Задача 1. Открытие школы 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 мегабайт
Входные данные	стандартный ввод
Выходные данные	стандартный вывод

Профессор Икс, он же Чарльз Фрэнсис Ксавье, известный как лидер и основатель Людей-Икс, выбирает дату открытия школы для одарённых мутантов. Во вселенной профессора используется сквозная нумерация дней. По его теории, если открыть школу в день с номером N, то через K дней надо обязательно отпраздновать юбилей открытия школы. При этом K должно быть особенным числом – наименьшим натуральным числом с суммой цифр равной N, которое является также и кратным N.

Требуется для заданного числа N, найти наименьшее натуральное число, кратное N с суммой цифр равной N.

Формат входных данных

Первая строка содержит одно натуральное число N ($N \leq 50$).

Формат выходных данных

Одно число – наименьшее натуральное число с суммой цифр равной N, которое является также и кратным N.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
5	5
10	190

Описание системы оценивания

Баллы начисляются за каждый пройденный тест.

Решение

Для решения задачи воспользуемся функцией для определения суммы цифр числа. Решение задачи проведём путём прямого перебора с шагом n.

Пример программы для решения задачи.

```
# определение суммы цифр числа
def sumd(n):
    sm = 0
    while (n != 0):
        sm = sm + n % 10
        n = n // 10
    return sm

n = int(input())
flag = False
k = n
# перебор чисел по возрастанию
while not flag:
    if n == sumd(k) and k%n == 0:
        flag = True
    else:
   # числа кратные n идут через n
```

```
        k += n
print(k)
```

Задача 2. Простая игра 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 мегабайт
Входные данные	стандартный ввод
Выходные данные	стандартный вывод

Профессор Икс иногда играет в следующую игру с антигероем Магнито. Первый игрок пишет строку, состоящую только из цифр и заглавных букв латиницы. Задача второго игрока быстро найти минимальное основание системы счисления, в которой записано число и записать его в десятичной системе счисления. У Шарля Фрэнсиса Ксавье (он же профессор Икс) это получается очень быстро, так как профессор – научный гений со сверхчеловеческой способностью обрабатывать информацию. А вот антигерою Магнито требуется помощь в виде программы, позволяющей это сделать за него.

Требуется найти по заданной строке, состоящей из цифр и заглавных букв латинского алфавита, являющейся записью числа в некоторой позиционной системе счисления, как выглядит минимально возможное десятичное число.

Алфавит 36-ричной системы счисления:

0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ

Формат входных данных

Первая строка состоит из N символов ($N \leq 20$).

Формат выходных данных

Одна строка – минимальное значение десятичного числа

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
1010	10
12	5
C0	156

Описание системы оценивания

Баллы начисляются за каждый пройденный тест.

Решение

Вариант 1

Для определения минимальной системы счисления достаточно определить максимальную цифру в числе, она и будет ограничивать алфавит системы счисления.

Пример программы для решения задачи.

```
a = input()
# Заполним список с максимальным алфавитом (36-ричной системы)
dig = []
for i in range(10):
    dig.append(str(i))
for i in range(26):
    dig.append(chr(i+ord('A')))
# Найдём минимально возможное основание
p = 0
```

```
for i in a:
    if dig.index(i) > p:
        p = dig.index(i)
p += 1
```
 # Переведём число в 10-чную систему счисления
```
s = 0
st = len(a)-1
for i in a:
    s += dig.index(i) * (p ** st)
    st -= 1
print(s)
```

Вариант 2

Если сосредоточиться на концепции языка программирования Python, то решение может получиться значительно короче.

 # функция для нахождения суммы цифр заданного числа
```
def get_min_digit_system(s):
    abc = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    ss = 0;
```
 # для каждого символа данной строки s определяем его место в алфавите и
 # запоминаем максимальное из таких значений.
```
    for symbol in s:
        ss = max(ss, abc.find(symbol))
    return ss + 1
```
 # непосредственно программа
```
s = input()
digit_system = get_min_digit_system(s)
answer = int(s, digit_system)
print(answer)
```

Задача 3. Дорожное движение — 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 мегабайт
Входные данные	стандартный ввод
Выходные данные	стандартный вывод

Магнето, заклятый враг команды Людей Икс, планирует захват сети населённых пунктов. Не смотря на то, что некоторые антигерои умеют летать, некоторым из них приходится передвигаться по дорогам.

