Задача 1. Том Сойер

Ограничение по времени: 1 секунда

> Одного только не хватало мистеру Уолтерсу для полного счастья: возможности вручить наградную Библию и похвастать чудом учёности. У некоторых школьников имелись жёлтые билетики, но ни у кого не было столько, сколько надо, — он уже опросил всех первых учеников. И в ту самую минуту, когда всякая надежда покинула его, вперёд выступил Том Сойер с девятью жёлтыми билетиками, девятью красными и десятью синими и потребовал себе Библию.
>
> Марк Твен, «Приключения Тома Сойера».

Для получения одной награды нужно предъявить 10 жёлтых билетиков. 10 красных билетиков можно заменить на один жёлтый. 10 синих билетиков можно заменить на один красный. У Тома сейчас y жёлтых билетиков, r красных и b синих. Сколько наград Том может получить?

Формат входных данных

Три строки входного файла содержат три натуральных числа: y, r и b. Все числа не превосходят 2×10^9.

Формат выходных данных

Выведите одно неотрицательное целое число — количество наград, которые может получить Том. В записи этого числа не должно быть десятичной точки, то есть вывод «1.0» вместо «1» является неправильным.

Система оценки

Решения, верно работающие при $b = r = 0$, будут оцениваться в 10 баллов.

Решения, верно работающие при $b = 0$, будут оцениваться в 30 баллов.

Решения, верно работающие при $y, r, b \leqslant 10^5$, будут оцениваться в 55 баллов.

Пример

стандартный ввод	стандартный вывод
9 9 10	1

Замечание

Пример из условия соответствует эпиграфу. Том обменяет 10 синих билетиков на 1 красный, после чего у него станет $9+1 = 10$ красных билетиков. Далее он обменяет эти 10 красных билетиков на 1 жёлтый, и у него станет $9 + 1 = 10$ жёлтых билетиков. В конце он обменяет эти 10 жёлтых билетиков на одну награду.

Задача 2. Прогрессия

Ограничение по времени: 1 секунда

> Если цифры составляют часть системы символов, в них обычно заметен некий смысл, например, они являются математической прогрессией или определённой комбинацией, то есть каким-то образом связаны друг с другом.
>
> Дэн Браун, «Код да Винчи»

Зайдя в класс, Вова увидел на доске три числа, записанные в ряд. Он не заметил никакой взаимосвязи между ними, и ему сказали, что изначально чисел было четыре и разности между четвертым и третьим, третьим и вторым, вторым и первым равнялись друг другу. Иными словами, на доске была записана *арифметическая прогрессия* из четырех чисел. Однако затем одно из чисел с доски стерли.

Помогите Вове придумать и дописать на доску какое-нибудь число так, чтобы описанное условие снова начало выполняться.

Формат входных данных

Программа получает на вход три целых положительных числа, не превосходящих 10^5 каждое, по одному в строке, в том порядке, в котором они шли на доске.

Формат выходных данных

В первой строке выведите число, которое Вове необходимо написать. Можно доказать, что это число обязательно должно быть целым. В записи этого числа не должно быть десятичной точки, то есть вывод «13.0» вместо «13» является неправильным.

Во второй строке выведите целое число от 1 до 4 — место, на которое его необходимо написать. 1 означает, что указанное число необходимо выписать перед первым из трех приведенных во входных данных чисел, 2 — между первым и вторым, 3 — между вторым и третьим и 4 — после третьего числа.

Гарантируется, что входные данные таковы, что существует хотя бы один способ дополнить их до арифметической прогрессии. Если подходящих способов несколько, выведите любой из них.

Пример

стандартный ввод	стандартный вывод
10 16 19	13 2

Замечание

В примере из условия Вова увидел на доске числа 10, 16 и 19. Если он напишет на доску между первым и вторым из них число 13, то в получившейся четверке чисел 10 13 16 19 разность между четвертым и третьим $(19 - 16)$, третьим и вторым $(16 - 13)$ и вторым и первым $(13 - 10)$ окажется одна и та же, поэтому эта четверка будет арифметической прогрессией.

Задача 3. Расклейка афиш

Ограничение по времени: 1 секунда

> С утра по Васюкам ходил высокий худой старик в золотом пенсне и в коротких, очень грязных, испачканных клеевыми красками сапогах. Он наклеивал на стены рукописные афиши.
>
> И.Ильф, Е.Петров. «Двенадцать стульев».

Ипполит Матвеевич Воробьянинов ходит вдоль улицы из n домов, пронумерованных числами от 1 до n, и расклеивает афиши. Сначала он наклеил афиши на каждый дом, номер которого делился без остатка на a. Поскольку афиш осталось еще много, вторым проходом он наклеил афиши на каждый дом, номер которого делился без остатка на b. При этом, если на доме уже была наклеена афиша, новую Воробьянинов не клеил. Сколько всего афиш расклеил бывший предводитель дворянства?

