

Всероссийская олимпиада школьников по информатике, 2019/20 уч. год  
Первый (школьный) этап, г. Москва  
Разбор заданий для 9–11 классов

Примеры решений приведены на языке Python версии 3.

## Задача 1. Часовые пояса

Таня решила позвонить своей подруге, но вспомнила, что та живёт очень далеко, поэтому в часовом поясе подруги может быть слишком поздно или рано. Часы у Тани показывают ровно  $H$  часов, Таня живёт в часовом поясе UTC+ $A$ , а её подруга – в часовом поясе UTC+ $B$ . Помогите Тане определить время в часовом поясе подруги в этот момент.

Программа получает на вход три целых числа  $H$ ,  $A$  и  $B$ ,  $0 \leq H \leq 23$ ,  $-11 \leq A \leq 12$ ,  $-11 \leq B \leq 12$ .

В часовом поясе UTC+ $A$  местное время больше, чем время в часовом поясе UTC+0 на  $A$  часов (если же  $A < 0$ , то меньше на  $|A|$  часов). Например, если в часовом поясе UTC+0 сейчас 12 часов, то в часовом поясе UTC+1 – 13 часов, а в часовом поясе UTC–1 – 11 часов.

Программа должна вывести одно число – время (количество часов) в часовом поясе подруги.

Под временем в этой задаче подразумевается количество часов, которое может принимать значения от 0 до 23. При решении задачи обратите внимание, что в часовом поясе подруги может быть уже следующая дата или предыдущая дата, программа должна вывести количество часов на часах подруги в этот момент, то есть число от 0 до 23.

### Пример входных и выходных данных

Ввод	Вывод	Примечание
15 3 –5	7	У Тани – 15 часов, она живёт в часовом поясе UTC+3. В часовом поясе UTC+0 сейчас 12 часов. Подруга живёт в часовом поясе UTC–5, и у неё сейчас 7 часов.

### Решение

Если в часовом поясе UTC+ $A$  время  $H$  часов, то в поясе UTC+0 время  $H - A$ , а в часовом поясе UTC+ $B$  время будет равно  $H - A + B$ . Но при этом не учитываются возможные переходы между сутками — время может оказаться в следующих сутках или в предыдущих сутках. Это можно учесть при помощи двух условий: если получили время, больше или равное 24 часа, то нужно вычесть 24 часа, а если получили отрицательное время, то нужно прибавить 24 часа. Получается следующее решение:

```
H = int(input())
A = int(input())
B = int(input())
answer = H - A + B
if answer >= 24:
    answer -= 24
if answer < 0:
    answer += 24
print(answer)
```

Но вместо условий проще взять остаток от деления на 24:

```
H = int(input())
A = int(input())
B = int(input())
answer = (H - A + B) % 24
print(answer)
```

Заметим, что в большинстве языков программирования (Pascal, C++, Java, C#) при взятии остатка от деления отрицательного числа на положительное получится отрицательный результат, то есть решение задачи будет неверным. В этом случае нужно действовать так (пример для Pascal).

```
answer := (H - A + B + 24) mod 24
```

## Задача 2. Чётные – нечётные

Маша любит чётные числа, а Миша – нечётные. Поэтому они всегда радуются, если встречаются числа, которые им нравятся.

Сегодня им встретились все целые числа от  $A$  до  $B$  включительно. Маша решила посчитать сумму всех чётных чисел от  $A$  до  $B$ , а Миша – сумму всех нечётных, после чего они начали спорить, у кого получилась сумма больше. Помогите им – найдите разность между суммой Маши и суммой Миши.

Программа получает на вход два целых положительных числа  $A$  и  $B$ , не превосходящие  $2 \times 10^9$ . Программа должна вывести одно число – разность между суммой чётных чисел и суммой нечётных чисел от  $A$  до  $B$ .