Некоторые населённые пункты, соединены дорогами с односторонним, а иногда и двусторонним движением. Антигерою для планирования захвата необходимо знать, сколько дорог и каких видов соединяют населённые пункты.

Для отображения схемы соединения дорог Магнето использует матрицу, в которой если стоит цифра 0, то из пункта i в пункт j дороги нет, а если цифра 1, то есть.

Например. Пусть у нас есть 4 населённых пункта и схема дорог представлена следующей таблицей:

Населённый пункт		i			
		1	**2**	**3**	**4**
j	**1**	1	1	0	0
	2	1	1	0	1

	3	0	1	1	0
	4	0	0	0	1

Будем рассматривать последовательности $i \to j$, $j \to i$ ($i \neq j$). Из этой схемы видно, что, например, 1 и 2 населённые пункты имеют двустороннюю дорогу, так как и из 1 в 2 есть дорога, и из 2 в 1. Из пункта 2 в пункт 3 ведёт односторонняя дорога, так как из пункта 3 в пункт 2 дороги нет.

Требуется найти количество дорог с двусторонним и с односторонним движением.

Формат входных данных

В первой строке содержится одно число N ($2 \le N \le 10^3$) – количество населённых пунктов.

Далее N строк, состоящих из N символов 0 или 1, разделённых пробелом – схема соединения дорог

Формат выходных данных

В первой строке – количество дорог с односторонним движением. ВО второй строке – количество дорог с двусторонним движением.

Пример входных и выходных данных

Стандартный ввод	Стандартный вывод
4 1 1 0 0 1 1 0 1 0 1 1 0 0 0 0 1	2 1

Описание системы оценивания

Баллы начисляются за каждый пройденный тест.

Решение

Для решения задачи можно просматривать элементы над главной диагональю и зеркальные к ним элементы под главной диагональю матрица. Если зеркальные элементы равны 1, то это дорога с двусторонним движением. Если только один из зеркальных элементов равен единице, то это дорога с односторонним движением.

Пример программы для решения задачи.

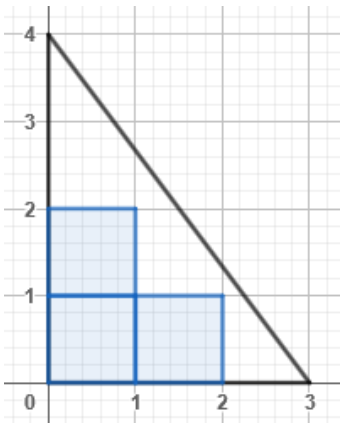
```
# для удобства будем читать данные из файла
f = open('input.txt')
n = int(f.readline())
a = []
# Читаем данные в массив
for i in range(n):
    a.append([int(t) for t in f.readline().split()])
f.close()
k2, k1 = 0, 0
# Рассматриваем половину матрицы. Если сумма зеркальных элементов
# равна двум, то дорога двустороння, если 1, то односторонняя
for row in range(n):
    for col in range(row+1,n):
        if a[row][col] + a[col][row] == 2:
            k2 += 1
        if a[row][col] + a[col][row] == 1:
            k1 += 1
```

```
print(k1)
print(k2)
```

Задача 4. Треугольная операция 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 мегабайт
Входные данные	стандартный ввод
Выходные данные	стандартный вывод

Джеймс Хоулетт (он же Росомаха) не только эксперт многих видов рукопашного боя, но и великолепный тактик и стратег. При планировании стратегической операции на территории, в форме прямоугольного треугольника, было принято решение разбить его на квадраты с единичной стороной, для того, чтобы можно было обеспечить патрулирование каждого квадрата одним из членов Команды Икс.



Угол первого квадрата совпадает с прямым углом треугольника. Остальные квадраты по горизонтали и вертикали прилегают к нему стороной, а по диагонали – вершиной (см. рисунок).

Стратегу Росомахе очень важно знать, сколько полных квадратов можно поместить таким образом в прямоугольный треугольник.

Требуется по известным катетам A и B прямоугольного треугольника определить, сколько полных квадратов с единичной стороной в него поместиться.

Формат входных данных

В первой строке записаны два числа A и B, разделённые пробелом N ($1 \le A, B \le 10^6$).

Формат выходных данных

Одно число – количество поместившихся квадратов.

