Пример кода на Python:

```
N = int(input())
a = list(map(int, input().split()))

result = 1
for i in range(N):
    result *= a[i]
print(result)
```

Подгруппа 3 (полные ограничения)

Теперь чисел может быть очень много, и каждое число от -1000 до 1000. Если попытаться перемножать их все подряд, то произведение станет невероятно огромным. Компьютеру придётся тратить очень много времени на умножение гигантских чисел, и программа не уложится в одну секунду (ограничение времени работы программы). Но нам ведь нужен только знак произведения. Достаточно узнать, есть ли среди чисел ноль, и сколько отрицательных чисел.

Если хотя бы одно число равно нулю, то всё произведение равно нулю, и ответ 0.

Если нулей нет, то смотрим на количество отрицательных чисел. Если их чётное количество, то произведение положительное (ответ 1). Если нечётное, то произведение отрицательное (ответ -1). Это работает, потому что каждое отрицательное число меняет знак итогового произведения на противоположный.

На положительные числа мы не обращаем внимания, так как они не меняют знак итогового произведения.

Пример кода на Python:

```
N = int(input())
a = list(map(int, input().split()))

zero = False
negative_numbers = 0
for i in range(N):
    if a[i] == 0:
        zero = True
        break
    if a[i] < 0:
        negative_numbers += 1

if zero == True:
    print(0)
elif negative_numbers % 2 == 0:
    print(1)
else:
    print(-1)
```

Е. Невырожденный треугольник

Подгруппа 1 (координаты x_1 , x_2 и x_3 равны только 0 или 1)

В задаче нужно проверить, не лежат ли три точки на одной прямой. В этой подгруппе точки лежат на одной прямой, только если все координаты x одинаковые. Если все координаты x равны, то точки находятся на одной вертикальной прямой и треугольника не получится. Если среди координат x есть различные, то треугольник невырожденный.

Пример кода на Python:

```
x1, y1 = map(int, input().split())
x2, y2 = map(int, input().split())
x3, y3 = map(int, input().split())
if x1 == x2 == x3:
    print("NO")
else:
    print("YES")
```

Подгруппа 2 (все координаты от -1 до 1)

Здесь все координаты x и y могут быть равны только -1, 0 или 1. Точки лежат на одной прямой в следующих случаях: или все три точки имеют одинаковую координату x (вертикаль); или все три имеют одинаковую координату y (горизонталь); или они лежат на одной диагонали между точками (-1, -1) и (1, 1) (то есть разность y равна разности x для любой пары); или на диагонали между точками (-1, 1) и (1, -1) (разность y равна минус разности x). Проверяем эти четыре условия. Если ни одно не выполнено, треугольник невырожденный.

Пример кода на Python:

```
x1, y1 = map(int, input().split())
x2, y2 = map(int, input().split())
x3, y3 = map(int, input().split())

if x1 == x2 == x3:
    print("NO")
elif y1 == y2 == y3:
    print("NO")
elif (y2 - y1 == x2 - x1) and (y3 - y1 == x3 - x1) and (y3 - y2 == x3 - x2):
    print("NO")
elif (y2 - y1 == -(x2 - x1)) and (y3 - y1 == -(x3 - x1)) and (y3 - y2 == -(x3 - x2)):
    print("NO")
else:
    print("YES")
```

Подгруппа 3 (полные ограничения)

Чтобы проверить, лежат ли точки на одной прямой, сравним угловые коэффициенты прямых, проходящих через первую точку и каждую из двух

других. Если прямые имеют одинаковый наклон, то все три точки лежат на одной прямой. Наклон вычисляется как $(y_2 - y_1) / (x_2 - x_1)$ для прямой, образованной первой и второй точкой, и как $(y_3 - y_1) / (x_3 - x_1)$ для прямой, образованной первой и третьей точкой. Теперь необходимо сравнить $(y_2 - y_1) / (x_2 - x_1)$ и $(y_3 - y_1) / (x_3 - x_1)$. Но если $x_1 = x_2$ или $x_1 = x_3$, то мы будем делить на ноль, что делать нельзя. Поэтому умножим обе дроби на значение $(x_2 - x_1) \cdot (x_3 - x_1)$. Получим $(y_2 - y_1) \cdot (x_3 - x_1)$ и $(y_3 - y_1) \cdot (x_2 - x_1)$. Эти значения и нужно сравнить.

Пример кода на Python:

```
x1, y1 = map(int, input().split())
x2, y2 = map(int, input().split())
x3, y3 = map(int, input().split())

if (y2 - y1) * (x3 - x1) == (y3 - y1) * (x2 - x1):
    print("NO")
else:
    print("YES")
```