Формат входных данных

Три строки содержат три натуральных числа: n — количество домов на улице, a и b — выбранные Воробьяниновым числа. Все числа не превосходят 10^9.

Формат выходных данных

Выведите одно неотрицательное целое число — количество расклеенных афиш.

Система оценки

Решения, верно работающие при $n \leqslant 10^5$, будут оцениваться в 60 баллов.

Решения, верно работающие при $a = 2$, будут оцениваться в 20 баллов.

Примеры

стандартный ввод	стандартный вывод
10 2 3	7
5 10 20	0

Замечание

В первом примере на улице 10 домов. Ипполит Матвеевич первым проходом расклеил пять афиш на дома, номера которых делятся на 2, то есть на дома с номерами 2, 4, 6, 8, 10. Вторым проходом он расклеил две афиши на дома, номера которых делятся на 3, то есть на дома с номерами 3 и 9. Дом номер 6 он пропустил — на нем афиша уже висит. Всего наклеено 7 афиш.

Во втором примере Воробьянинов не наклеит ни одной афиши.

Задача 4. Жребий Крижановского

Ограничение по времени: 1 секунда

> Одиночество есть жребий всех выдающихся умов.
>
> Артур Шопенгауэр

Однажды в летнем лагере после ужина осталась лишняя булочка. Выяснить, кому она достанется, дети решили с помощью жребия Крижановского. Правила этой игры такие: каждый участник называет ведущему натуральное число. Среди этих чисел выбираются те, которые были названы ровно один раз, а назвавший минимальное из этих чисел объявляется победителем. Обратите внимание, что победителя может не быть, если среди названных чисел каждое встречается несколько раз.

Вас назначили ведущим. Помогите установить победителя или определить, что такого нет.

Формат входных данных

В первой строке дано одно число n ($1 \leqslant n \leqslant 10^5$) — количество участников игры. Далее в n строках вводятся названные участниками натуральные числа, не превосходящие 10^9.

Формат выходных данных

Программа должна вывести число, написанное победителем. Если победителя нет, то нужно вывести число -1.

Система оценки

Решения, правильно работающие при $n \leqslant 10^3$, будут оцениваться в 40 баллов.

Решения, правильно работающие, когда все числа не превосходят 10^5, будут оцениваться в 30 баллов.

Примеры

стандартный ввод	стандартный вывод
7 5 1 1 3 4 3 1	4
7 2 3 9 3 9 9 2	-1

Замечание

В первом примере из условия участвовали 7 игроков и они назвали числа 5, 1, 1, 3, 4, 3, 1. Сначала оставим только те числа, которые встречаются ровно один раз: 5 и 4. Минимальное из этих чисел равно 4.

Во втором примере победителя нет, т.к. каждое из названных чисел встречается несколько раз.

Задача 5. Долгое вычитание

Ограничение по времени: 1 секунда

> Вызываемые мальчики подходили к доске и должны были писать мелом требуемые цифры и считать их как-то от правой руки к левой...
>
> Сергей Аксаков, «Детские годы Багрова-внука».

На доске написано число n, с которым несколько раз производят следующую операцию: если в записи числа на доске есть хотя бы одна нечётная цифра, то очередной мальчик вычитает из него 1, в противном случае — делит на 2. Сколько мальчиков нужно вызвать, чтобы на доске получился ноль?

Формат входных данных

Единственная строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 10^{18}$).

Обратите внимание, что входные данные в этой задаче могут превышать возможное значение 32-битной целочисленной переменной, поэтому необходимо использовать 64-битные целочисленные типы данных (тип int64 в языке Pascal, тип long long в C и C++, тип long в Java и C#).

Формат выходных данных

Выведите одно натуральное число — ответ на вопрос задачи.

Система оценки

Решения, верно работающие при $n \leqslant 10^5$, будут оцениваться в 40 баллов.

Пример

стандартный ввод	стандартный вывод
25	10

Замечание

В примере дано $n = 25$. Число имеет в своей записи нечётную цифру 5, поэтому после первой операции n уменьшится на 1 и станет равно 24.

Число 24 не имеет в своей записи нечётных цифр, поэтому после второй операции n уменьшится в 2 раза и станет равно 12.

Далее n будет принимать значения: 11, 10, 9, 8, 4, 2, 1 и 0. Всего потребуется 10 операций.