Пример входных и выходных данных

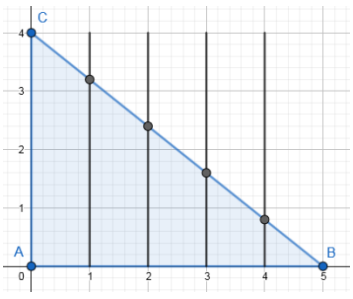
Стандартный ввод	Стандартный вывод
4 3	3
4 4	6

Описание системы оценивания

Баллы начисляются за каждый пройденный тест.

Решение

Вариант 1



Рассмотрим прямоугольный треугольник ABC с катетами AB (далее обозначим в программе dx) и AC (далее обозначим в программе dy). Проведём, например, вертикальные линии с шагом 1 и найдём точки пересечения их с гипотенузой. Целая часть ординаты точки пересечения и есть количество квадратов, которые попадают в прямоугольный треугольник слева от этой прямой. Остаётся только найти сумму целых частей ординат всех точек пересечения, полученных вертикальных линий с гипотенузой.

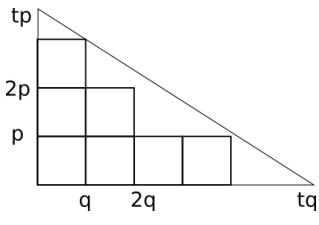
Пример программы для решения задачи.

```
# Нахождение точки пересечения двух прямых
def cross(x1,y1,x2,y2,x3,y3,x4,y4):
    if y2 - y1 != 0:
        q = (x2 - x1) / (y1 - y2)
        sn = (x3 - x4) + (y3 - y4) * q
        fn = (x3 - x1) + (y3 - y1) * q
        n = fn / sn
    else:
        n = (y3 - y1) / (y3 - y4)
    x = x3 + (x4 - x3) * n
    y = y3 + (y4 - y3) * n
    return [x,y]

dx, dy = map(int, input().split())
n = 0
for x in range(1,dx):
    cr = cross(dx,0,0,dy,x,0,x,dx)
    n += int(cr[1])
print(n)
```

Вариант 2 (Порублев И.Н., Ставровский А.Б. Алгоритмы и программы. Решение олимпиадных задач – М. : ООО «И.Д. Вильямс», 2007.)

Рассмотрим длины катетов, имеющие общий делитель t, т.е. числа вида m = p·t, n = q·t. Тогда количество квадратов единичной длины описывает формула



$$K(pt, qt) = \frac{t(t-1)}{2} \cdot p \cdot q + t \cdot K(p, q)$$

В границы треугольника попадают (t-1)+(t-2)+…+1 = $\frac{t(t-1)}{2}$ прямоугольников размерами p×q и t одинаковых прямоугольных треугольников p и q.

Если длины катетов взаимно просты, то имеет место формула

$$K(m, n) = \big(m \cdot n - (m + n - 1)\big)\ div\ 2$$

Пример программы для решения задачи.

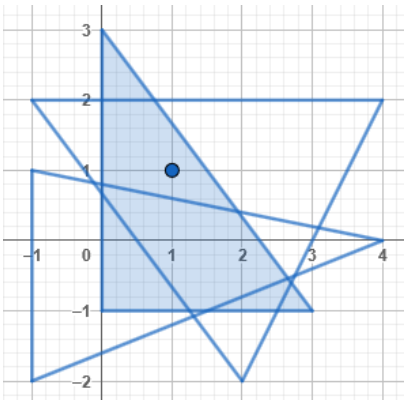
```
# функция для нахождения наибольшего общего делителя
def nod(a,b):
    while a!=0 and b!=0:
        if a > b:
            a = a % b
        else:
            b = b % a
    return a+b
# функция для числа единичных квадратов
def k(x,y):
    if nod(x,y) == 1:
        return (x*y-(x+y-1))//2
    else:
        t = nod(x,y)
        a = x // t
        b = y // t
        return ((t*(t-1))//2)*a*b+t*k(a,b)
n, m = map(int, input().split())
```

```
print(k(n,m))
```

Задача 5. Минимальный захват — 100 баллов

Ограничение по времени	1 секунда
Ограничение по памяти	1 мегабайт
Входные данные	стандартный ввод
Выходные данные	стандартный вывод

Скотт «Слим» Саммерс (тот, что Циклоп) обладает способностью выпускать из глаз лазерные лучи. Во время очередной битвы с врагом человечества он поставил перед собой цель захватить врага живьём. Для этого он стремительно перемещался по полю боя и, останавливаясь на мгновение, пускал лазерные лучи. Делал он это таким образом, чтобы лучи, выпущенные из трёх точек, образовывали треугольник, в который, по его мнению, должен попасться враг. К сожалению, так получалось не всегда. Циклоп проделал описанную операцию множество раз, таким образом, что образовалось множество треугольников.



