Задача 1. Поиск подстроки

В первом вопросе строка «БА» входит 2 раза в строку «БАОБАБ», если её повторить 100 раз, то число повторений будет равно 200.

Во втором вопросе строка «АР» входит один раз в строку «РЕМАРКА» (всего 100 раз), и ещё один раз входит на границу между двумя строками «РЕМАРКАРЕМАРКА», что даёт ещё 99 вхождений. Итого 199 вхождений.

В третьем вопросе строка «АУАУ» входит два раза в строку «АУАУОАУАУ»>, и ещё одно вхождение образуется “на границе” между двумя вхождениями. Итого 299 вхождений.

В четвёртом вопросе если строку «ОЙОЙ» повторить 100 раз, то получится 200 повторений слога «ОЙ». Нам нужно посчитать число вхождений «ЙОЙОЙ», для этого нужно три слога идущих подряд. Это можно сделать 198-ю способами.

Наконец, в пятом вопросе нужно посчитать количество появлений строки из 50 букв «А» внутри строки из 100 букв «А». Это можно сделать 51-м способом: до выбранной строки может идти от 0 до 50 букв «А».

Правильный ответ:

```
200
199
299
198
51
```

Задача 2. Скандинавский флаг

Сторона левого квадрата равна $(n-k)/2$. Это же число равно высоте прямоугольника, площадь которого нужно определить. Посчитаем его ширину. Она равна m за вычетом ширины полосы и найденной стороны квадрата, то есть $m-k-(n-k)/2$. Нужно перемножить эти значения и записать ответ в виде

```
(n - k) / 2 * (m - k - (n - k) / 2)
```

или в виде любого эквивалентного выражения.

Задача 3. Социальная дистанция

В задании 3-1 $N = 3$, $M = 5$, $d = 2$. В этом задании нужно рассадить участников в шахматном порядке (заняв все углы).

```
10101
01010
10101
```

В задании 3-2 $N = 6$, $M = 10$, $d = 4$. При таком расстоянии наиболее плотная упаковка мест (в случае бесконечного поля) будет достигаться при размещении зрителей на местах в вершинах квадратов. Используя такой метод можно рассадить 8 человек.

```
1000100010
0000000000
0010001000
0000000000
1000100010
0000000000
```

Но можно немного подвинуть некоторых участников, и тогда получится разместить 9 человек.

```
0001000100
1000000000
0000010001
0010000000
0000000100
1000100000
```

В задании 3-3 $N = 4$, $M = 6$, $d = 3$. При такой расстоянии наиболее плотное размещение на бесконечной плоскости будет, если размещать участников в вершинах повёрнутого квадрата, сторона квадрата выглядит, как “ход коня”:

```
0100
0001
1000
0010
```

При такой рассадке можно в зале размером 4×6 посадить 5 человек.

```
010000
000100
100001
001000
```

Но также если немного подвинуть места, то можно разместить 6 человек.

```
100100
000001
001000
100010
```

В задании 3-4 $N = 7$, $M = 10$, $d = 3$. Занимая места в вершинах квадрата можно рассадить 14 человек.

```
0010000100
0000100001
0100001000
0001000010
1000010000
0010000100
0000100001
```

Но можно посадить и 15 человек, например, следующим образом.

```
0010010001
1000000100
0001000000
0100001001
0000100000
0010000100
1000010001
```

Задача 4. Сортировка

Самое короткое решение содержит 5 шагов. Пример такого решения.

```
HGFEDCBA
EHGFDCBA
EFDCHGBA
EFGBADCH
ADEFGBCH
ABCDEFGH
```

Задача 5. Кинотеатр

На 60 баллов можно было написать решение, суммирующее количество мест в каждом ряду: N, $N + 1$, N, $N + 1$, ... пока сумма станет не меньше K.

Для решения на полный балл заметим, что в двух соседних рядах сумма мест равна $2N + 1$, и если разбить ряды на пары, то можно определить, в какую пару рядов попадает K-е место, поделив и взяв остаток от деления на $2K+1$. Также нужно потом рассмотреть один из двух случаев: в какой ряд (нечётный или чётный) попало наше место, сравнив остаток от деления на $2K+1$ с числом мест в первом ряду N.

Заметим, что в таких задачах удобней нумеровать объекты с 0. В приведённом ниже решении осуществляетя переход в 0-нумерацию, вычитанием числа 1, а при выводе ответа прибавляется 1.

