

## Разбор задач

## Задача 1. Прикольные числа

Несложно заметить, что выполнить все критерии невозможно, т.к. если в числе есть цифра 8 и две пары различных цифр, то в числе будут две цифры 8 и тогда сумма цифр не может быть равна 14. Значит, нужно рассмотреть числа, для которых выполнялись бы три критерия из четырёх.

Чтобы получить наименьшее число, на первое место поставим цифру 1, на второе — 0. Осталось добавить ещё две цифры, отказавшись от одного правила. Последнее правило сложно выполнить, т.к. в этом случае нам придётся добавить ещё одну цифру 1 и цифру 0, тогда не будут выполнены правила 1 и 2. Откажемся от этого правила, тогда нужно добавить две цифры, одна из которых — 8, а сумма всех цифр равна 14. Значит, ещё одна цифра равна 5. Минимальное подходящее число — 1058.

Чтобы получить наибольшее число, на первое место поставим цифру 9. На второе место нельзя поставить ещё одну цифру 9, т.к. тогда перестанут выполняться правила 2 и 3. Поставим на второе место цифру 8. Теперь чтобы выполнить правила 3 и 4 (соседние цифры должны различаться, в числе две пары равных цифр) добавим ещё цифры 9 и 8. Наибольшее подходящее число — 9898.

## Задача 2. Собрание сочинений

В последовательности томов 6 2 7 8 4 5 1 9 3 пары томов 7 8 и 4 5 уже стоят рядом, поэтому эти пары не надо разделять и они всегда будут переставляться вместе. Все остальные пары соседних томов нужно будет разделить, и тогда последовательность томов разбивается на фрагменты

$$6 - 2 - 7\ 8 - 4\ 5 - 1 - 9 - 3$$

Всего 7 фрагментов. Из них можно выбрать три фрагмента, уже идущих по возрастанию номеров томов: 2 — 4 5 — 9 (получится выбрать и другие три фрагмента, но не четыре фрагмента, идущих по возрастанию). Оставшиеся четыре фрагмента необходимо вставить в требуемом порядке между тремя уже упорядоченными фрагментами, для этого понадобятся четыре операции. За три операции задачу решить нельзя.

Пример решения (оставим для наглядности знаки «—», но в ответе их быть не должно).

$$6 - 2 - 7\ 8 - 4\ 5 - 1 - 9 - 3$$

$$2 - 7\ 8 - 4\ 5 - 6 - 1 - 9 - 3$$

$$2 - 4\ 5 - 6 - 7\ 8 - 1 - 9 - 3$$

$$1 - 2 - 4\ 5 - 6 - 7\ 8 - 9 - 3$$

$$1 - 2 - 3 - 4\ 5 - 6 - 7\ 8 - 9$$

## Задача 3. Сигнализация

В данной задаче предлагается реализовать логическую функцию, известную как функция голосования, истинную тогда и только тогда, когда хотя бы два из трёх входных параметров истинны.

Для начала можно сформировать условия для отдельных строк таблицы истинности и объединить их через «ИЛИ»:  $(x \text{ and } y \text{ and } z) \text{ or } (x \text{ and } y) \text{ or } (x \text{ and } z) \text{ or } (y \text{ and } z)$ . Такой подход, требующий 8 операций, оценивался в 20 баллов.

Дальше можно заметить, что первое слагаемое выражения избыточно — если оно истинно, то истинны и все остальные. Поэтому его можно исключить:  $(x \text{ and } y) \text{ or } (x \text{ and } z) \text{ or } (y \text{ and } z)$ . Такой подход с 5 операциями оценивался в 50 баллов.

Для получения оптимального ответа нужно рассмотреть второе и третье слагаемые и «вынести» из них  $z$ :  $(x \text{ and } y) \text{ or } (z \text{ and } (x \text{ or } y))$ . Такой подход требует 4 операций, получая 100 баллов. Можно вынести и другую переменную, например,  $x$  или  $y$ .

## Задача 4. Рекламные паузы

К длине фильма нужно добавить количество рекламных пауз, умноженное на продолжительность одной паузы  $b$ . Рекламные паузы вставляются через каждые  $a$  минут, поэтому для подсчёта их числа нужно поделить  $n$  нацело на  $a$ . Но если  $n$  делится на  $a$  (остаток от деления  $n$  на  $a$  равен 0), то время показа последней рекламной паузы совпадёт с окончанием фильма, поэтому реклама показываться не будет и число рекламных пауз нужно уменьшить на 1. Возможное решение.

```
a = int(input())
b = int(input())
n = int(input())
n_pauses = n // a
if n % a == 0:
    n_pauses -= 1
print(n + n_pauses * b)
```

Можно написать решение и без использования `if`.

```
a = int(input())
b = int(input())
n = int(input())
n_pauses = (n - 1) // a
print(n + n_pauses * b)
```

## Задача 5. Популярный пост

Если какое-то из чисел  $a$ ,  $b$ ,  $c$  больше суммы двух других, то это число и является ответом: меньшее количество пользователей не могло оставить столько реакций, а если каждый пользователь поставил самую популярную реакцию из трёх и какую-то из двух оставшихся (или поставил одну самую популярную реакцию), то этот вариант будет реализован. Поэтому можно просто вывести наибольшее из трёх чисел.

```
a = int(input())
b = int(input())
c = int(input())
print(max(a, b, c))
```

Такое решение набирает 50 баллов.