### Примеры входных и выходных данных

Ввод	Вывод	Примечание
3 6	2	Сумма чётных чисел равна $4 + 6 = 10$ , сумма нечётных чисел равна $3 + 5 = 8$ , разность равна 2.
3 7	-5	Сумма чётных чисел равна $4 + 6 = 10$ , сумма нечётных чисел равна $3 + 5 + 7 = 15$ , разность равна -5.

### Система оценивания

Решение, правильно работающее только для случаев, когда числа  $A$  и  $B$  не превосходят 100, будет оцениваться в 60 баллов.

### Решение

На 60 баллов можно перебрать все числа от  $A$  до  $B$  и сложить их со знаком «+» или «-» в зависимости от чётности. Пример решения.

```
a = int(input())
b = int(input())
s = 0
for i in range(a, b + 1):
    if i % 2 == 0:
        s += i
    else:
        s -= i
print(s)
```

На полный балл можно посчитать сумму всех чётных и нечётных чисел на отрезке, используя сумму арифметической прогрессии. А именно, если числа  $A$  и  $B$  оба чётные (или, наоборот, оба нечётные), то сумма всех чётных (или нечётных) чисел на отрезке от  $A$  до  $B$  равна  $(A + B) / 2 * ((B - A) / 2 + 1)$ . Но для того, чтобы использовать эту формулу, необходимо сделать границы отрезка одной чётности. Исправим границы отрезка – прибавим 1 к числу  $A$ , если  $A$  – нечётное, а также вычтем из  $B$  число 1, если  $B$  – нечётное. Получим два чётных значения  $a2$  и  $b2$ , соответствующих границам чётного отрезка, содержащего чётные числа от  $A$  до  $B$ . Аналогично, получим два нечётных значения  $a1$  и  $b1$ , соответствующих границам нечётного отрезка. Затем посчитаем сумму чётных чисел на отрезке от  $a2$  до  $b2$  и сумму нечётных чисел на отрезке от  $a1$  до  $b1$ .

```
a = int(input())
b = int(input())
a2 = a + a % 2
b2 = b - b % 2
even = (b2 + a2) // 2 * ((b2 - a2) // 2 + 1)
a1 = a + (1 - a % 2)
b1 = b - (1 - b % 2)
odd = (b1 + a1) // 2 * ((b1 - a1) // 2 + 1)
print(even - odd)
```

Заметим, что при реализации такого решения на языках Pascal, C++, Java, C# необходимо использовать 64-битные целочисленные переменные для хранения переменных `even` и `odd` во

избежани переполнения. Можно реализовать решение и без переполнения 32-битных чисел если заметить, что разница между двумя соседними числами равна 1, далее подсчитать количество пар чисел. Но в таком решении придётся разбирать несколько случаев (4 возможных случая четности каждой из двух границ), поэтому такое решение более сложно в реализации.

### Задача 3. Уточка

Как известно, при разработке и отладке программ большую помощь могут оказать игрушечные жёлтые уточки (см. статью «Метод утёнка» в википедии), поэтому Денис собрал большую коллекцию жёлтых уточек. Коллекция уже настолько большая, что Денис решил расставить уточек на полки шкафа. Сначала он начал ставить на каждую полку по  $A$  уточек, но одна уточка оказалась лишней. Тогда он заново начал расставлять уточек на полки, ставя на каждую полку по  $B$  уточек, но в этом случае ему не хватило одной уточки, чтобы на каждой полке оказалось ровно  $B$  уточек. Определите минимальное число уточек, которое могло быть в коллекции Дениса.

Программа получает на вход два целых положительных числа  $A$  и  $B$ ,  $2 \leq A \leq 2 \times 10^9$ ,  $2 \leq B \leq 2 \times 10^9$  – количество уточек при расстановке на полке в первом и во втором случаях. Программа должна вывести одно число – минимально возможное количество уточек в коллекции Дениса. Гарантируется, что ответ существует и не превосходит  $2 \times 10^9$ .