Пример решения.

```
n = int(input())
k = int(input())
k -= 1
rows2 = k // (2 * n +  1)
k %= 2 * n + 1
if k < n:
    print(2 * rows2 + 1, k + 1)
else:
    print(2 * rows2 + 2, k + 1 - n)
```

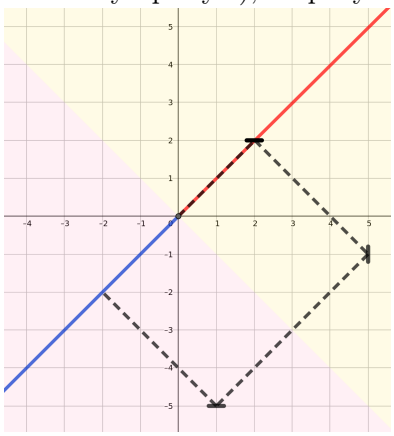
Задача 6. Лазерная пушка

В этой задаче нужно аккуратно разобрать все возможные случаи.

Заметим, что мы можем попасть только в те целочисленные точки, у которых сумма $x + y$ — чётная. Если сумма нечётная, то ответ «-1».

Для поражения точек, лежащих на луче $y = x$, $x > 0$ зеркала не нужны, и ответ в этом случае «0». На рисунке этот луч закрашен красным цветом.

В любом другом случае первое зеркало необходимо поставить на этом луче. Тогда луч можно отразить так, чтобы он прошёл через точки полуплоскости, лежащей выше прямой $y = -x$ (не включая эту прямую), на рисунке ниже эта область закрашена жёлтым цветом.



При этом для того, чтобы поразить в этой области цель ниже прямой $y = x$ (т.е. ниже луча, после его выхода из пушки),нужно поставить горизонтальное зеркало, а для того, чтобы поразить цель выше прямой $y = x$, необходимо вертикальное зеркало. В решении должны быть разобраны эти два случая.

Для поражения оставшейся части плоскости, кроме луча $y = x$, $x < 0$ понадобится два зеркала. Эта область покрашена на рисунке розовым цветом. Второе зеркало должно быть ориентировано не так, как первое. Два зеркала также понадобятся для точек на границе жёлтой и розовой областей (это прямая $y = -x$).

Наконец, для поражения цели на луче $y = x$, $x < 0$, то есть на продолжении того луча, который выходит из пушки, в противоположную сторону (на рисунке он покрашен в синий цвет), понадобится три зеркала.

На рисунке изображён пример расстановки трёх зеркал для поражения цели на этом луче. Луч изображён пунктирной линией, зеркала — короткими отрезками. Рисунок также иллюстрирует примеры расстановки зеркал и для других случаев.

Пример решения.

```
x = int(input())
y = int(input())

if (x + y) % 2 == 1:  # Нет решения
```

```
    print(-1)
elif x == y > 0:  # Нет зеркал
    print(0)
elif -x < y < x:  # Одно зеркало, ниже прямой y = x
    print(1)
    print((x + y) // 2, (x + y) // 2, 'H')
elif y > -x: # Одно зеркало, выше прямой y = x
    print(1)
    print((x + y) // 2, (x + y) // 2, 'V')
elif y < x:  # Два зеркала, ниже прямой y = x
    print(2)
    print(1, 1, 'H')
    print(1 + (x - y) // 2, 1 - (x - y) // 2, 'V')
elif y > x:  # Два зеркала выше прямой y = x
    print(2)
    print(1, 1, 'V')
    print(1 - (y - x) // 2, 1 + (y - x) // 2, 'H')
else:  # Три зеркала, на луче y = x, x < 0
    print(3)
    print(1, 1, 'H')
    print(2, 0, 'V')
    print(x + 1, y - 1, 'H')
```

Задача 7. Найдите отсутствующего

Заведём массив `present` из N элементов (при нумерации элементов массива с нуля удобней завести массив из $N + 1$ элемента), отмечая в нём, присутствовал ли этот человек или нет. Первоначально все элементы этого массива будут иметь значение «Ложь», а после считывания числа k присвоим k-му элементу массива значение «Истина». Потом пройдём по всему массиву и выведем индекс такого элемента, значение которого осталось «Ложь».

Пример такого решения.

```
n = int(input())
present = [False] * (n + 1)
for i in range(n - 1):
    k = int(input())
    present[k] = True
for i in range(1, n + 1):
    if not present[i]:
        print(i)
```

Но у этой задачи есть и красивое решение, не использующее массивов. Для этого заметим, что если рассмотреть сумму чисел $1 + 2 + ... + n$ и вычесть данные числа, то получится как раз недостающее число. Пример такого решения (к переменной `s` добавляются числа от 1 до n, и вычитаются данные числа):

```
n = int(input())
s = 0
for i in range(1, n):
    s += i
    s -= int(input())
s += n
print(s)
```