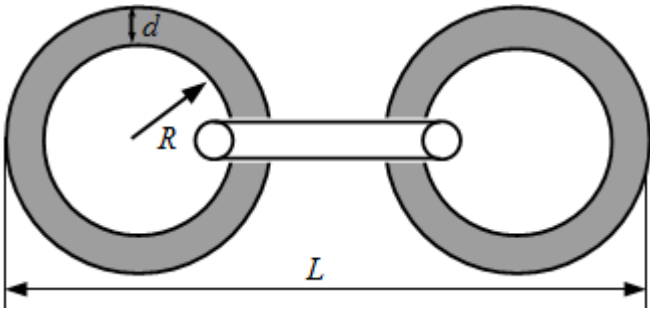
1. # Задача 1. Цепь

Условие

Из проволоки толщиной *d* миллиметров сделали кольца. Внутренний радиус каждого кольца составляет *R* миллиметров. Всего сделали *n* колец и их соединили в цепь. Определите длину получившейся цепи. На рисунке изображен пример для *n* = 3.



Программа получает на вход три числа, записанных в отдельных строчках. В первой строке задана толщина проволоки *d* (в миллиметрах). Во второй строке задан внутренний радиус кольца *R* (в миллиметрах). В третьей строке задано число звеньев *n*. Все числа — натуральные, не превосходящие 100, при этом *d* < *R*.

Программа должна вывести одно целое число *L* — суммарную длину получившейся цепи.

Примеры входных и выходных данных

Ввод	Вывод
2 10 3	64

Решение

Длина цепей складывается из суммарной длины *n* внутренних частей колец радиусом *R* каждая и двух кусков проволоки по краям толщины *d*, поэтому программа должна вывести значение 2*nR* + 2*d*.

Пример правильного решения на языке Python версии 3:

```
d = int(input())
R = int(input())
n = int(input())
L = 2 * R * n + 2 * d
print(L)
```

Задача 2. Лифт

Условие

В торговом центре этажи нумеруются так: …, –3, –2, –1, 1, 2, 3, … (то есть нет нулевого этажа). Вася спустился на лифте с этажа с номером *A* на *B* этажей, а затем поднялся на лифте на *C* этажей. Определите, на каком этаже он оказался.

Программа получает на вход три целых числа: в первой строке записано число *A*, во второй — *B*, в третьей — *C*. Число *A* не равно нулю и не превосходит по модулю 100, числа *B* и *C* — положительные и не превосходят 100.

Программа должна вывести одно целое число — номер этажа, на котором оказался Вася.

Примеры входных и выходных данных

Ввод	Вывод
5 2 10	13
3 10 1	-7

Система оценивания

Решение, правильно работающее только для случая, когда лифт не опускается ниже этажа номер 1 или не поднимается выше этажа номер -1 будет оцениваться в 2 балла.

Решение

Считав входные данные, перенумеруем этажи так, чтобы нумерация этажей стала соответствовать целым числам, то есть если номер этажа *A* был отрицательным, прибавим к числу *A* значение 1. Теперь у этажей нормальная нумерация: …, –2, –1, 0, 1, 2, 3, …

Поэтому если лифт спустится на *B* этажей вниз, а затем поднимется на *C* этажей вверх, то Вася окажется на этаже с номером *A* – *B* + *C*. Теперь вернем нумерацию этажей к исходной: если номер этажа меньше или равен 0, то вычтем из него число 1.

Пример правильного решения на языке Python версии 3:

```
A = int(input())
B = int(input())
C = int(input())
if A < 0:
   A += 1
Ans = A - B + C
if Ans <= 0:
   Ans -= 1
print(Ans)
```

1. # Задача 3. Длинное число

   Дано натуральное число. Разделите точками цифры этого числа группами по три, начиная справа.

Программа получает на вход натуральное число, содержащее не более 100 цифр.

Программа должна вывести то же число, с точками между некоторыми цифрами этого числа.