#### Пример входных и выходных данных

Ввод	Вывод	Примечание
5	11	$11 = 5 \times 2 + 1$
3		$11 = 3 \times 4 - 1$

#### Система оценивания

Решение, правильно работающее только для случаев, когда числа  $A$  и  $B$  не превосходят 100, будет оцениваться в 40 баллов.

#### Решение

На 40 баллов можно перебрать все числа, пока не будет найдено число, дающее остаток 1 при делении на  $A$  и дающее остаток  $B - 1$  при делении на  $B$ . Пример решения.

```
a = int(input())
b = int(input())
ans = 1
while ans % a != 1 or ans % b != (b - 1):
    ans += 1
print(ans)
```

Для ускорения этого решения можно перебирать не все числа, а только числа, дающие остаток 1 при делении на  $A$ . Поскольку мы начали с числа 1, которое даёт остаток 1 при делении на  $A$ , то дальше нужно прибавлять к числу значение  $A$  вместо 1. Достаточно в этом примере заменить `ans += 1` на `ans += a`, и решение будет набирать 70 баллов. Его можно ещё упростить, если убрать из цикла проверку того, что `ans` даёт остаток 1 при делении на  $a$ . Пример решения на 70 баллов.

```
a = int(input())
b = int(input())
ans = 1
while ans % b != (b - 1):
    ans += a
print(ans)
```

Аналогично перебирать числа, которые дают остаток  $b - 1$  при делении на  $b$ , если начать с числа  $b - 1$  и дальше идти с шагом  $b$ . Такое решение будет также набирать 70 баллов.

Для того, чтобы набрать 100 баллов, необходимо выбрать из чисел  $a$  и  $b$  наибольшее и выбрать шаг, равный большему из двух чисел.

```

a = int(input())
b = int(input())
if a >= b:
    ans = 1
    while ans % b != b - 1:
        ans += a
else:
    ans = b - 1
    while ans % a != 1:
        ans += b
print(ans)

```

Это решение корректно, поскольку в условии задачи говорится, что ответ существует и не превосходит  $2 \times 10^9$ , то есть при больших числах  $A$  и  $B$  количество шагов цикла не превосходит  $2 \times 10^9 / \max(A, B)$ .

## Задача 4. Неправильный палиндром

Палиндромом называется слово, которое одинаково читается как слева направо, так и справа налево, например, в английском языке такими словами являются «radar» и «gacesar».

Света изучает английский язык и решила принять участие в дистанционном конкурсе знатоков английского языка. Но, когда она писала ответ на задание «найдите самое длинное слово, которое является палиндромом», ошиблась и нажала на клавиатуре одну лишнюю клавишу. Определите, какую букву нужно удалить в набранном Светой слове, чтобы это слово стало палиндромом.

Программа получает на вход строку из строчных английских букв, содержащую не менее 2 и не более 100 000 символов.

Программа должна вывести единственное число – номер буквы в строке, при удалении которой слово становится палиндромом. Если при удалении любой буквы слово не станет палиндромом, программа должна вывести число 0.

### Примеры входных и выходных данных

Ввод	Вывод
raceczar	6
car	0

### Система оценивания

Решение, правильно работающее только для случаев, когда длина строки не превосходит 100 символов, будет оцениваться в 60 баллов.

### Решение

Для начала напишем простой перебор всех возможных ответов: переберём все символы строки, удалим их, если после удаления символа получится палиндром, выведем номер этого символа. Если не нашлось такого символа, выведем 0. Пример решения.

```

s = input()
for i in range(len(s)):
    if s[:i] + s[i + 1:] == (s[:i] + s[i + 1:])[::-1]:
        print(i + 1)
        break
else:
    print(0)

```

В этом решении используется ряд удобств языка Python, поэтому объясним подробнее.  $s[:i]$  – это срез строки: подстрока, состоящая из символов до  $i$ -го, не включая  $i$ -й символ.  $s[i + 1:]$  – это срез, от символа с индексом  $i + 1$  до конца строки.  $s[:i] + s[i + 1:]$  – это конкатенация двух срезов, то есть исходная строка, без символа с индексом  $i$ .  $[::-1]$  – это срез из символов строки, взятых с шагом  $-1$ , то есть строка, развёрнутая в обратном порядке, то есть в инструкции `if`

проверяется, верно ли, что после выбрасывания символа с индексом  $i$  получается палиндром. Программа выводит  $i + 1$  потому что в языке Python индексация элементов строки начинается с 0, а нужно вывести ответ в нумерации символов с 1.