Разбор задач

Задача 1. Том Сойер

В первой подзадаче ($b = r = 0$) у Тома только жёлтые билетики, за каждый десяток которых он получает одну награду. Найдём частное от деления y на 10, это число и является ответом.

```
y = int(input())
r = int(input())
b = int(input())
ans = y // 10
print(ans)
```

Во второй подзадаче ($b = 0$) у Тома только жёлтые и красные билетики. Сведём эту задачу к предыдущей: обменяем все красные билетики на жёлтые, добавим их к тем, которые уже есть у Тома и потом обменяем их все на награды.

```
ans = (r // 10 + y) // 10
```

Третья подзадача ($y, r, b \leqslant 10^5$) позволяет получить баллы за моделирование обменов билетиков с помощью циклов, например, так:

```
while b >= 10:
    b -= 10
    r += 1
while r >= 10:
    r -= 10
    y += 1
ans = 0
while y >= 10:
    y -= 10
    ans += 1
```

Полное решение: узнаем, сколько дополнительных красных билетиков можно получить из синих, и добавим их к уже имеющимся. Аналогично узнаем, сколько дополнительных жёлтых билетиков можно получить из красных, и добавим их к уже имеющимся. Определим количество наград за жёлтые билетики.

```
y = int(input())
r = int(input())
b = int(input())
ans = ((b // 10 + r) // 10 + y) // 10
print(ans)
```

Задача 2. Прогрессия

Решение заключается в разборе различных случаев. Посчитаем разности между выписанными числами: $d_1 = b - a$, $d_2 = c - b$.

Если $d_1 = d_2$, то записанные числа уже образуют арифметическую прогрессию. Значит, стёртое число было первым или четвёртым. Если оно стояло четвёртым, то принимало значение $c + d_1$. Если оно стояло первым, то равнялось $a - d_1$. Можно вывести любой из этих вариантов, они оба правильные.

В остальных случаях стёртое число находилось либо между a и b, либо между b и c. Если $d_1 = 2d_2$, то было стёрто второе число из четырёх, это число равно среднему арифметическому a и b. В противном случае стёрли третье число из четырёх, оно равно среднему арифметическому b и c.

```
a = int(input())
b = int(input())
```

```
c = int(input())

d1 = b - a
d2 = c - b

if d1 == d2:
    print(c + d1, 4)
elif d1 == 2 * d2:
    print((a + b) // 2, 2)
else:
    print((b + c) // 2, 3)
```

При этом нельзя использовать для анализа случаев условие $d_1 > d_2$ вместо $d_1 = 2d_2$, так как арифметическая прогрессия может быть убывающей, тогда значения d_1 и d_2 будут отрицательными. Но возможно использовать абсолютные значения: $|d_1| > |d_2|$.

Задача 3. Расклейка афиш

Первую подзадачу ($n \leqslant 10^5$) можно решить с использованием перебора, например, так:

```
n = int(input())
a = int(input())
b = int(input())
ans = 0
for i in range(1, n + 1):
    if i % a == 0 or i % b == 0:
        ans += 1
print(ans)
```

Вторую подзадачу ($a = 2$) можно решить с использованием разбора двух случаев.

Если b тоже чётное, то при втором проходе Воробьянинов не расклеит новых афиш.

Если b — нечётное, то каждый второй дом, на который нужно наклеить афишу на втором проходе, уже будет иметь афишу (так как сумма двух нечётных чисел всегда чётна). Чтобы найти число подходящих номеров поделим n на b, а потом полученное число поделим на 2 с округлением вверх.

```
ans = n // 2
if b % 2:
    ans += (n // b + 1) // 2
print(ans)
```

Для решения задачи на полный балл нужно сложить количество афиш, наклеенных Воробьяниновым при первом проходе, и количество афиш, которые он наклеит при втором проходе так, если бы первого прохода не было. Мы дважды посчитали дома, номера которых делятся нацело и на a, и на b.

Посчитаем количество номеров домов, которые делятся и на a, и на b. Первое такое числоа должно делиться нацело и на a и на b и быть наименьшим возможным. Согласно определению, это наименьшее общее кратное a и b. Его можно найти через каноническое разложение обоих чисел на простые множители или через наибольший общий делитель (который в свою очередь находится с помощью алгоритма Евклида):

$$\text{НОК}(a, b) = \frac{a \times b}{\text{НОД}(a, b)}$$

Итоговое количество вычитаемых чисел составит n // (a * b // НОД(a, b))

```
def gcd(n, m):
    if m == 0:
        return n
```

```
    return gcd(m, n % m)

n = int(input())
a = int(input())
b = int(input())
ans = n // a + n // b - n // (a * b // gcd(a, b))
print(ans)
```

Задача 4. Жребий Крижановского

В этой задаче нужно найти наименьшее число, которое встречается в данном массиве ровно один раз.

