Задача A. Пчелы на зимовку

Если количество *правильных* пчёл делится на 3, то для них можно построить трехместные ульи и один двухместный для *неправильных*.

Если количество *правильных* имеет остаток 2 при делении на 3, то эти две пчелы селим в двухместный, так же в двухместный селим *неправильных* пчёл, а всех остальных в трехместные.

Если количество *правильных* имеет остаток 1 при делении на 3, то эти четырех пчёл селим в двуместный, так же в двухместный селим *неправильных* пчёл, а всех остальных в трехместные.

Авторское решение на языке Python:

```
n = int(input())
if n % 3 == 0:
    print(1, n // 3)
elif n % 3 == 2:
    print(2, (n - 2) // 3)
else:
    print(3, (n - 4) // 3)
```

Задача B. Параллелепипед

Отсортируем стороны A, B, C в порядке возрастания.

Рассмотрим возможные случаи:

1) $A = 1$, $B = 1$, $C = 1$. Один кубик, у которого все шесть граней грязные.

2) $A = 1$, $B = 1$, $C > 1$. Длинная полоска, у крайних кубиков по пять грязных граней, у остальных $C - 2$ кубиков по четыре.

3) $A = 1$, $B > 1$, $C > 1$. Прямоугольник высоты 1. У угловых кубиков по четыре грязные грани, у других крайних (их всего $2B + 2C - 4$, а без угловых $2B + 2C - 8$) по три грязные стороны, у остальных по две грязные стороны, их $(B - 2)(C - 2)$.

4) $A > 1$, $B > 1$, $C > 1$. Общий случай, у нас восемь угловых кубиков с тремя грязными сторонами. Две грязные стороны у кубиков, которые расположены вдоль ребер исходного параллелепипеда, их всего двенадцать, по 4 ребра каждой из длин, итого $4(A - 2) + 4(B - 2) + 4(C - 2)$ кубиков (вычитаем два для каждого ребра, потому что угловые имеют три грязные грани). По одной грязной стороне имеют кубики, расположенные внутри каждой из граней исходного параллелепипеда, это прямоугольники, где каждая сторона уменьшена на 2 относительно исходного параллелепипеда. Прямоугольник каждого размера два раза встречается на параллелепипеде (противоположные грани). Итого $2(A-2)(B-2)+2(A-2)(C-2)+2(B-2)(C-2)$. Чтобы посчитать сколько кубиком не замарались, можно просто снять верхнюю оболочку параллелепипеда, получится новый, где каждая сторона уменьшена на 2. В нем $(A - 2)(B - 2)(C - 2)$ кубиков.

Авторское решение на языке Python:

```
a = int(input())
b = int(input())
c = int(input())
if a > b:
    a, b = b, a
if b > c:
    c, b = b, c
if a > b:
    a, b = b, a
if a == 1:
    if b == 1:
        if c == 1:
            print(0, 0, 0, 0, 0, 0, 1)
        else:
            print(0, 0, 0, 0, c - 2, 2, 0)
    else:
        print(0, 0, (c - 2) * (b - 2), (c - 2 + b - 2) * 2, 4, 0, 0)
```

```
else:
    print((c - 2) * (b - 2) * (a - 2), (c - 2) * (b - 2) * 2 + (c - 2) * (a - 2) * 2
        + (a - 2) * (b - 2) * 2, (a - 2 + b - 2 + c - 2) * 4, 8, 0, 0, 0)
```

Задача C. Битва с драконом

Заметим, что каждый день здоровье дракона A меняется по правилу $A := max(0, min(D, A \cdot B - C))$.

Подгруппа 1.

Для решения подгруппы с $K \leqslant 1000$ можно честно промоделировать все K дней.

```
for i in range(K):
   A = max(0, min(D, A * B - C))
print(A)
```

Подгруппа 2.

Для решения подгруппы с $B = 1$ нужно вывести $max(0, A - C \cdot K)$.

Подгруппа 3.

Для решения всей задачи улучшим алгоритм из первой подгруппы, а именно поймём, при каких условиях можно закончить цикл, так как мы уже однозначно знаем ответ.

- Если $A = 0$, то и ответ будет равен 0.
- Если $A = max(0, min(D, A \cdot B - C))$, то мы пришли в «цикл» и здоровье дракона A за оставшиеся дни никак не поменяется.
- Если $A = D$ и прошёл хотя бы 1 день. Утверждается, что тогда ответ тоже будет D. Почему? Рассмотрим $A \cdot B - C \geqslant A \Leftrightarrow A \cdot (B - 1) \geqslant C$. Если $B = 1$, то либо $C = 0$ и A не меняется, либо $C \geqslant 1$, и A будет только уменьшаться, то есть $A < D$ после первого дня. Тогда если $B > 1$, то при $A > \frac{C}{B-1}$ здоровье дракона будет увеличиваться. То есть если A больше какого-то значения, то оно всегда будет увеличиваться, значит, если до какого-то вечера A стало больше или равно D, то и дальше оно не станет уменьшаться, но все равно не сможет стать больше D.