Инструкция `else` относится к циклу `for`, она выполняется после завершения цикла. Если цикл был прерван по инструкции `break`, то `else` не выполняется, поэтому число 0 будет напечатано, только если не был найден ответ ранее.

Поскольку проверка на палиндромность осуществляется за время, пропорциональное длине строки (нужно развернуть строку и сравнить две строки), то такое решение будет иметь сложность  $O(n^2)$ , где  $n$  – длина строки, и будет набирать 60 баллов.

Приведём решение, имеющее сложность  $O(n)$ , набирающее полный балл. Сравним первый и последний символ. Если они различаются, то, значит, один из них должен быть удалён. Если они равны, то сравним второй и предпоследний символ, если они равны, то перейдём к следующим от концов строки символам и т.д. Нам необходимо найти такие два символа, отстоящие на равное расстояние от концов строки, которые различаются. Один из этих символов должен быть удалён. Проверим, что после удаления одного из этих символов, строка становится палиндромом, то есть ответ найден. Если же при удалении каждого из этих двух символов строка не становится палиндромом, то задача не имеет решения и нужно вывести 0.

Наконец, если такие символы не были найдены, то строка уже является палиндромом. В этом случае нужно удалить средний символ в строке (в случае строки чётной длины – один из двух средних символов). Пример решения.

```
s = input()
i = 0
j = len(s) - 1
while i < j and s[i] == s[j]:
    i += 1 # Поиск двух различных символов i и j,
    j -= 1 # равноудалённых от концов строки
if i >= j:
    print(i + 1) # строка - уже палиндром, выводим центральный символ
elif s[:i] + s[i + 1:] == (s[:i] + s[i + 1:])[::-1]:
    print(i + 1) # можно удалить символ i
elif s[:j] + s[j + 1:] == (s[:j] + s[j + 1:])[::-1]:
    print(j + 1) # можно удалить символ j
else:
    print(0) # решения нет
```

## Задача 5. Квест

Новый квест, в котором участники должны выбраться с территории проведения, представляет собой прямоугольник из  $N \times M$  комнат. Каждая комната имеет четыре двери, ведущие в соседние комнаты, из комнат на краю прямоугольника двери ведут наружу, через эти двери можно покинуть территорию проведения квеста.

В начале квеста в каждой комнате находится по человеку, а все двери заперты. После начала квеста организаторы дистанционно открывают в каждой комнате запирающий механизм одной из четырёх дверей. Теперь человек, находящийся в этой комнате, может открыть эту дверь и перейти в соседнюю комнату, через другие три двери выйти из этой комнаты нельзя. При этом может оказаться так, что дверь, соединяющая две комнаты, будет отпираться только с одной стороны, тогда пройти через эту дверь можно только с той стороны, с которой она будет открываться, проходить через дверь в обратном направлении нельзя, если в соседней комнате будет отперта не эта дверь, а какая-то другая. Если комната находится на краю территории и из этой комнаты открыта дверь наружу, то, пройдя через эту дверь, участник навсегда покидает территорию квеста.

После начала квеста и отпирания дверей участники начинают перемещаться между комнатами. Каждый участник перемещается в соседнюю открытую комнату и продолжает перемещаться до тех пор, пока не покинет территорию квеста. Однако возможна ситуация, когда некоторые участники будут бесконечно перемещаться между комнатами и никогда не выйдут наружу.