Первую группу ($n \leqslant 10^3$) можно решить перебором всех элементов массива, посчитав количество появлений каждого элемента в массиве. Среди тех элементов, которые встречаются ровно один раз, требуется выбрать наименьший. Сложность такого решения $O(n^2)$.

```
n = int(input())
a = [int(input()) for i in range(n)]

ans = -1
for elem in a:
    if a.count(elem) == 1:
        if ans == -1 or elem < ans:
            ans = elem
print(ans)
```

Во второй группе чисел может быть много, но все они не превосходят 10^5. Для решения этой задачи воспользуемся идеей «подсчёта»: для каждого допустимого значения элемента массива подсчитаем, сколько раз оно встречается в исходном массиве. Для этого заведём массив `cnt`, в котором индекс i принимает значение не превышающее 10^5, и если мы считали число num, то увеличим на 1 значение `cnt[`num`]`.

После этого в массиве `cnt` найдём такое минимальное значение индекса i, что `cnt[`i`]`$= 1$. Если же подходящее значение не найдётся, то выведем -1.

```
n = int(input())
cnt = [0] * (10 ** 5 + 1)
for i in range(n):
    num = int(input())
    cnt[num] += 1

i = 0
while i < len(cnt) and cnt[i] != 1:
    i += 1

if i < len(cnt):
    print(i)
else:
    print(-1)
```

Однако, если числа большие, то массив `cnt` не поместится в памяти. Чтобы такое решение проходило все тесты, нужно заменить массив `cnt` на структуру данных типа «словарь» (например, `dict` в Python или `map` в C++). Затем переберём ключи словаря в порядке возрастания и найти минимальный ключ, для которого значение в словаре будет равно 1.

```
n = int(input())
cnt = dict()
```

```
for i in range(n):
    num = int(input())
    if num not in cnt:
        cnt[num] = 0
    cnt[num] += 1
for num in sorted(cnt.keys()):
    if cnt[num] == 1:
        print(num)
        break
else:
    print(-1)
```

Есть и другое полное решение. Заметим, что если в массиве есть другой элемент, равный рассматриваемому, то после сортировки массива эти числа будут стоять рядом. А если число встречалось в массиве только один раз, то после сортировки соседями этого числа окажутся неравные ему числа. Отсортируем массив, пройдём по нему в порядке возрастания. Если текущий элемент не имеет равного соседа, то он и является ответом, так как если бы было меньшее число, удовлетворяющее условию, то мы бы уже нашли его. Если же мы прошли по всему массиву и не нашли ответ, то нужно вывести «-1»

```
n = int(input())
a = [int(input()) for i in range(n)]
a.sort()
a = [0] + a + [a[-1] + 1]
for i in range(1, n + 1):
    if a[i - 1] < a[i] < a[i + 1]:
        print(a[i])
        break
else:
    print(-1)
```

В этом варианте решения для удобства обработки мы добавили в массив два фиктивных элемента: 0 в начало и число, большее максимального элемента в массиве на 1 в конец для того, чтобы у первого и последнего элемента в массиве также были два соседа.

Задача 5. Долгое вычитание

В первой подзадаче достаточно просто реализовать предложенную последовательность действий.

```
n = int(input())
ans = 0
while n:
    s = str(n)
    if '1' in s or '3' in s or '5' in s or '7' in s or '9' in s:
        n -= 1
    else:
        n //= 2
    ans += 1
print(ans)
```

Для решения задачи на полный балл заметим, что количество операций деления не превосходит $\log_2 n \leqslant 60$, а вот количество выполненных подряд вычитаний единицы может быть очень большим. Например, при $n = 777$ вычитать придётся до числа 688 (почти сто раз).

Попробуем определить, до какого числа нам придётся вычитать единицу, пока не произойдёт следующая операция деления. Очевидно, что первая встретившаяся в числе нечётная цифра d должна стать чётной: ближайшей её четной цифрой, меньшей d, является $d - 1$. На последующих позициях должны стоять наибольшие чётные цифры, то есть восмёрки.

Было: <любые чётные цифры> <нечётная цифра d> <любые цифры>

Стало: <те же чётные цифры> <чётная цифра $d-1$> <8...8>

Количество пропущенных операций вычитания единицы равно разности между старым и новым числом. В предложенном ниже решении функция `first_odd` возвращает индекс первой нечётной цифры, и тогда следующее число строится по описанному алгоритму. Если же нечётных цифр нет, то функция возвращает -1 и тогда выполняется операция деления на 2.

```
n = input()

def first_odd(n):
    for i in range(len(n)):
        if int(n[i]) % 2 == 1:
            return i
    return -1

ans = 0
while int(n) > 0:
    i = first_odd(n)
    if i == -1:
        n = str(int(n) // 2)
        ans += 1
    else:
        new_n = n[:i] + str(int(n[i]) - 1) + "8" * (len(n) - i - 1)
        ans += int(n) - int(new_n)
        n = new_n
print(ans)
```