Джеймс Хоулетт, анализируя в дальнейшем тактику боя, решил оптимизировать энергетические затраты и найти тот треугольник, который имел бы минимальную площадь и содержал внутри себя (или на линии огня) врага.

Требуется найти площадь самого маленького треугольника, которому принадлежит точка. Точку, попадающую на сторону треугольника, будем считать принадлежащей ему. Гарантируется, что хотя бы один такой треугольник существует.

Формат входных данных

В первой строке заданы три числа, разделённые пробелом: натуральное число N ($1 \le N \le 10^4$) – количество треугольников, вещественные числа ($-10^3 \le X, Y \le 10^3$) – координаты врага.

Далее N строк, состоящих из шести вещественных чисел (X_1; Y_1), (X_2; Y_2), (X_3; Y_3), разделённых пробелом ($-10^3 \le X_1, Y_1, X_2, Y_2, X_3, Y_3 \le 10^3$) – координаты вершин треугольников. Каждая из N строк описывает один треугольник.

Формат выходных данных

Одно число – наименьшая площадь треугольника, в который попадает враг, с точностью до двух знаков в десятичной части.

Пример входных и выходных данных

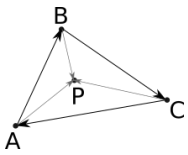
Стандартный ввод	Стандартный вывод
3 1 1 -1 2 4 2 2 -2 0 -1 0 3 3 -1 -1 1 4 0 -1 -2	6.00
2 1.5 1.5 0 0 0 5.5 6.5 0 0 5.5 6.5 0 -3 -3	17.87

Описание системы оценивания

Баллы начисляются за каждый пройденный тест.

Решение

Решение задачи проведём, воспользовавшись понятием косого произведения векторов. Знак этого произведения определяет принадлежность точки одной из полуплоскостей, равенство нулю говорит о принадлежности точки прямой.



Теперь, если обойти треугольник по или против часовой стрелки и найти все косые произведения $\overrightarrow{AB}\,\overrightarrow{AP}$, $\overrightarrow{BC}\,\overrightarrow{BP}$, $\overrightarrow{CA}\,\overrightarrow{CP}$, то одинаковость их знаков говорит о том, что точка находится внутри треугольника. Равенство значения нулю соответствует попаданию точки на сторону.

Пример программы для решения задачи.

```
from math import sqrt
# нахождение косого произведения векторов
def skmult(a1x,a1y,a2x,a2y,b1x,b1y,b2x,b2y):
    ax = a2x - a1x
    ay = a2y - a1y
    bx = b2x - b1x
    by = b2y - b1y
    return ax * by - bx * ay
# площадь треугольника (формула Герона)
def stri(x1,y1,x2,y2,x3,y3):
    a = sqrt((x2-x1)**2+(y2-y1)**2)
    b = sqrt((x3-x2)**2+(y3-y2)**2)
    c = sqrt((x1-x3)**2+(y1-y3)**2)
    p = (a+b+c)/2
    return round(sqrt(p*(p-a)*(p-b)*(p-c)),2)
# определение принадлежности точки треугольнику
def intri(x, y, x1, y1, x2, y2, x3, y3):
    res1 = skmult(x1,y1,x2,y2,x1,y1,x,y)
    res2 = skmult(x2,y2,x3,y3,x2,y2,x,y)
    res3 = skmult(x3,y3,x1,y1,x3,y3,x,y)
    if res1<=0 and res2<=0 and res3<=0 or res1>=0 and res2>=0
and res3>=0:
        return True
    else:
        return False
# для удобства будем читать данные из файла
f = open('input.txt')
n, x, y = map(float, f.readline().split())
n = int(n)
smin = -1
for i in range(n):
    x1, y1, x2, y2, x3, y3 = map(float, f.readline().split())
    r = intri(x, y, x1, y1, x2, y2, x3, y3)
    if r:
        s = stri(x1,y1,x2,y2,x3,y3)
        if smin == -1:
            smin = s
        if s < smin:
            smin = s
f.close()
print("{0:1.2f}".format(smin))
```