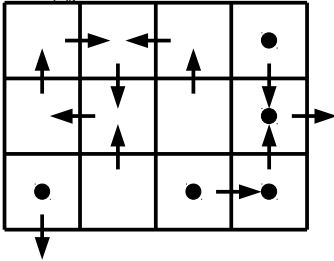
Разработчики квеста попросили Вас составить такой план отпирания дверей, при котором ровно

$K$  человек смогут выбраться наружу с территории квеста.

Программа получает на вход три числа  $N, M, K, 1 \leq N \leq 100, 1 \leq M \leq 100, 0 \leq K \leq NM$ .  $N$  и  $M$  – количество строк и столбцов в прямоугольном плане квеста,  $K$  – количество человек, которые должны выйти из квеста.

Программа должна вывести план территории квеста в виде  $N$  строк, каждая из которых должна содержать  $M$  символов. Символ соответствует тому, какая дверь будет открыта в данной комнате и может быть одной из следующих заглавных английских букв: U (дверь в верхнюю по данному плану комнату), D (дверь в нижнюю комнату), L (дверь в левую комнату), R (дверь в правую комнату). Необходимо вывести один любой подходящий план решения задачи. Если ни одного подходящего плана не существует, программа должна вывести одну строчку «IMPOSSIBLE».

### Примеры входных и выходных данных

Ввод	Вывод	Примечание
1 2 1	IMPOSSIBLE	Территория квеста состоит из 1 строки и 2 столбцов. При любом способе открытия дверей из квеста не сможет выбраться ровно один человек.
3 4 5	RDLD ULUR DURU	Территория квеста состоит из 3 строк и 4 столбцов. Из квеста должны выйти 5 человек. На рисунке ниже приведена картинка, соответствующая ответу из примера. Стрелками обозначены открытые двери, точками помечены комнаты, обитатели которых выйдут из квеста. 

### Система оценивания

Решение, правильно работающее только для случая  $N = 1$ , будет оцениваться в 30 баллов.

### Решение

Люди, которые не смогли выйти из квеста, должны ходить по циклу. Цикл должен содержать как минимум 2 комнаты (нельзя организовать цикл из одной комнаты), поэтому решения нет в случае, когда  $K = NM + 1$ . Если  $K = NM$ , то все люди выходят из квеста, например, пусть все движутся вправо. Если же  $K \leq NM - 2$ , то решение существует. Для этого организуем цикл из двух самых левых клеток в самой первой строке:

```
RL.....
.....
.....
.....
```

Затем добавим людей, которые должны остаться в квесте. Они должны прийти в этот цикл. Люди из первой строки будут двигаться влево, чтобы прийти в цикл.

```
RLLLLL..
.....
.....
.....
```

Если необходимо добавить ещё людей, которые должны прийти в цикл, то из следующих строк люди будут двигаться вверх, тогда они придут в цикл.

```
RLLLLLLL
UUUUUUUU
UUU.....
.....
```

В приведённом выше примере в цикл приходят все люди из первой и второй строки, и три человека из третьей строки. Таким образом можно привести в цикл любое необходимое число людей. Оставшиеся люди должны выйти из квеста, пусть они двигаются вправо.

```
RLLLLLLL
UUUUUUUU
UUURRRRR
RRRRRRRR
```

Далее необходимо реализовать программу, которая выводит подобную конструкцию. Отдельно нужно разобрать случай  $M = 1$ , то есть план квеста состоит из одного столбца, тогда цикл необходимо разместить вертикально.

Пример решения.

```
import sys
n = int(input())
m = int(input())
k = int(input())

if k == n * m - 1:
    print("IMPOSSIBLE")
    sys.exit(0)
if k == n * m:
    print(("R" * m + "\n") * n)
    sys.exit(0)
if m == 1:
    print("D")
    for i in range(n * m - k - 1):
        print("U")
    for i in range(k):
        print("R")
    sys.exit(0)

for i in range(n):
    for j in range(m):
        if i * m + j < n * m - k:
            if i == 0:
                if j == 0:
                    print('R', end='')
                else:
                    print('L', end='')
            else:
                print('U', end='')
        else:
            print('R', end = '')
    print()
```