Рассмотрим теперь случай, когда максимальное из трёх чисел меньше суммы двух других. Необходимо наименьшему количеству пользователей назначить по две или одной реакции, чтобы общее число реакций трёх видов было равно  $a$ ,  $b$ ,  $c$ . Реализуем «жадный» алгоритм — назначим очередному пользователю две различные реакции, уменьшив на 1 счётчики их количеств. Чтобы последующим пользователям также оставалась возможность выбрать две реакции, необходимо избежать ситуации, когда только одно из чисел  $a$ ,  $b$ ,  $c$  является ненулевым. то есть выбирать на каждом шаге два наибольших значения из  $a$ ,  $b$ ,  $c$ .

Приведём пример такого решения: входные числа упорядочиваются так, чтобы выполнялось условие  $a \leq b \leq c$ , затем в цикле наибольшие значения  $c$  и  $b$  уменьшаются на 1, а значение ответа увеличивается на 1, после чего числа опять переупорядочиваются.

```
a = int(input())
b = int(input())
c = int(input())
ans = 0
```

```

a, b, c = sorted([a, b, c])
while c > 0:
    c -= 1
    b -= 1
    ans += 1
    a, b, c = sorted([a, b, c])
print(ans)

```

Такое решение набирает 50 баллов. Но если объединить эти два решения и выводить  $\max(a, b, c)$  в случае, когда одно из чисел не меньше суммы двух других, а иначе реализовать «жадный» алгоритм, то решение получит 75 баллов.

Теперь избавимся от цикла в «жадном» алгоритме. Поскольку пользователь не мог оставить более двух реакций, то для количества пользователей  $k$  выполняется неравенство

$$k \geq \left\lceil \frac{a+b+c}{2} \right\rceil,$$

где  $\lceil x \rceil$  — значение  $x$ , округлённое вверх до целого числа.

Наш жадный алгоритм и реализует такую формулу: на каждом шаге, кроме, возможно, последнего, будут уменьшаться два из трёх чисел  $a, b, c$ .

Таким образом, решение задачи дают формулы:  $k = \max(a, b, c)$  если одно из чисел не меньше суммы двух других (то есть если  $2 \max(a, b, c) \geq a + b + c$ ) или  $k \lceil \frac{a+b+c}{2} \rceil$  иначе. Это можно записать и при помощи одной формулы

$$k = \max\left(a, b, c, \left\lceil \frac{a+b+c}{2} \right\rceil\right).$$

Деление с округлением вверх можно реализовать, например, на языке Python при помощи выражения  $(a+b+c+1) // 2$ .

Пример такого решения.

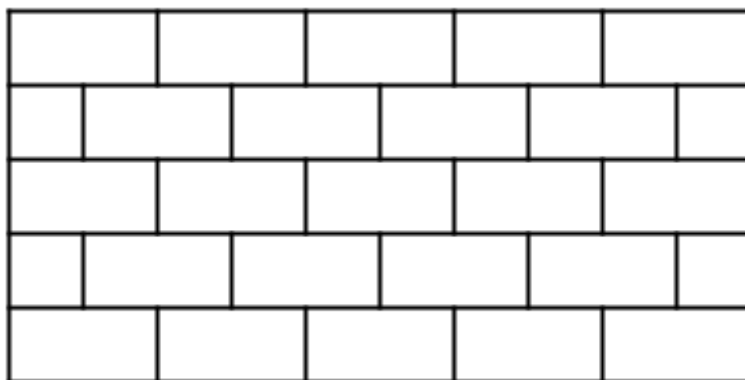
```

a = int(input())
b = int(input())
c = int(input())
print(max(a, b, c, (a+b+c+1) // 2))

```

## Задача 6. Раскраска стены

Для покраски всегда достаточно трёх цветов. Рассмотрим достаточно большой фрагмент стены и попробуем покрасить его в три цвета.



Начнём с целого кирпича в левом нижнем углу. Если три кирпича попарно соприкасаются, то они будут покрашены в разные цвета.

|   |   |   |   |  |  |
|---|---|---|---|--|--|
|   |   |   |   |  |  |
|   |   |   |   |  |  |
|   |   |   |   |  |  |
|   | 3 | 3 |   |  |  |
| 1 | 1 | 2 | 2 |  |  |

Продолжим, покрасив в три цвета два нижних ряда кирпичей.

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |
|   | 3 | 3 | 1 | 1 | 2 | 2 | 3 | 3 |   |
| 1 | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 |

Остальные ряды будут повторением двух нижних рядов.

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 |
| 2 | 3 | 3 | 1 | 1 | 2 | 2 | 3 | 3 | 1 |
| 1 | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 |
| 2 | 3 | 3 | 1 | 1 | 2 | 2 | 3 | 3 | 1 |
| 1 | 1 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 2 |

Видим, что в нижнем и всех нечётных рядах повторяется последовательность цифр 112233, а во втором и других чётных рядах — та же последовательность, но со сдвигом на 3 цифры. Решение будет таким: создадим строку, повторив последовательность 112233 несколько раз. Для построения нечётных (считая снизу) рядов кирпичей выведем первые  $m$  символов этой строки, а в чётных рядах —  $m$  символов, но пропустив 3 начальных символа последовательности.

Есть ещё один частный случай. При  $n = 1$  для раскраски достаточно двух цветов, т.к. ряд кирпичей всего один. В этом случае нужно образовать строку повторением последовательности 1122. При  $n = 1$ ,  $m = 2$  нужен только один цвет (т.к. кирпич всего один), но этот случай можно не рассматривать отдельно.

Пример правильного решения.

```
n = int(input())
m = int(input())
if n > 1:
    s = "112233" * 4
else:
    s = "1122" * 5
for i in range(n, 0, -1):
    if i % 2 == 1:
        print(s[:m])
    else:
        print(s[3:3 + m])
```