Примеры входных и выходных данных

Ввод	Вывод
1000	1.000
12345678	12.345.678

Система оценивания

Решение, правильно работающее для случая, когда число содержит не более четырех цифр, будет оцениваться в 30 баллов.

Решение, правильно работающее для случая, когда число содержит не более девяти цифр, будет оцениваться в 60 баллов.

Решение

Поскольку число может иметь длину до 100 цифр, то для работы с таким числом нужно использовать строковый тип данных. Дальнейшее решение задачи представляет только техническую трудность в реализации – необходимо брать “с конца” строки по 3 символа и добавлять перед ними точку.

Пример правильного решения на языке Python версии 3:

```
S = input()[::-1]
ans = ""
for i in range(0, len(S), 3):
    ans += S[i:i + 3] + "."
ans = ans[-2::-1]
print(ans)
```

Другой вариант правильного решения. Будем выводить по одному цифры строки в цикле, а после вывода каждого символа делаем проверку: если текущий символ не последний и количество символов в строке после текущего делится на 3, то необходимо вывести точку после текущего символа.

```
S = input()
for i in range(0, len(S)):
    print(S[i], end='')
    if i < len(S) - 1 and (len(S) – 1 - i) % 3 == 0:
        print('.', end='')
```

Частичные решения

Решение, использующее строки, но выводящее лишнюю точку в начале числа, если количество цифр в числе делится на 3, набирает 70 баллов.

Решение, использующее 16-битные целые числа вместо строк, набирает 30 баллов.

Решение, использующее 32-битные целые числа вместо строк, набирает 60 баллов.

Задача 4. Сумма цифр

Даны два числа *A* и *B*. Подсчитайте количество натуральных чисел на отрезке от *A* до *B*, сумма цифр которых четна.

Программа получает на вход два натуральных числа *A* и *B*, не превосходящих 10^9, $A \le B$.

Программа должна вывести одно число — количество натуральных чисел, больше или равных *A* и меньших или равных *B*, сумма цифр которых четна.

Примеры входных и выходных данных

Ввод	Вывод
10 20	6
10 10	0

Система оценивания

Решение, правильно работающее для случая, когда числа *A* и *B* — однозначные, будет оцениваться в 20 баллов.

Решение, правильно работающее для случая, когда числа *A* и *B* не превосходят 100, будет оцениваться в 40 баллов.

Решение, правильно работающее для случая, когда числа *A* и *B* не превосходят 10000, будет оцениваться в 60 баллов.

Решение

Решение, в котором перебираются все числа от *A* до *B* и для каждого из них подсчитывается сумма цифр, не пройдет тесты при больших *A* и *B* ввиду наличия ограничения по времени, но может набрать частичные баллы.

Для полного решения задачи нужно заметить, что в каждом десятке (числа, отличающиеся только последней цифрой) ровно 5 чисел имеют четную сумму цифр.

Поэтому можно сначала циклом пройти от числа *A* до ближайшего числа, заканчивающегося цифрой 0, для каждого из этих чисел посчитать сумму цифр и увеличивать ответ на 1, если сумма цифр числа четная. Затем посчитать количество полных десятков до числа *B*, добавив к ответу это количество, умноженное на 5. Наконец, перебрать числа, начиная с последнего числа последнего десятка, увеличенного на 1, до *B*, и также найти сколько среди них имеет четную сумму цифр.

Пример правильного решения на языке Python версии 3:

```
def check(n): # Проверка четности суммы цифр числа n
    s = 0
    while n > 0:
        s += n % 10
        n //= 10
    return int(s % 2 == 0)
A = int(input())
```

```
B = int(input())
ans = 0
while A <= B and A % 10 != 0:
    ans += check(A)
    A += 1
if A < B:
    ans += (B - A) // 10 * 5
    A += (B - A) // 10 * 10
while A <= B:
    ans += check(A)
    A += 1
print(ans)
```

Частичные решения

Решения, в которых все числа от *A* до *B* перебираются в цикле и для каждого числа проверяется четность суммы цифр набирают 60 баллов.