Всё это значит, что не более чем за D дней мы придём к одному из трёх исходов, значит наш алгоритм будет работать за $O(D)$.

Получаем окончательное решение:

```
for i in range(K):
   A = max(0, min(A * B - C, D))
   if A == 0 or A == D or A * B - C == A:
      break
print(A)
```

Задача D. Очень интересная игра с массивом

Давайте для каждого элемента предподсчитаем, сколько элементов левее него с таким же значением, и сохраним эти значения. Также посчитаем, сколько у нас каждый элемент всего встречается раз.

Как это будем делать? Во время прохода будем поддерживать количество каждого числа на префиксе в словаре (будем делать это структурах данных map в c++ или dict в python). В ответ для каждой позиции занесем значение из словаря.

Таким образом, зная для каждой позиции, сколько элементов слева и сколько их в массиве всего, мы можем с легкостью отвечать на каждый запрос.

Авторское решение на языке Python:

```
n = int(input())
k = int(input())

a = [0] * n
cnt = {}
```

```
before = [0] * len(a)
for i in range(n):
    a[i] = int(input())
    before[i] = cnt.setdefault(a[i], 0)
    cnt[a[i]] += 1

for i in range(k):
    x = int(input())
    x -= 1
    print(before[x], cnt[a[x]] - 1 - before[x])
```

Задача E. Гена и числа

Переберём сумму цифр в искомых числах от 1 до $9n$: для фиксированной суммы сконструируем максимальное и минимальное возможное число с такой суммой. Максимальное число строится таким образом: ставим 9, пока можем, затем дописываем остаток, а дальше 0.

Минимальное число строится похожим образом: если последняя цифра в максимальном числе не 0, то минимальное — это развёрнутое максимальное. Иначе из последней ненулевой цифры вычтем 1 и прибавим к последней цифре, а затем развернём. Обновим ответ разностью максимального и минимального числа. Итоговая асимптотика — $\mathcal{O}(n^2)$.

В решении далее минимальное число построено по аналогу с максимальным. Первой ставим единицу (если это невозможно, то такая пара нас не интересует), в конце поставим много девяток ((sum-1)/9) штук, чтобы получить нужную сумму цифр, после единицы поставим нужное число нолей, чтобы количество цифр было равно n, и в середину еще одну цифру, чтобы получить в точности нужную сумму цифр.

Авторское решение на Python:

```
n = int(input())
ans = 0
if n == 1:
    print(0)
    exit()
for sum in range(1, 9 * n - 7):
    cnt9 = sum // 9
    cnt0 = n - cnt9 - 1
    big = int("9" * cnt9 + str(sum % 9) + "0" * cnt0)

    cnt9 = (sum - 1) // 9
    cnt0 = max(0, n - cnt9 - 2)
    cnt = min(n - cnt9 - 1, 1)
    small = int("1" + "0" * cnt0 + str((sum - 1) % 9) * cnt + "9" * cnt9)
    ans = max(ans, big - small)
print(ans)
```

Задача F. Петя и лотерея

Пусть в числе цифры a и b идут подряд, при этом $a < b$. Тогда, вычеркнув цифру a, мы увеличим число.

С помощью этого факта можно решать задачу с помощью жадного алгоритма:

1. Пусть мы уже вычеркнули x цифр из числа так, что число является максимально возможным.
2. Найдем самое левое вхождение подходящей пары цифр a,b. Если мы вычеркнем цифру a, то полученное число станет оптимальным ответом после вычеркивания $x + 1$-й цифры.

Повторим эту операцию k раз. Если в какой-то момент подходящей пары нет, то цифры в числе не возрастают. В таком случае необходимо вычеркнуть последнюю цифру.

Наивное решение может быть реализовано за $O(k \cdot n)$.

Для решения за $O(n)$ будем итерировать по цифрам в числе слева направо, поддерживаю текущий стек s следующим образом:

- Пока текущая цифра больше, чем последний элемент $last$ в s, то вычеркнем из числа $last$ и удалим его из стека.
- Затем добавим в конец s текущую цифру.

Авторское решение на языке Python:

```
s = input()
k = int(input())
new_len = len(s) - k
st = ['z']
for c in s:
    while k > 0 and st[-1] < c:
        st.pop()
        k -= 1
    st.append(c)
print(''.join(st[1: new_len + 1]))
```